ICT-318784 STP TROPIC

Distributed computing, storage and radio resource allocation over cooperative femtocells

D51

System level aspects of TROPIC femto-clouding

Contractual Date of Delivery to the CEC:	**1st November 2014**
Actual Date of Delivery to the CEC:	**26th December 2014**
Author(s):	**Zdenek Becvar, Michal Vondra, Jan Plachy, Matej Rohlik (CTU), Sergio Barbarossa, Stefania Sardellitti, Paolo Di Lorenzo, Andrea Baiocchi (SAP), Olga Muñoz, Antonio Pascual, Josep Vidal (UPC), Miguel Angel Puente, Felicia Lobillo Vilela (ATOS), Enrico de Marinis, Fabrizio Gambetti (DUNE), Francesco Lo Presti (CINI), Hadi Hariyanto, Anton Herutomo, Lunel Chandra, Fidar A. Laksono, F.X. Ari Wibowo (TELK), Mariana Goldhamer (4GC), Mireille Sarkiss, Wael Labidi, Mohamed Kamoun, Jessica Oueis, Emilio Calvanese Strinati (CEA)**
Participant(s):	**CTU (Editor), SAP, UPC, ATOS, DUNE, CINI, TELK, 4GC, CEA**
Workpackage:	**5**
Est. person months:	**80**
Security:	**Internal**
Dissemination Level:	**PU**
Version:	**a**
Total number of pages:	**286**

Abstract:

This deliverable is focused on implementation issues of the small cells cloud from the modeling and system level point of view. The document provides models required for evaluation of the overall system performance as well as abstraction models for system implementation in frame of WP6. Particularly, models for various types of backhaul used in small cells are derived from the measurement carried out in a test bed of operator. In addition, abstraction models of physical layer, application, and small cells are proposed and described. Besides models, also system level aspects and system overview including interaction among cloud and radio parts of small cell cloud are addressed. It includes definition of final architecture and functionalities of the small cell cloud system. Then, algorithms and protocols for the application offloading from a user equipment to the small cells cloud are presented. Furthermore, aspects related to mobility and solutions for management of mobile users are outlined.

Keyword list: models, offloading, optimization, architecture, small cell cloud, interfaces, mobility.

Document Revision History

DATE	ISSUE	AUTHOR	SUMMARY OF MAIN CHANGES
26-12-2014	a	All	Firts release of the deliverable

Executive Summary

This document provides final outcomes of the TROPIC project at the system level and defines final solutions to be adopted by WP6 for proof of concept. Based on previous work done in TROPIC, major **assumptions and technical scenarios** considered for offloading of computation from UEs to the SCC are defined in **Section 2**.

To enable proper decision about offloading, it is necessary to define **models of individual entities and parts of the small cell cloud** including applications. The models of network nodes and physical layer are based on data and information available in literature and others developed in previous WPs in the project. For backhauling system, long-term measurements in network of operator are exploited to define model of backhaul. All models are described in **Section 3**.

The SCC merges mobile network and cloud computing into one concept. This requires modification of **functionalities and interfaces of the network nodes and entities**. The overall functional scheme of the SCC including new modules and interaction among them is drafted in **Section 4**. In this section, also **final architecture of the SCC** for TROPIC is summarized and potential enhancements of the SCC management for advanced architectures, defined in D22, are depicted. The SCC suitable architecture enhances the network with Small Cell Cloud Management (SCM) deployed in the core network. To enable efficient computation, legacy SCeNBs must be enhanced with computing CPU and with gateway enabling differentiation of SCC-related traffic from conventional traffic.

Key part of the **offloading process** is to decide whether to offload an application or not (and in case of the offloading, whether the whole processing or only a part of it should be offloaded). Many different metrics, parameters, and scenarios can influence the offloading decision. To that end, several offloading decision algorithms are defined and evaluated in **Section 5** to cover all possible scenarios and specific needs of different types of applications. This section also provides comprehensive summary and comparison of all investigated algorithms together with recommendations for implementation in WP6.

The SCC has been designed with focus on static or nomadic users. However, mobility is native part of the mobile networks. Impact of mobility on the SCC is addressed in **Section 6**, where possible mobility scenarios are analyzed and appropriate **solutions for mobility handling** are designed beyond original scope of the project.

Section 7 provides major interface towards WP6 and summarizes **recommendations for implementation and demonstration** of the SCC with specific focus on selection of algorithms for system level simulator and system architecture for the simulator and demonstration.

The major **conclusions** and high-level summary of whole document is given in **Section 8**.

DISCLAIMER

The work associated with this report has been carried out in accordance with the highest technical standards and the TROPIC partners have endeavored to achieve the degree of accuracy and reliability appropriate to the work in question. However since the partners have no control over the use to which the information contained within the report is to be put by any other party, any other such party shall be deemed to have satisfied itself as to the suitability and reliability of the information in relation to any particular use, purpose or application.

Under no circumstances will any of the partners, their servants, employees or agents accept any liability whatsoever arising out of any error or inaccuracy contained in this report (or any further consolidation, summary, publication or dissemination of the information contained within this report) and/or the connected work and disclaim all liability for any loss, damage, expenses, claims or infringement of third party rights.

Table of Contents

References

[3GPP TS 23.829] 3GPP TR 23.829, “Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO)”, v10.0.1, November 2011.

[3GPP TS 24.301] 3GPP TS 24.301, “Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3”, v12.6.0, September 2014.

[3GPP TS 29.281] 3GPP TS 29.281, “General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)”, v12.0.0, September 2014.

[3GPP TS 36.300] 3GPP TS 36.300, “Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2”, v12.3.0. September 2014.

[3GPP TR 36.922] 3GPP TR 36.922 V9.1.0. Evolved Universal Terrestrial Radio Access (E-UTRA); TDD Home eNode B (HeNB) Radio Frequency (RF) requirements analysis (Release 9). *Technical report, 3rd Generation Partnership Project*, 2010.

[3GPP TR 37.801] 3GPP TR 37.801, “UMTS-LTE 3500MHz Work Item Technical Report”, Technical Specification Group Radio Access Networks.

[3GPP TR 36.814] 3GPP TR 36.814 V9.0.0 “Further advancements for E-UTRA physical layer aspects”, March 2010.

[Al-Karaki04] J.N. Al-Karaki, A.E. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications, IEEE*, vol.11, no.6, pp.6,28, Dec. 2004.

[Alliance12] NGMN Alliance, “Small Cell Backhaul Requirement”. The Engine of Broadband Wireless Innovation, 2012.

[Altman99] E. Altman, “Constrained Markov Decision Processes”. *Chapman and Hall/CRC Press*, 1999.

[Barb-Sard-PDL13] S. Barbarossa, S. Sardellitti, P. Di Lorenzo, “Joint Allocation of Computation and Communication Resources in Multiuser Mobile Cloud Computing,” in Proc. of the 14th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Darmstadt, Germany, 16-19 June 2013.

[Barb13] S. Barbarossa, S. Sardellitti, P. Di Lorenzo, “Computation Offloading for Mobile Cloud Computing Based on Wide Cross-Layer Optimization,” in Proc. of the Future Network & Mobile Summit (FuNeMS), Lisbon, Portugal, 03-05 July 2013.

[Barbarossa14] S. Barbarossa, P. Di Lorenzo, S. Sardellitti, “Computation Offloading Strategies based on Energy Minimization under Computational Rate Constraints,” in Proc. of European Conference on Networks and Communications (EuCNC), Bologna, June 2014.

[BarbSardDiLor] S. Barbarossa, S. Sardellitti, P. Di Lorenzo, “Communicating while Computing: Distributed Cloud Computing over 5G Heterogeneous Networks,” IEEE Signal Processing Magazine, pp. 45-55, vol. 31, no. 3, November 2014.

[Becvar14] Z. Becvar, J. Plachy, P. Mach, "Path Selection Using Handover in Mobile Networks with Cloud-enabled Small Cells," IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2014), Washington, USA, September 2014.

[Bertsekas07] D. P. Bertsekas, “Dynamic programming and Optimal control”, *3rd ed. Athena Scientific*, 2007.

[Boyd04] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge Univ. Press, 2004.

[Broadband11] Think Broadband, “UK Broadband looking to enable LTE by 2012”, June 2011.

[Callahan90] D. Callahan, A. Carle, M.W. Hall, K. Kennedy, “Constructing the procedure call multigraph,” *IEEE Transactions on Software Engineering*, vol.16, no.4, pp.483-487, Apr. 1990.

[Cale07] Cale, I., “Gigabit Passive Optical Network - GPON”. International Conference on Information Technology Interfaces, Croatia, 2007.

[CFS] Inside the Linux 2.6 Completely Fair Scheduler, IBM, http://www.ibm.com/developerworks/linux/library/l-completely-fair-scheduler/.

[Cher07] Ludmila Cherkasova, Diwaker Gupta, Amin Vahdat, “Comparison of the three CPU schedulers in Xen”, *ACM SIGMETRICS*, 2007.

[CreditScheduler] Credit Scheduler, CPU scheduler for Xen. http://wiki.xen.org/wiki/Credit_Scheduler.

[Dahlen11] A. Dahlen, A. Johansson, F. Gunnarsson, J. Moe, T. Rimhagen, H. Kallin, "Evaluations of LTE Automatic Neighbor Relations," *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd* , vol., no., pp.1,5, 15-18 May 2011.

[Design06] RF Design, “Multigigabit wireless technology ot 70 GHz, 80 GHz and 90 GHz, May 2006.

Epitiro08 Epitiro Technologies, Ltd. “Femtocell Deployment Guide: An Operator-focused Strategy for a Successful Femtocell Rollout”, 2008.

[Forum05] Metro Ethernet Forum, “Ethernet Passive Optical Network (EPON) A Tutorial”, 2005.

[Fuden-Tir91] D. Fudenberg, J. Tirole, *Game Theory*, MIT Press, 1991.

[Gi09] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, “Calling the cloud: Enabling mobile phones as interfaces to cloud applications,” Middleware 2009, pp. 83–102, 2009.

[Fujiwara] Kayo Fujiwara - *Cost and accuracy of packet-level vs. analytical network simulations: an empirical study* - Master of Science in Computer Science - University of Hawaii - May 2007.

[Georgiadis06] L. Georgiadis, M. Neely, and L. Tassiulas, *Resource allocation and cross-layer control in wireless networks*, Foundations and Trends in Networking, pp. 1-144, 2006.

[Grove97] D. Grove, G. DeFouw, J. Dean, C. Chambers, “Call graph construction in object-oriented languages,” *SIGPLAN Notices*, vol. 32, no. 10, pp. 108-124, Oct. 1997.

[Grove01] D. Grove, C. Chambers, “A framework for call graph construction algorithms,” *ACM Transactions on Programming Languages and Systems*, vol. 23, no. 6, pp. 685-746, Nov. 2001.

[Hariyanto12] H. Hariyanto, R. Wulansari, A. Kurniawan, Hendarawan, “Femtocell Performance Over Non-SLA xDSL Access Network”, Book Chapter of Mobile Networks, In Tech Open Science, 2012.

[HaPi13] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan. 2013. Just-in-time provisioning for cyber foraging. In Proceeding of the 11th annual international conference on Mobile systems, applications, and services (MobiSys '13). ACM, New York, NY, USA, 153-166.

[Ikuno01] Josep Colom Ikuno, Martin Wrulich, Markus Rupp - *System level simulation of LTE networks.*

[Ikuno02] Josep Colom Ikuno, Martin Wrulich, Markus Rupp - *Performance and modeling of LTE H-ARQ.*

[ITU-TG.984.1] ITU-TG.984.1, Gigabit-capable Passive Optical Networks (GPON): General characteristics, Series G: Transmission Systems and Media, Digital Systems and Networks, 2003.

[Johnson01] D.B. Johnson, D.A. Maltz and J. Broch. “DSR: the dynamic source routing protocol for multihop wireless ad hoc networks.” In *Ad hoc networking*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA 139-172. 2001.

[Johnson07] S. G. Johnson and M. Frigo, “A modified split-radix FFT with fewer arithmetic operations”, *IEEE Transactions on Signal Processing*, vol. 55, pp. 111-119, 2007.

[Kamoun15] M. Kamoun, W. Labidi, and M. Sarkiss, “Joint Resource Allocation and Offloading Strategies in Cloud Enabled Cellular Networks,” submitted to IEEE International Conference on Communications (ICC) 2015.

[Khimsara10] S. Khimsara, K.K.R. Kambhatla, J. Hwang, S. Kumar, J.D. Matyjas, “AM-AOMDV: Adaptive Multi-metric Ad-Hoc On-Demand Multipath Distance Vector Routing”, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, 2010.

[Kincaid09] J. Kincaid, ``YouTube Mobile Uploads Up 400 since iPhone 3GS Launch," http://www.techcrunch.com/2009/06/25/youtube-mobile-uploads-up-400-since-iphone-3gs-launch/, 2009.

[Kumar13] S. Kumar, S. Khimsara, K. Kambhatla, K. Girivanesh, J.D. Matyjas, and M. Medley, “Robust On-Demand Multipath Routing with Dynamic Path Upgrade for Delay-Sensitive Data over Ad Hoc Networks,” *Journal of Computer Networks and Communications*, vol. 2013.

[Kushner97] H. J. Kushner and G. G. Yin, “Stochastic Approximation Algorithms and Applications”. *New York: Springer-Verlag*, 1997.

[Labidi15] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-Optimal Resource Scheduling and Computation Offloading in Small Cell Networks", submitted to IEEE International Conference on Telecommunications (ICT) 2015.

[Lewin95] R. A. Powers, ``Batteries for low power electronics," *Proceedings of the IEEE*, vol.83, pp. 687-693, April 1995.

[Lewin08] J. Lewin, ``iPhone Users 30 Times More Likely To Watch YouTube Videos," http://www.podcastingnews.com/2008/03/19/iphone-users-30-times-watch-youtube-videos/, 2008.

[Lin11] Z. Lin, G. Wood, "TMS320TCI6618 - TI's high-performance LTE physical layer solution", Texas Instruments white paper, 2011.

[Lop07] V. Cardellini, E. Casalicchio, V. Grassi, F. Lo Presti, "Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes", in the Proc. of the IEEE 2007 International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA.

[Magen08] Dan Magenheimer,"Memory Overcommit without the Commitment",*Xen Summit*, 2008.

[Maui10] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," *Proc. of the ACM International Conference on Mobile Systems, Applications, and Services*, pp.49-62, San Francisco, CA, USA, 15-18 June 2010.

[Marina06] M. K. Marina and S. R. Das, "Ad hoc on-demand multipath distance vector routing," Wireless Communications and Mobile Computing, vol. 6, no. 7, pp. 969–988, 2006.

[Mezzavilla] Marco Mezzavilla, Marco Miozzo, Michele Rossi, Nicola Baldo, Michele Zorzi - *A lightweight and accurate Link Abstraction Model for LTE networks in ns-3.*

[Miettinen10] A.P. Miettinen, J.K. Nurminen, "Energy Efficiency of Mobile Clients in Cloud Computing", in *Proc. 2nd USENIX Conference on Hot Topics in Clod Computing 2010* (HotClout'10), Boston (USA), June 2010.

[Motwani96] R. Motwani, Raghavan, "Randomized Algorithms", *ACM Computing Surveys*, pp. 33 - 37, 1996.

[Munoz13a] O. Muñoz, A. Pascual Iserte, J. Vidal, "Joint Allocation of Radio and Computational Resources in Wireless Application Offloading". Proceedings FUNEMS 2013 (Future Network & Mobile Summit), pp. 1-10, ISBN 978-1-905824-37-3. Lisbon (Portugal), July 2013.

[Munoz13b] O. Muñoz, A. Pascual Iserte, J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading". Accepted to be published at *IEEE Trans. on Vehicular Technology*, November 2014. Available online at http://arxiv.org/abs/1405.4487

[Munoz14a] Olga Muñoz, Antonio Pascual Iserte, Josep Vidal, Marc Molina, "Energy-Latency Trade-off for Multiuser Wireless Computation Offloading". Proceedings WCNC 2014 (IEEE Wireless Communications and Networking Conference), workshop CLEEN (Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks), pp. 29-33. Istanbul (Turkey), April 2014.

[Munoz14b] Marc Molina, Olga Muñoz, Antonio Pascual-Iserte, Josep Vidal, "Joint Scheduling of Communication and Computation Resources in Multiuser Wireless Application Offloading". Proceedings PIMRC 2014 (IEEE International Symposium on Persona, Indoor, and Mobile Radio Communications). Washington, September 2014.

[Naylon13] Naylon, J., "Breaking Barriers for small cell backhaul". Cambridge Broadband Networks, 2013.

[Networks12] Cambridge Broadband Networks, “White Paper: Easy Small Cell Backhaul, an analysis of small cell backhaul requirements and comparison of solutions”. Global market leaders in multipoint microwave access and backhaul solutions, 2012.

[Nemh-Wolsey88] G. L. Nemhauser, L. A. Wolsey, *Integer and combinatorial optimization*, Wiley, 1988.

[NFV12] “Network Functions Virtualisation. An Introduction, Benefits, Enablers, Challenges & Call for Action”, SDN and OpenFlow World Congress, October 2012.

[Noon11] A. Noon, A. Kalakech, S. Kadry, “A New Round Robin Based Scheduling Algorithm for Operation Systems: Dynamic Quantum Using the Mean Average”, *Internation Journal of Computer Science Issues (IJCSI)*, vol. 8, issue 3, no. 1, May 2011.

[ns3] "ns-3 Model Library - Release ns-3.18" - August 30, 2013, available at http://www.nsnam.org/docs/models/html/lte.html

[ONF12] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks", White paper, April 2012.

[OpenCV] http://opencv.org/.

[Palacin09] M. R. Palacin, ``Recent advances in rechargeable battery materials: a chemists perspective," *Chemical Society Reviews*, no. 38, pp. 2565-2575, 2009.

[Palomar07] D. P. Palomar, and M. Chiang, “Alternative distributed algorithms for network utility maximization: Framework and applications,’” IEEE Trans. on Automatic Control, vol. 52, no. 12, pp. 2254-2269, December 2007.

[PDL-Barb13] P. Di Lorenzo, S. Barbarossa, and S. Sardellitti, “Joint Optimization of Radio Resources and Code Partitioning in Mobile Cloud Computing,” submitted to IEEE Transactions on Wireless Communications, July 2013, available on Tropic website.

[Perkins99] C.E. Perkins, E.M. Royer, "Ad-hoc on-demand distance vector routing," *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on* , vol., no., pp.90,100, 25-26 Feb 1999.

[Plachy14] J. Plachy, Z. Becvar, P. Mach, " Handover for efficient delivery of data between mobile users and cloud-enhanced small cells," submitted to Computer Networks, 2014.

[Rahmawan09] H. Rahmawan, Y. S. Gondokaryono, “The Simulation of Static Load Balancing Algorithms”, *in the Proc. of the 2007 International Conference on Electrical Engineering and Informatics*, pp. 640-645, Aug. 2009.

[Red13] Red Hat Enterprise Linux Virtualization Guide, edition 5.9, 2013.

[Ryder79] B. G. Ryder, “Constructing the Call Graph of a Program,” *IEEE Transactions on Software Engineering*, vol. SE-5, no. 3, pp. 216-226, May 1979.

[Sa09] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, “The case for VM-based cloudlets in mobile computing”, IEEE Pervasive Computing 8 (2009) 14–23.

[SardBarb14] S. Sardellitti, G. Scutari, S. Barbarossa, “Joint optimization of radio and computational resources in multicell mobile cloud computing,” IEEE Trans. on Signal Process., to be submitted, 2014.

[SardScutBarb] S. Sardellitti, G. Scutari, S. Barbarossa, "Joint optimization of radio and computational resources for mobile cloud computing," in Proc. of SPAWC, Toronto, June 2014.

[SardScutBarb14] S. Sardellitti, G. Scutari, S. Barbarossa, "Distributed joint optimization of radio and computational resources for mobile cloud computing," in Proc. of IEEE Int. Conf. on Cloud Networking (Cloudnet), Luxembourg, October 2014.

[SardBarb2014] S. Sardellitti, S. Barbarossa, G. Scutari, "Distributed Mobile Cloud Computing: Joint Optimization of Radio and Computational Resources", Globecom 2014 Workshop- WONC, Austin, Tx USA, December 2014.

[Salodkar08] N. Salodkar, A. Bhorkar, A. Karandikar, and V.S. Borkar, "An On- Line Learning Algorithm for Energy Efficient Delay Constrained Scheduling over a Fading Channel," *IEEE J. Selected Areas in Comm., special issue on control and communications*, vol. 26, no. 4, pp. 732-742, May 2008.

[Sharma08] S. Sharma, S. Singh, M. Sharma, "Performance Analysis of Load Balancing Algorithms," *in the Proc. of World Academy of Science, Engineering, and Technology*, pp. 269-272, 2008.

[Yang05] Y. Yang, J. Wang and R. Kravets, "Designing Routing Metrics for Mesh Networks", *Proceedings of the IEEE Workshop on Wireless Mesh Networks (WiMesh). IEEE Press* 2005.

[Sandvine13] Sandvine, "Global Internet Phenomena Report 1H2013", 2013.

[Saunders09] S. Saunders et all, "Femtocells: Opportunities and Challenges for Business and Technology", John Wiley & Sons, 2009.

[Sesia09] S. Sesia, I. Toufik, and M. Baker, *LTE-The UMTS Long Term Evolution: From Theory to Practice*, John Wiley & Sons, 2009.

[Subandrio10] Subandrio, A., "Pemanfaatan GPON sebagai solusi backhaul jaringan CDMA dan GSM". PT. Telekomunikasi Indonesia Tbk., 2010.

[ScutariFacch14] G. Scutari, F. Facchinei, L. Lampariello, and P. Song, "Parallel and distributed methods for nonconvex optimization," in Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 14), May 4-9, 2014, Florence, Italy.

[ScutFacchLamp] G. Scutari, F. Facchinei, L. Lampariello, and P. Song, "Parallel and distributed methods for nonconvex optimization-Part I & II: Theory & Applications," IEEE Trans. on Signal Processing, (under review).

[ScutariFacchinei] G. Scutari, F. Facchinei, P. Song, D.P. Palomar, and J.-S. Pang, "Decomposition by partial linearization: Parallel optimization of multi-agent systems," IEEE Trans. on Signal Process., Vol. 63, no. 3, pp. 641-656, Feb. 2014.

[Sutton98] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An introduction". *Cambridge, MA: MIT Press*, 1998.

[Takemura09] Chris Takemura, Luke S. Crawford, "The Book of Xen", *No Starch Press*, 2009.

[Tang05] S. Tang, non-member, B. Zhang, M. Watanabe, S. Tanaka. "A Link Heterogeneity-Aware On-Demand Routing (LHAOR) Protocol Utilizing Local Update and RSSI Information" *In IEICE Transactions*, (88-B) 9: 3588-3597, 2005.

[Tri94] K.S. Trivedi, M. Malhotra, R.M. Fricks, "Markov Reward Approach to Performability and Reliability Analysis", *Proc. of Mascots 94*, Feb. 1994.

[Trojer08] Trojer, E., Dahlfort, S., Hood, D. Mickelsson, H., "Current and next generation PON: A technical overview of present and future PON technology". Ericsson Review, 2008.

[TROPIC_D21] Deliverable D21, "Scenarios and requirements", TROPIC project deliverable, issue b, www.ict-tropic.eu.

[TROPIC_D22] Deliverable D22, "Design of network architecture for femto-cloud computing", TROPIC project deliverable, issue b, www.ict-tropic.eu.

[TROPIC_D311] Deliverable D311, "Building blocks for MP2MP energy-constrained communication systems", TROPIC project deliverable, www.ict-tropic.eu.

[TROPIC_D53] Deliverable D53, "Security mechanisms for femto-cloud systems", TROPIC project deliverable, www.ict-tropic.eu.

[XenDMC] Xen Dynamic Memory Control, http://wiki.xen.org/wiki/XCP_FAQ_Dynamic_Memory_Control.

[Wald02] C. A. Waldspurger. "Memory Resource Management in VMware ESX Server". In Proc. of the 5th USENIX Symp. on Operating System Design and Implementation, December 2002.

[Wigard09] J. Wigard, T. Kolding, L. Dalsgaard, and C. Coletti, "On the User Performance of LTE UE Power Savings Schemes with Discontinuous Reception in LTE," in Proc. IEEE International Conference on Communications Workshops, ICC Workshops 2009, June 2009.

List of abbreviations & symbols

ADSL	Asymmetric Digital Subscriber Line
AM-AOMDV	Ad-Hoc On-Demand Multipath Distance Vector
ANR	Automatic Neighbour Relation
ANS	Answer
AODV	Ad-Hoc On-Demand Distance Vector
AOMDV	Ad-Hoc On-Demand Multipath Distance Vector
AOMDV-DPU	Ad-hoc on-demand distance vector with dynamic path Update
AWGN	Additive White Gaussian Noise
BVT	Borrowed Virtual
CB	Code Block
CFS	Completely Fair Scheduler
CMDP	Constrained Markov Decision Problem
CPU	Central Processing Unit
DL	DownLink
DMC	Dynamic Memory Control
DomU	Domain Unprivileged
DSL	Digital Subscriber Line
DSR	Dynamic Source Routing
EESM	Exponential Effective SINR Mapping
eNB	enhanced Node B
ESM	Effective SINR Mapping
H-SCM	Hierarchical Small cell Cloud Manager
HDD	Hard Disk Drive
HeNB	Home enhanced Node B
L2S	Link-to-system
LDSC	Large-Scale Distributed Computing
LHAOR	Link Heterogeneity-Aware On-Demand Routing
LTE	Long Term Evolution
MANET	Mobile Ad-Hoc Network
MIESM	Mutual Information Effective SINR Mapping
MME	Mobility Management Entity
MMIB	Mean Mutual Information per coded Bit
NCL	Neighbour Cell List
NEI ID	Neighbor ID
OS	Operating System
P-GW	Packet Gateway
PDS	Post Decision State
PER	Packet Error Rate
PSwH	Path Selection with Handover
PV	Para-virtualization / Para-virtualized
QoS	Quality Of Service
RAM	Random Access Memory
REQ	Request
RSSI	Received Signal Strength Indication
RVI	Relative Value Iteration
S-GW	Serving Gateway
SCC-GW	Small Cell Cloud Gateway
SCA	Successive Convex Approximation
SCC	Small Cell Cloud
SCeNB	Small Cell enhanced Node B
SCeNBce	Small Cell enhanced Node B cloud enabled
SCM	Small Cell Cloud manager

SCMP	Small cell Cloud Management Procedure
SEDF	Time Simple Earliest Deadline First
SINR	Signal Interference Ratio
SMP	Symmetric multiprocessing
SO	Serving cell Only
TB	Transport Block
TS	Time Stamp
UE	User Equipment
UL	Uplink
VCPU	Virtual CPU
VH-SCM	Virtual-Hierarchical Small cell Cloud Manager
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
WSN	Wireless Sensor Networks

1 INTRODUCTION

The small cell cloud is a combination of mobile network and cloud computing providing proximity access to computing resources to users. In common mobile networks, cloud computing is not considered and transmission of data offloaded from user equipments (UE) is handled as common data transmission without awareness of computing resources. Both radio and cloud parts of the SCC have been addressed in WP3 and WP4, respectively. However, separated management of radio and computation might lead to inefficient utilization of resources and draining of UE's battery. Therefore, this document focuses on integration of both radio and computation into a single framework managing computation with respect to status of all elements in the network. To enable joint management of radio and computing resources, we exploit preliminary architecture with small cell cloud manager (SCM) proposed in WP2 and elaborated in WP4. However, joint management of radio and cloud requires additional enhancements in order to enable efficient management of on offloading and exchange of all required system information. Therefore, this document refines the proposed architecture and provides finalized design of the SCC system including definition of individual components and their functionalities, description of interfaces and SCC architecture with related protocols for the system management and control.

To maximize energy efficiency and minimize delay, it is necessary to make proper decision on whether to offload application to the SCC (maybe partially) or compute it locally at the UE. To that end, offloading decision algorithm considering all known information about UE's capabilities and status of network (including radio, backhaul and computing characteristics) are designed. As the offloading implies additional traffic, allocation of resources is adapted to handle the increased traffic demands. Being offloading decision dependent on both the radio and computational aspects, in this deliverable, we propose a cross-layer offloading optimization strategy, which encompasses application, control and management and physical layer into a joint framework where the optimal radio and computational resources are found out in order to minimize the energy consumption at the mobile sides under transmit power and latency constraints. We first formulate the problem in a single cell scenario and then we extend this optimization strategy in a dense deployment scenario where offloading becomes much more complicated because of intercell interference. In this context the designed strategy provides an optimal way to jointly manage radio and computational resources required to perform computation offloading by guaranteeing the prescribed user QoS.

The SCC has been designed as the system mainly exploited by static or nomadic users attached to the same base station during computation. However, during the project life-time, it comes out that interesting scenarios might covers also fully mobile users. Hence, we extend the scope of this deliverable to cover also issues related to mobility management for users on move to avoid drop of connection or loss of computed data. We propose algorithm selecting the most suitable path (by means of serving cell) in order to minimize time required to deliver data from the UE to all computing cells and back. Also, possibility of virtual machines migration is analyzed. In case of computation offloading, handover comes to depend on both radio and computing resources where virtual machines are running. Therefore, in a multi cloud scenario, we provide a new way to handle handover by jointly optimizing the radio and computing resources according to the channel condition, application parameters and backhaul state. We jointly find out the optimal association of mobile users to base stations and clouds which provides a mechanism for optimal instantiation of virtual machines.

The rest of the document is organized as follows. The next section gives an overview on considered scenarios and system assumptions taken into account in this document. Section 3 provides description of model of backhaul and abstraction models of PHY layer, application, and small cell. The backhaul modeling is obtained by means of real network measurements. Contrary, the abstraction models are derived as a selection among existing approaches in literature. Final design of the SCC system, including design of interfaces between radio and cloud parts, modules, protocols and system integration are addressed in Section 4. The application offloading and radio resource allocation for transmission of the offloaded data and computation results are addressed in Section 5. In Section 6, we investigate impact of mobility on of small cell cloud services and we define approaches how to handle

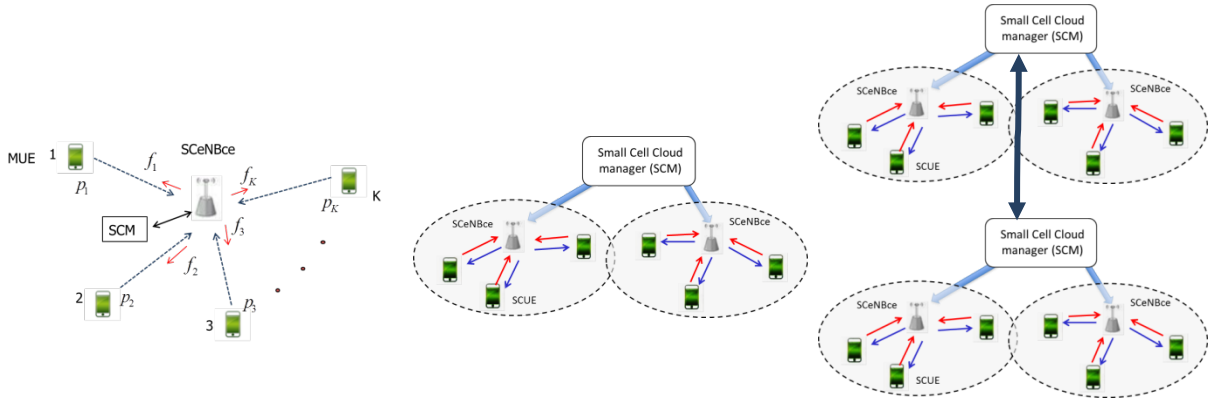
mobility of user exploiting cloud services. Then, Section 7 provides overview on the recommendations for stakeholders and recommendations towards WP6. The last section summarizes the major conclusions.

2 SCENARIOS, ASSUMPTIONS, AND ARCHITECTURE

In general, scenarios, assumptions and architecture defined in [TROPIC_D21] are used as a base line for this document. Nevertheless, as the D21 was issued in the first phase of the project, some assumptions, models and scenarios are slightly updated and aligned with the needs of the SCC. This section gives high level overview on major aspects considered in this milestone. It also summarizes major and general parameter considered for evaluations.

For modeling of the small cells' backhaul, all three business scenarios introduced in [TROPIC_D21] will be addressed and models for them will be carried out in D51.

In general, three basic technical scenarios can be considered, as shown in Figure 1: a) a single SCM associated to a single SceNB, serving a set of mobile users (Figure 1a); b) a single SCM serving multiple SceNB's, each of them serving a set of mobile users (Figure 1b); c) multiple SCM's serving multiple SCeNBces (Figure 1c).



a) **b)** **c)**

Figure 1. Technical scenarios considered for offloading aspects.

These three scenarios are envisaged to represent different level of complexity and, at the same time, opportunities. Given the difficulty of the overall problem, it is advisable to proceed from the simpler case (Figure 1a) to the most challenging one (Figure 1c). The different levels of challenges/opportunities pertaining to the three scenarios are listed below:

1) The single SCM scenario of (Figure 1a) is useful to find out the optimal resource allocation, including both radio and computing resources, for a set of users served by a single pair of base station/cloud. In such a case, there is no interference because all radio resources are allocated by a single entity;

2) In the multiple SceNB/single SCM case of Figure 1b), radio resources are assigned from multiple base stations. In such a case, there could be intercell interference among mobile users. The computational resources are still assigned by a single entity. The challenge is to analyze the effect of intercell interference on the overall resource allocation;

3) In the most general scenario reported in Figure 1c), the computations may be performed in multiple cloud servers, linked through high speed channels. This scenario is useful to study resource allocations across multiple servers, taking into account the backhaul between the servers. It is also useful to analyze hand-over mechanisms, for low mobility users, where handover now incorporates the association not only between a mobile user and one or more base stations, as in conventional handover, but between the mobile user and one or more clouds, through one or more base stations. In such a case, the challenge is to evaluate how the computation offloading strategies are affected by the wireless traffic, interference,

computational load across multiple clouds and the quality of the backhaul linking base stations and clouds.

Any specific parameters are listed in individual sections with technical content.

Assumption and requirements considered in this document and also by the TROPIC project are summarized in Table 42 in Annex I.

Regarding the network architecture, we consider centralized approaches as recommended in [TROPIC_D21] and further analyzed in WP4. From the system level point of view, all three centralized approaches (SCM in HeNB gateway, SCM in MME, standalone SCM) are equivalent as the specific deployment of the SCM influences signaling delay but not data, which does not pass the SCM. Beyond the planned activities related to architecture, we also investigate possibility to enhance the centralized control towards advanced architectures and we design high-level management procedures suitable for it.

3 SCC SYSTEM MODELING

One of the main questions to be solved within TROPIC is how to distribute the computation and communication capabilities between the mobile handsets and the SCC (small cells cloud) in an efficient way.

For an energy-limited mobile terminal that wants to launch an application which is highly computationally demanding, different options may be considered, as for instance:

- running the application totally at the mobile terminal,
- running the application totally at the SCC, and
- running the application partially at the mobile terminal and partially at the SCC provided that the application can be parallelized and/or the data to be processed can be partitioned.

Given a set of possible options, we would like to take the best decision according to a certain criterion. As a criterion it could be considered for instance the energy spent by the mobile terminal or the total execution time, as it will be explained in Section 5. Note that despite of the mobile terminal is expected to have less processing capabilities than those available in the SCC, offloading an application requires additional energy and time to send the input and output data between the mobile terminal and the serving small-cell. As a result, it is not clear which the best decision is, and surely it will depend on the channel conditions and the communication capabilities of the terminals. The goal of WP5 is to provide optimization methods that will find the best solution considering the time and energy consumption at both communication and computation segments. In order to formulate mathematically the optimization problem, the first thing needed is to have appropriate models for the communication techniques in the physical layer, application models, and a model of the SCeNBce.

3.1 *Network model*

3.1.1 Network models overview

A communication network can be simulated by several models with different accuracy and hence a different complexity and computation load. The selection of the simulation model is a trade-off between simulation accuracy (i.e. how close is the simulation to the real-world) and simulation cost (i.e. how much time and resources are needed to run the simulation); the decision process, therefore, is based on two main factors: the purpose of the simulation and the dimension of the network.

The simulation of a network protocol needs a model with good accuracy but it is usually limited to a small number of nodes, while the simulation of distributed application/computation (often referred as Large-Scale Distributed Computing - LSDC) has to encompass a large number of nodes and it leverages on a simpler model.

The main network models proposed in the literature and used in the most popular simulators are:

- packet-level model
- fluid-flow model (or flow-level)
- delay-based model

Packet-level model

The packet-level simulation achieves high accuracy as it simulates every individual packet going through the network links and the routers. This kind of simulation uses discrete event simulation techniques. A flow over network links can be represented as a sequence of events, each of which occurs at a certain discrete time: the departure of a packet and its arrival are treated as two separate events. The simulator models such events and keeps a list of events to be processed. The simulator assigns a time to each event, and when the simulation clock reaches this time, the simulator executes the event and changes the statuses of other events if necessary.

Since each packet is simulated, the accuracy of the simulation is high and it provides in depth information about the network elements, but this involve a high number of events also for simple communications. Considering that the simulation time increases roughly in proportion to the number of events, such simulations are generally rather slow and not scalable.

Packet-level models capture the role of buffers as well as queuing and propagation delays. They are useful to estimate quantities such as throughput and they have been heavily used to study the stability of congestion control algorithms in TCP networks.

These very fine models are not always meaningful at a very large scale and their use in the context of distributed computing raises several difficult issues:

- such complex models are much harder to instantiate than simple ones (i.e. detailed topology, network characteristics of the whole platform);
- they require an instantiation with parameters that are often not available (such as a complete description of the Internet); at such scale, these models may be unstable and, badly instantiated, are worse than simple models;
- packet-level simulation is so heavy that it becomes completely unusable beyond a few thousands hosts (due to both the memory capacity and the process time).

Fluid-flow model

The fluid model makes an abstraction of individual packets. For the case of packets based simulation, the simulator has to treat all the packets events in the network, while in the fluid simulation, it is only necessary to treat the rate change events in different network nodes.
The abstraction takes place when, packet flows with little time slots separations are considered in the same fluid flow with a constant fluid rate. Little time variations among packets are not considered and in this way, the number of events is reduced.
In a packet event simulator, the number of events is proportional to the number of packets produced by the traffic source, while in a fluid simulator; the number of events is proportional to the number of rate transitions.

It is assumed that there is an underlying mechanism that effects ‘statistical bandwidth sharing’, i.e., the rate of each flow is adjusted instantaneously based on the available bandwidth on its path and the number of flows that it shares the bandwidth with at that instant. At each instant of time, a flow transmits data through the network according to the rate allotted to it.

The time needed to transfer a message of size $\boldsymbol{S}$ between hosts $\boldsymbol{i}$ and $\boldsymbol{j}$ is then given by:

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j} \quad (1)$$

where $\boldsymbol{L_{i,j}}$ (resp. $\boldsymbol{B_{i,j}}$) is the network latency (resp. bandwidth) on the route connecting i and j. Although determining $\boldsymbol{L_{i,j}}$ may be easy, estimating the bandwidth $\boldsymbol{B_{i,j}}$ is more difficult as it requires to account for interactions with every other flow. Then the simulation amounts to solving a bandwidth sharing problem, i.e., determining how much bandwidth is allocated to each flow.
Given the computed bandwidth allocation, which defines all data transfer rates, and the size of the data to be transmitted by each flow, one can determine which flow will complete first. Upon completion of a flow, or upon arrival of a new flow, the bandwidth allocation can be reevaluated. Usually, this reevaluation is memory-less and does not depend on past bandwidth allocations. This approach makes it possible to quickly step forward through (simulated) time, and thus is attractive for implementing scalable simulations of LDSC systems with potentially large amounts of communicated data.

Delay-based model
In many simulations the representation of the communications on the network simply through a constant delay or a statistical distribution may be sufficient. Several models determine delays based on coordinate-based systems.

P2P simulators generally rely on delay-based models and forget about the underlying physical topology; using this approach they can simulate platforms that scale up to millions of nodes. However, these very scalable models ignore network congestion and typically assume an extremely large bandwidth. The following table shows the model adopted by the most popular network simulators.

	Packet-level model	Fluid-flow model	Delay model
OMNET++	★(using INET)		
ns-3 (ns-2)	★		
CloudSim			★
Green Cloud	★		
OPNET	★		
SimGrid	★ (using ns-3)	★	
GTNetS	★		
SSFNet	★		
FluidSim		★	
OverSim	★(using INET)		★(coordinate-based)

Table 1. Models adopted in network simulators.

Comparison of packet-level and flow-level models
The accuracy of a network model is not the sole aspect considered in the selection of a simulator but it has to be always placed in relation with the computational burden.

The packet-level simulators have the issue that simulation time can be orders of magnitude larger than simulated time for simulations that involve realistic topologies with a great number of flows.

To give an idea of the computational complexity of a packet-level simulator with respect to a flow-level simulator, the following table shows the result of a study ([Fujiwara]) based on GTNetS as packet-level simulator and SimGrid as flow-level simulator.

Number of flow	Packet-level			Flow-level		
	Simulated (s)	Simulation (s)	Ratio	Simulated (s)	Simulation (s)	Ratio
10	70.31	65.34	0.92	72.72	0.00176	0.00002
25	87.77	163.10	1.86	80.67	0.00836	0.00010
50	93.73	364.67	3.89	98.44	0.02839	0.00029
100	93.19	753.42	8.08	97.21	0.13810	0.00142
200	124.10	1562.90	12.59	134.06	0.53833	0.00402

Table 2. Simulated time and simulation time in packet and flow level models.

The ratio is defined as simulation time divided by simulated time, and thus, a high ratio means a high simulation cost. The time to transfer 100 MB data is measured for a randomly generated topology with 200 nodes and various numbers of flows.

3.1.2 Network attributes description

A communication network may be modeled using different sets of attribute, depending on the objective of the simulation. The following metrics have been chosen for the simulations in the TROPIC project.

Latency (delay)
Latency refers to the amount of time (usually measured in milliseconds) it takes for data to travel from one location to another across a network. It's often referred to as *delay*, because a software is waiting to execute some function while data travels back and forth across the network.

The delay is the sum of delays on each subnet link traversed by the packet. Each link delay, in turn, consists of the following components:

- processing delay: is the difference between the time the packet is correctly received at the head node of the link and the time the packet is assigned to an outgoing link queue for transmission (due to error control, routing decisions etc.);
- queueing delay: is the difference between the time the packet is assigned to a queue for transmission and the time it starts being transmitted. During this time, the packet waits while other packets in the transmission queue are transmitted. The queues occur when incoming traffic to a node is larger than the forwarding capacity (for some period of time);
- transmission delay: (also called the *store-and-forward delay*) is the difference between the times that the first and last bits of the packet are transmitted;
- propagation delay: is the difference between the time the last bit is transmitted at the head node of the link and the time the last bit is received at the tail node. This is proportional to the physical distance between transmitter and receiver; it can be relatively substantial, particularly for a satellite link or a very high speed link.

The propagation delay depends on the physical characteristics of the link and is independent of the traffic carried by the link. The processing delay is also independent of the amount of traffic handled by the corresponding node if computation power is not a limiting resource.

The above components of the delay are not always characterized individually, but often it is used the end-to-end delay of a network link, from sender to (final) receiver.

Jitter
The *delay variation* or *jitter* expresses the variation of the latency over time. It's often described by the maximum difference in delay, the mean difference in delay and the standard deviation.

Packet loss rate
It is defined as the percentage of packets lost, due to congestion (buffer overflow) and bit errors. The latter has greater effects in the wireless networks.

The way in which the packets are lost in a network (how and when it happens) depends on the queuing management algorithm used by the queues in the network.

In the *DropTail* algorithm, packets are only discarded when the buffer is completely full. On the contrary, for *RED*, with its goal of achieving a high network utilization and avoid source synchronization, packets are discarded randomly even before the buffer is completely full. This preventive packet drop probability used by RED is calculated as a function of an average mean queue length.

Two extreme assumptions about packet loss are common in the TCP modeling literature. The first approach considers packets to be independent, where RED is usually referred to for support of this assumption. The second approach assumes that packet loss, after the first lost packet, is strongly

correlated so that once a packet is lost every packet is dropped for a short period of time. The support for this strong correlation structure is a DropTail queue.

Packet Error Rate
The packet error rate (PER) is the number of incorrectly received data packets divided by the total number of received packets. A packet is declared incorrect if at least one bit is erroneous.

The probability of packet error depends on the probability of bit error, the packet length and, for wireless networks, the channel coherence time. Therefore, the PER is also important parameter indicating the performance of the system with a given packet length.

Queue length
The queues length in the communication networks is a function of several factors, as the the traffic on the network, both intensity and type (i.e. periodic or bursty), the capacity of the link and the internal buffer dimension. Its statistical distribution provides information about the delay and the packet loss.

Link capacity (bandwith)
It expresses the bits per second that can be sent through the network (usually measured in kbps, Mbps or Gbps).

The terms *bandwidth* and *throughput* are sometimes used interchangeably, and though they are related, they're not quite the same. They both refer to the amount of data transferred between two points on a network in a given period of time, but bandwidth generally refers to a theoretical maximum, while throughput is a real-world, practical measurement.

Availability
Availability, sometimes referred to as *uptime* or *responsiveness*, refers to the amount of time that a computer or a network connection is functioning and usable.

3.1.3 **Modeling of Small Cells Backhaul**

Definition of small cell was rather difficult to conclude, especially when we need to agree with industry requirement. Based on current definitions about small cell deployment, there are some criteria for cells to be considered as Smallcell as listed below [Alliance12]:

- Provide the coverage of an area smaller than a macro cell (so that one macro cell overlaps several small cells in the same area)
- As macro cells, they are deployed and managed by operators
- Grant an open access to all users (of the same operator)
- Characterized by a lower equipment and installation cost if compared to macro cells
- Oriented to the support of data services, although voice services can also be supported

And besides those criteria there are some technical parameters that we must be considered as important at small cell deployment:

- Capacity (average, peak)
- Services supported (best effort data, real time data, voice)
- Mobility support (support of handover between macro and small cells or among small cells, support of the X2 interface, S1/X2 traffic ratio)
- Service requirements in terms of QoS (latency, jitter, packet loss, availability) and time and frequency synchronization requirements
- Deployment requirements
    - Power consumption (related to backhauling)
    - Operational conditions (public access, operator's deployed backhaul)
    - Potential location (indoor/outdoor, a few meters above street level or rooftop).

From above parameters, the following illustration Figure 2 shows small cell backhaul infrastructure, including the nodes used here.

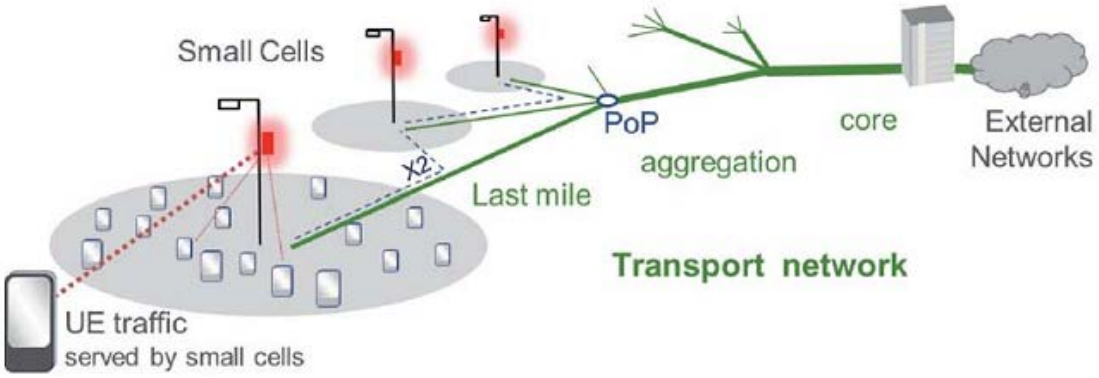


Figure 2. Abstract view of small cell backhaul [Alliance12].

As seen in figure above, backhaul provides connection between small cell and core network and also to provide ability to manage desire QoS level for the service offered. Background from development of backhaul is to provide connectivity between the small cells and the core network nodes with desired QoS level.

For operators not having an existing macro cell layer (also known as Small Cell Greenfield operators) the later described deployment scenarios will be similar. The difference will be however in the usage of fixed networks (based on operator's own or on leased lines) for the hand-off from the dedicated small cell backhaul network to the existing infrastructure.

The new challenge brought by small cells is in providing connectivity from the hard-to-reach locations below rooftop to a site being part of the existing transport infrastructure.

3.1.4 Access and Transport Network Characteristics

3.1.4.1 *Component of Backhaul Traffic*

Backhaul traffic consist of three major types of traffics: background traffic, specific application traffic and overhead traffic management. Specific application traffic is generated by an end system service application.

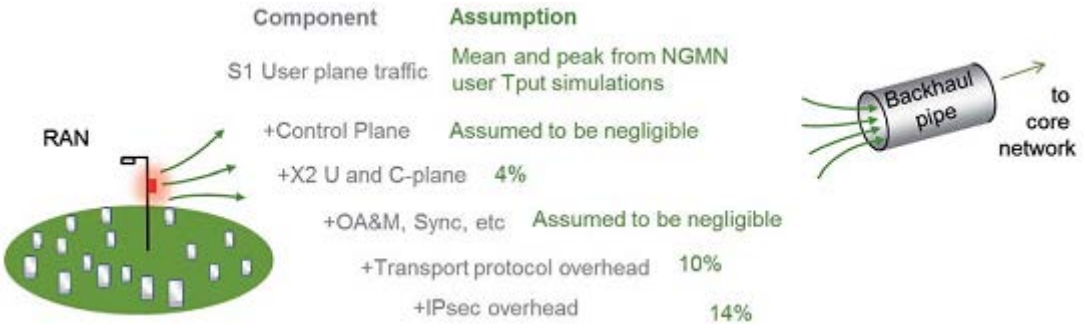


Figure 3. Componenets of LTE backhaul traffic and overhead assumptions [Alliance12].

Various traffic components generated at each smallcell which have to be backhauled, as shown in Figure 3. Most of the traffic is the user plane data. Other components can be considered as overheads and are expressed as a proportion of the user plane traffic. It should be noted that the X2 is not

supported for release 8 LTE Home eNodeBs, however in the absence of X2, the S1 handover would be performed.

In practice, the level of overhead may vary with the traffic type. For example, a large number of low data rate connections (e.g. Machine to Machine) will have a higher proportion of signaling than a small number of high data rate users (e.g. file sharing). Figure 3 represents a mix of different traffic types, and is consistent with NGMN's previous LTE provisioning guidelines. For HSPA small cells, we assume the Iu-h overheads to be similar to the Iub.

3.1.4.2 *QoS Support*

Quality of Service refers to the performance of the connectivity for all user, management and control plane traffic, which may include (but not limited to) the following:

- Data rate
- Packet Delay (latency)
- Delay Variation (jitter)
- Packet Loss
- Connection setup time
- Connection availability
- Connection drop time
- Connection interruption time (e.g. during handover)

QoS for most of application plays important roles to user experience. Different application needs different QoS assurance. For backhaul segment typically provide different QoS class that mapped into small number of CoS (Classes of Service).

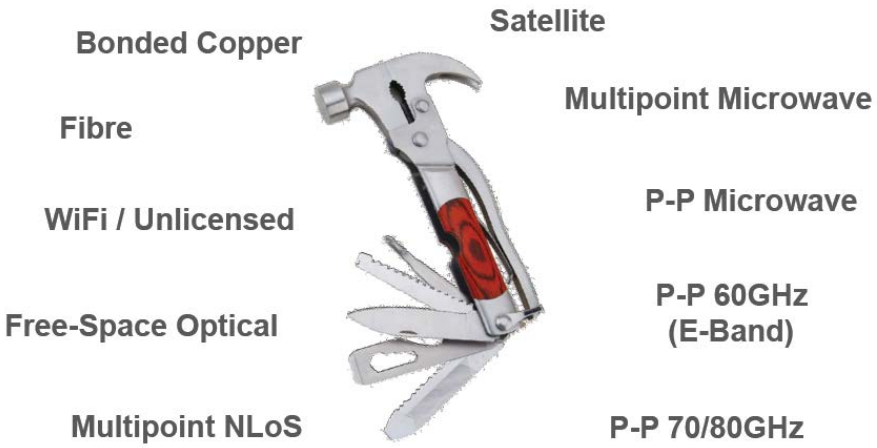


Figure 4. Small cell backhaul technology [Naylon13].

3.1.4.3 *Types of Small Cell Backhaul*

Small Cell Backhaul solutions are now emerging and there are many types of Small Cell Backhaul technology that were proposed. Here some technology types of small cell backhaul:

Fibre: Fibre technology provides low-latency connection and a very high-capacity. Fibre will be needed for connecting at PoPs, and will be present at an increasing of indoor locations.

DSL: DSL technology provides a basic level of connectivity. DSL data rates should satisfy the provisioning requirements for the 'loaded' HSPA and potentially 10MHz LTE, but it will not provide the peaks needed for a good end user quality of experience.

Non-line of sight (NLoS) wireless: Good for coverage, but capacity limited by available spectrum. NLoS wireless backhaul would be the perfect solution were it not for the small cells and Wi-Fi hotspots already using the entire low frequency spectrum available. NLoS propagation requires low carrier frequencies of less than a few GHz which are highly prized for mobile access itself, as shown in Figure 5. The height of the bar indicates the MHz of bandwidth for uplink and downlink traffic. As a general rule, the bandwidth available for backhaul needs to be at least as much as that for access. Some claim that the spectral efficiency of the backhaul will be higher to compensate, but this seems unlikely given that access and backhaul are operating in very similar (NLoS) propagation conditions and with interference from nearby co-channel transmitters.

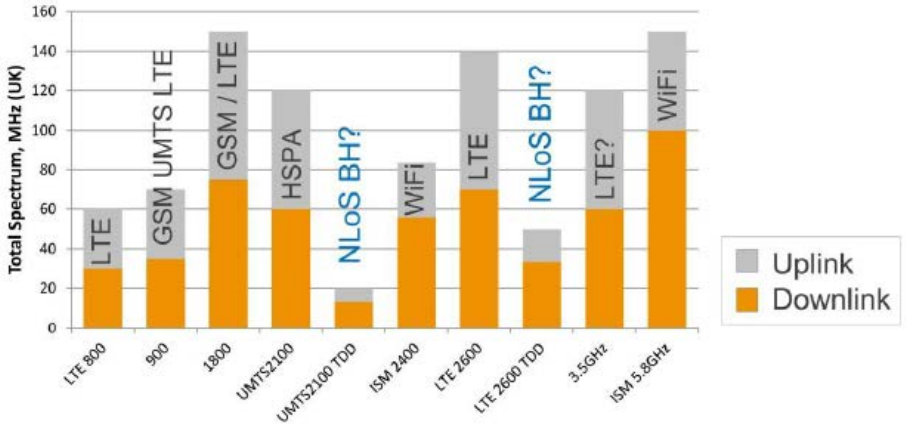


Figure 5. Comparison of small backhaul technology [Networks12].

Unpaired 'TDD' spectrum could potentially be used for NLoS backhaul, but as the graph shows, the quantity of this is small compared to the LTE and HSPA bands it will have to backhaul. The 3.5GHz band is large and underused, however 3GPP is currently incorporating this into both UMTS (HSPA) and LTE specifications [3GPP TR 37.801]. One operator has already announced its deployment in this band [Broadband11].

'Free' unlicensed spectrum – worth every penny: The ISM bands at 2.4 and 5.8GHz provide a large amount of freely available bandwidth. However, in the hotspots where additional data capacity is needed most, ISM spectrum is likely to be already heavily used by Wi-Fi for data off-load, Bluetooth and other equipment. All these other transmissions represent unwanted interference which will reduce signal quality, throughput and general quality of solutions using these bands. Niche opportunities exist where interference can be reduced in isolated locations or with smart antenna technologies. Backhaul in unlicensed spectrum represents a hostage to fortune: it may work on the day of installation, but could be fatally disrupted by a consumer's new device tomorrow.

NLoS backhaul is built into the LTE standards: A feature rather like NLoS backhaul, called 'in-band relay', is included in the LTE advanced standard [3GPP TR 36.814] where a base station can use half of the access spectrum to backhaul signals to a connected 'donor' cell site. Whilst this is good for extending coverage for early deployments, spectral efficiency for end-user traffic is effectively halved, so it is not a capacity-enhancing solution needed to meet increasing demand.

Microwave – plenty of spectrum, a mature technology for fixed links: Large amounts of bandwidth are available at 'microwave' [Network12] frequencies from 10-60GHz, which in turn means lots of capacity. These frequencies are already widely used for high-capacity fixed communication links with Point-to-Point and multipoint topologies. The small wavelength at these frequencies brings a mix of benefits and challenges. On the plus side, high-gain, compact antennas are easy to build which

improves link budgets, however such antennas need to be carefully aligned to the other end of the link. The short wavelength also means that effectively line of sight is the only option as diffraction and penetration around or through buildings and trees incurs high losses. This can be turned to an advantage as the high attenuation helps reduce interference from nearby links wishing to re-use the same frequencies. Given the maturity of technologies and availability of spectrum for high-capacity backhaul, microwave looks to be the mainstay of small cell backhaul solutions. This should not be a surprise; microwave backhaul has historically been used more for backhaul links than all the other technologies put together.

E-band: Around 10GHz of spectrum is available between 71-76 and 81GHz, a 'window' between peaks of high atmospheric absorption. Regulators have made this spectrum available under light licensing conditions to encourage innovation. Although background attenuation is several times higher here than in microwave bands, short high-capacity links of over 1 km are possible [Design06]. New technologies are emerging to exploit the wide bandwidths available to produce high-capacity Point-to-Point links. High-gain and highly-directional compact antennas are easily made given the very short wavelengths, and in fact are needed to make the link budget work. Wider beam widths for easier alignment or multipoint topologies are a challenge.

3.1.5 ADSL Characterization

xDSL technology offers fix broadband services over the existing copper twisted pair infrastructure. According to European Digital Agenda Scoreboard 2013, xDSL is still a predominant access technology in the EU broadband market with share 73.8% in January 2013. The number of xDSL lines increased by 1.7 million in 2012, which mostly contributed by VDSL line addition. However the VDSL share in Europe is still small around 3.9% of xDSL lines.

3.1.5.1 *Attainable Rate*

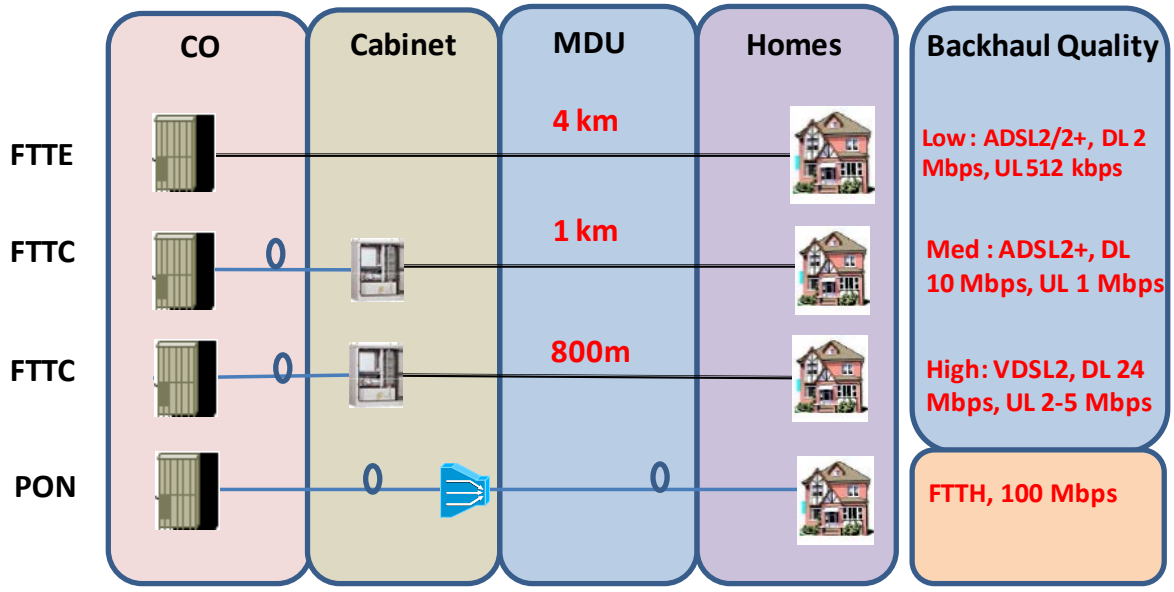


Figure 6. FTTx & xDSL Backhaul Quality Model [Hariyanto12].

The study of xDSL performance could not be separated by FTTx technologies implementation, since both are complementary to each other. Performance of transmission technologies over copper have been evaluated for the following reference architectures:

a. FTTE (DSLAM or MSAN in Exchange) which use technology: ADSL2/2+ technology as the last mile access

b. FTTC (MSAN in street cabinet) which uses technology: ADSL2/2+, VDSL2 (profile 8b, use the same Tx level as ADSL2/2+)

c. FTTB (ONU or MSAN at building) which uses technology: VDSL2 (profiles 17a and 30a)

Figure 6 shows xDSL backhaul quality model showing different broadband wireline configurations.

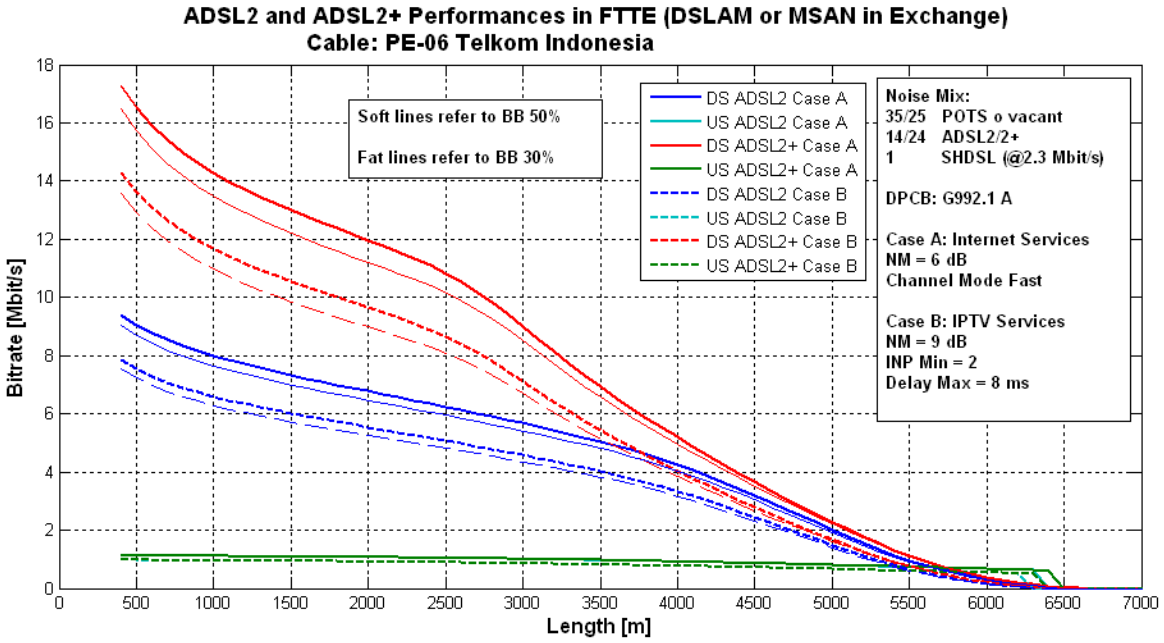


Figure 7. ADSL2 & ADSL2+ Performance in FTTE/MSAN in Exchange [Hariyanto12].

Based on experimental and study conducted by TELKOM in FREEDOM Project, xDSL attainable data-rate can be shown in Figure 7 and Figure 8 representing xDSL under FTTE and FTTC configuration respectively. While for FTTB and FTTH most operators will adopt GPON technology, which will be explained in the next chapter.

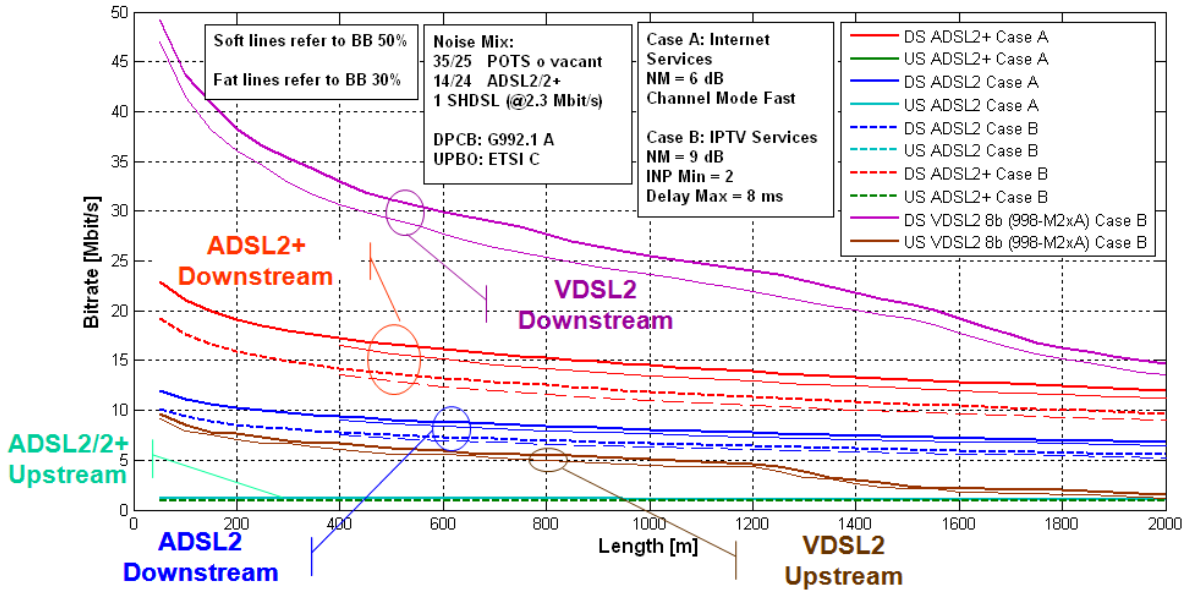


Figure 8. ADSL2, ADSL2+, VDSL Performance in FTTC (MSAN at street cabinet) [Hariyanto12].

3.1.5.2 *Transmission Delay*

The transmission delay of xDSL over various bandwidth profiles can be seen in Figure 9.

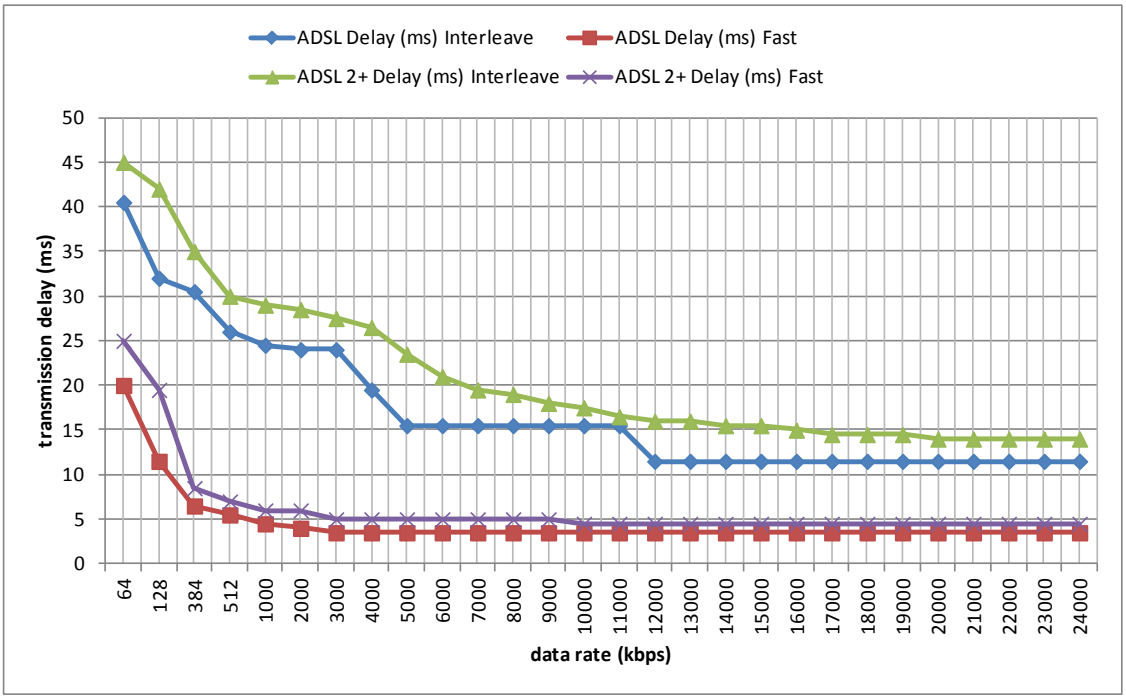


Figure 9. Average delay transmission of ADSL/ADSL2+ over various bandwidth profile [Hariyanto12].

The transmission delay of ADSL and ADSL2+ for various bandwidth profiles (i.e 64 kbps, 128 kbps, 512 kbps, 1 Mbps, 2 Mbps, 3 Mbps) and transmission modes (fast & interleave) are presented. In general, xDSL delay is mostly caused by xDSL operation mechanism when transmitting signals. The mechanism is to provide additional protection of transmitted data against noises (crosstalk, pulse interference) and disturbances. Several mechanisms are available including Reed-Solomon forward error correction, cyclic redundancy check, interleaving, scrambling, etc. Figure 8 show that xDSL with fast mode is outperform the ones with interleave mode.

The second reason of additional xDSL layer delay is caused by the operations necessary for DMT modulation due to Fourier transform, trellis coding & cyclic prefix operation process. However, these delays are usually constant and independent on the transmitted data and their amount. Today, thanks to the relatively powerful xDSL processors implemented in DSLAMs and modems, these delays caused by xDSL operations are very low and can be usually ignored [Vodrazka13].

xDSL transmission delay can be formulate as

$$t_t = t_s + t_{tx} + t_p + t_b \tag{2}$$

where t_s is a serialization data link delay, t_{tx} is a transmission delay, t_p is a delay necessary for xDSL operations both at the ATU-R modem and ATU-C DSLAM side (typically several or tens milliseconds) including forward error correction (FEC), interleaving, etc. And the last parameter is t_b is a delay used for buffering transmitted packets in network nodes.

Serialization delay t_s is defined as

$$t_s = \frac{8n_p}{B_l} \tag{3}$$

where, n_p is a length of packet in bytes (usually between 8 and 1490) and B_l is a link bit rate in bps.

Transmission delay t_{tx} can calculated from distance d between all network nodes and the velocity of signal propagation v as

$$t_{tx} = \frac{d}{v} \tag{4}$$

Buffering delay t_b is a pseudo-random value which depend on xDSL traffic reaching queueing buffer. [Kovac12] model this delay using a generalized Pareto distribution with probability density function given by

$$PDF(t_b) = \frac{1}{\sigma\left(1+\frac{\xi(t_b-\mu)}{\sigma}\right)^{\frac{1}{\xi}+1}}, t_b \geq \mu \tag{5}$$

where μ, σ and ξ are parameters of generalized Pareto distribution. Kobayashi02 suggest the variation of this delay uses generalized Gamma distribution. TELKOM will observe and model both delay and jitter distribution of mobile traffic through measurement in testbed, which will be presented in later chapter.

3.1.5.3 *xDSL Quality of Service*

xDSL supports layer-2 type of service (TOS) according to ATM service terminology as can be seen in Figure 10. Since femtocell supports several QoS types according to 3GPP term (conversational, streaming, interactive and background), the most appropriate TOS setup is constant bit rate (CBR) which allows fixed, guaranteed resource for both realtime application and non-realtime application over the femtocell.

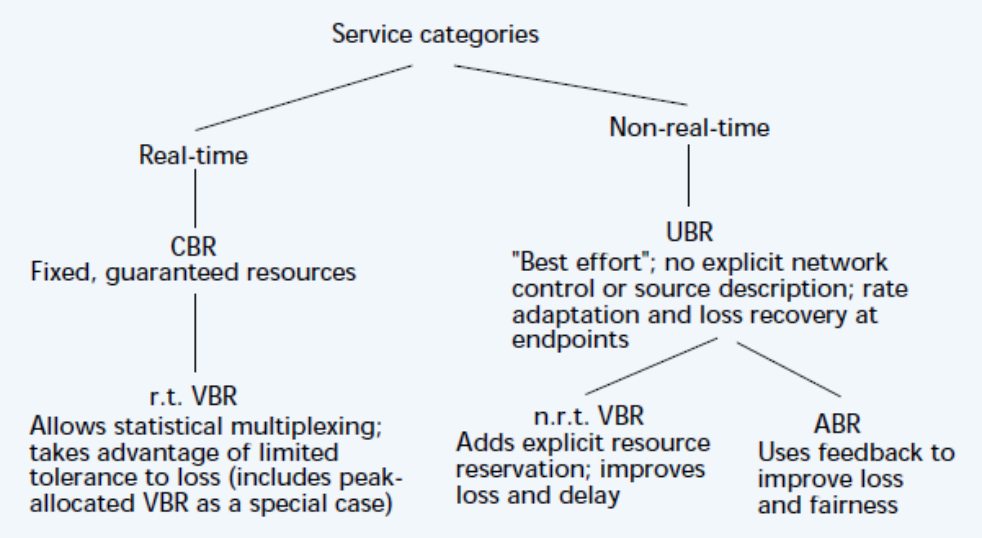


Figure 10. Architecture of ATM service categories. The fundamental dichotomy is between real-time and non-real-time applications [Garret96].

In order to support femtocell service, it is recommended that FAP services are transmitted over a separate channel (a dedicated physical virtual connection, or PVC), so that the home internet traffic will not affect the femtocell service performance. Other PVC such as home internet or IPTV can share

the same link but with lower priority or different QoS treatment. For example the channels for home internet service can be set to unspecified bit rate (UBR), for IPTV service set to variable bit rate for realtime application (VBR-rt) and femtocell service set to constant bit rate (CBR).

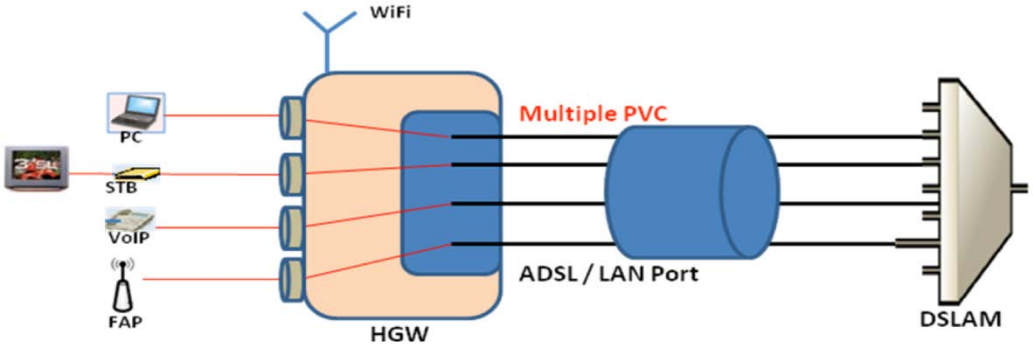


Figure 11. Several xDSL services mapped into multiple PVCs.

To maintain end-to-end mobile QoS from xDSL home gateway to femtocell gateway, the network should be able to support three QoS mechanisms including traffic classification, DiffServ and bandwidth reservation from FAPs, access network, edge, backbone, internet cloud and mobile core network. The SLA between mobile network, ISP and customer especially corporate customer are required. Figure 12 shows end-to-end network architecture to support femtocell network in TELKOM Indonesia. This configuration may vary from operator to operator. The SLA based network will allow mobile operator to locate the FAP-GW within mobile core network avoiding fluctuation in the internet.

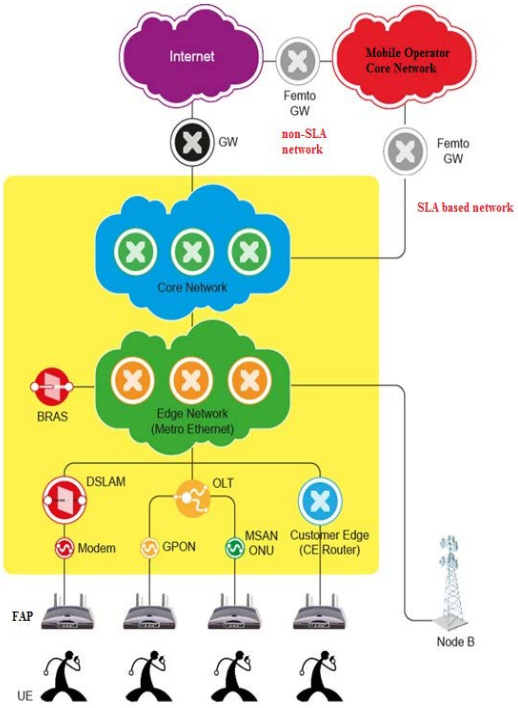


Figure 12. E2E network architecture for femtocell network.

In the SLA-based network, three QoS mechanism can be explained as following:

1) Traffic Classification. It will allow the femtocell traffic will be differentiated based on its VLAN ID, Class of Service (COS), and DSCP. Incoming packets will be mapped and treated according to QoS requirement.
2) DiffServ. In the uplink side FAP will mark the outgoing packet and give the priority to its services following 3GPP classification. BRAS, and other intermediate node in edge network and backbone must be able to map the services into different priority queuing according to the

DSCP value. Based on DSCP value, the network node will be able to schedule individual queue to ensure the QoS.

3) Bandwidth reservation. To ensure that the network is engineered to anticipate the traffic growth, specific amount of bandwidth should be reserved from access, edge to core network. TELKOM has calculated the bandwidth minimum requirement in the access part. When it comes to commercial, this effort should be carried further to engineer bandwidth requirement in edge network and backbone. Currently TELKOM allocated 100 Mbps to each segment from xDSL to metro Ethernet.

3.1.6 Fiber characterization

3.1.6.1 *GPON/FTTxTechnology*

There are three major PON standards such as BPON (Broadband PON), GPON (Gigabit-capable passive optical networks) and EPON (Ethernet passive optical networks). GPON and EPON are two prominence standards that are open to new improvement both for vendors and operators.

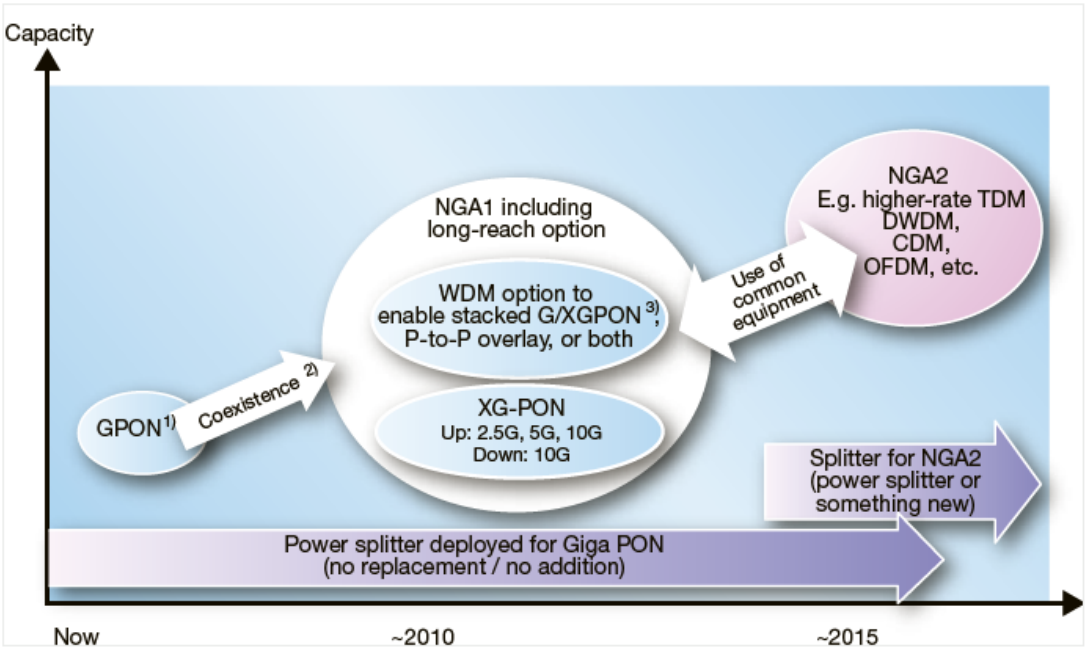


Figure 13. Future of GPON technology [Trojer08].

After some initial BPON deployments, the industry realized that a BPON-based optical distribution network (ODN) cannot be upgraded to next-generation technologies. The logistics of upgrading an entire PONwere daunting, and the installing cost of parallel upgrade PON was prohibitive. There were many candidates for next-generation of PON systems. The ITU community drafted G.984.5 that reserved wavelengths for use by next generation applications without constraining them. The EPON community saw limitation of bandwidth as the most problem and immediately began work on 10Gbps EPON (802.3av) as the successor to 1Gbps EPON.

The next-generation architecture (NGA) will have two stages of NGA1 and NGA2. NGA1 is compatible with GPON deployments in accordance with G.984.5. Some NGA1 candidates are:

- XGPON1, NGA that supports 10Gbps downstream data rate and 2.5Gbps upstream data rate.
- symmetric XGPON2, which supports10 Gbps downstream and upstream; and
- WDM option to overlay PONs and point-to-point connections on the same fiber infrastructure in G.984.5 enhancement bands.

3.1.6.1.1 GPON (ITU-T G.984.x standards)

Access Network is series of wires, cables, and equipment lying between a consumer/business termination point and the operator point. GPON is one of the backhaul technology used to connect between operator and customer. GPON systems are characterized, in general, by an Optical Line Termination (OLT) system and an Optical Network Unit (ONU) or Optical Network Termination (ONT) with a passive Optical Distribution Network (ODN) interconnecting them.

In order to derive backhaul quality model for GPON as femtocell backhaul, we will use simulation tools in supporting several GPON technologies. Even though the study initially performed to study Cloud implementation, we assumed the bandwidth requirement for the service is relevant also to support femtocell implementation.

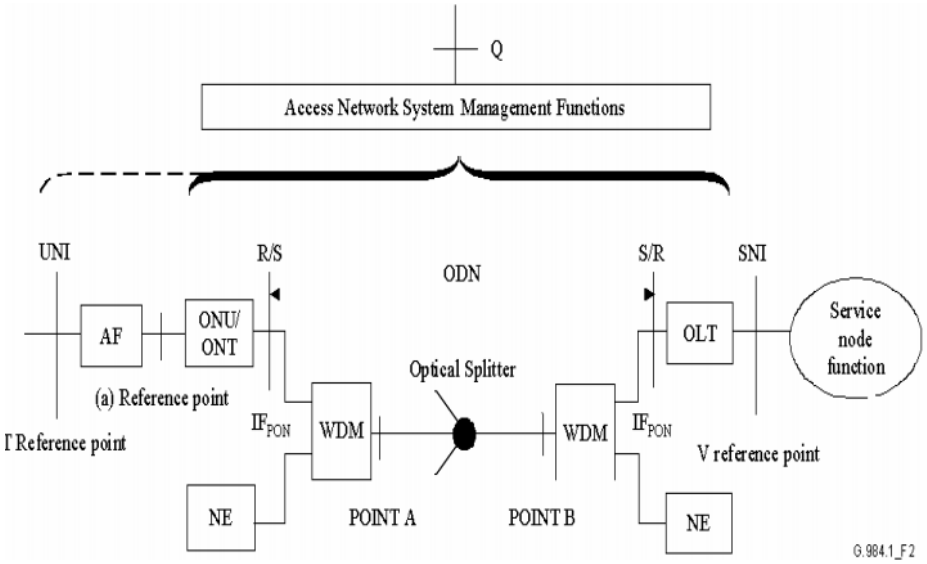


Figure 14. Reference configuration for GPON [ITU-TG.984.1].

Maximum mean signal transfer delay: [ITU-TG.984.1]

- GPON must accommodate services that require a maximum mean signal transfer delay of 1.5 ms.
- Specifically, GPON system must have a maximum mean signal transfer delay time of less than 1.5 ms between UNI (User Network Interface) – SNI (Service Node Interface) (or (a)-AF (Adaptation Function) – SNI, depending on operator's preference).
- Although a section of the delay measurement is UNI – SNI for FTTH system or AF – SNI for the other application in ITU-T Rec. G.982, in a GPON system the reference points are not restricted by the system configuration.

GPON technology is composed of OLT, ONUs and Optical Distribution Network (ODN) which constitutes the optical transmission media for connecting OLT to ONUs. The characteristics of ODN are very important in optical access network design. Generally it consists of the passive optical elements such as: single mode optical fibers, passive branching components, optical fiber connectors, passive optical attenuators and splices.

Table 3 shows the link budget of GPON technology. This budget covers all optical components between OLT and ONU.

Items	Unit	Path Loss
Minimum optical loss 1490 nm	dB	13
Minimum optical loss 1310 nm	dB	13
Maximum optical loss 1490 nm	dB	28
Maximum optical loss 1310 nm	dB	28

Table 3. Path loss budget for the GPON technology.

3.1.6.1.2 Power Budget

The power of transmitter and sensitivity of receiver are two important parameters that define the quality of the access network. Table 4 shows parameters of commercially burst mode transceivers that capable of supporting 1.25 Gbps.

The available power budget, typically, is around 22 dB and 23dB. Based on these values the total loss and maximum reach of the network is known and can be calculated as *P* [Cale07].

$$P = FCA \cdot L + SL + Penalties$$

Where: *P* is power budget, *FCA* is fiber cable attenuation in dB/m, L is a distance and *SL* is a splitter loss. *Penalties* are the additional costs such as losses at splices and connectors.

ONUs	L	λ	FCA	SL	Penalties	Required Power Budget
16	10 km	1310nm	0.4dB/m	14.5dB	2.5dB	21dB
16	20 km	1550nm	0.3dB/m	14.5dB	2.5dB	23dB
32	10 km	1310nm	0.4dB/m	17dB	2.5dB	23.5dB
32	20 km	1550nm	0.3dB/m	17dB	2.5dB	25.5dB

Table 4. The minimum power budget for different PON configuration.

3.1.6.1.3 Peak Data Rate

GPON technology design is based on ITU Standard G.984 and G.987. For ITU G.984, the data rate for Downstream is 2.488Gbits/s and for Upstream is 1.244/2.488Gbits/s, while on G.987 the data rate for Downstream is 10Gbits/s and Upstream is 2.488/10Gbits/s. This technology uses GPON Encapsulation Method (GEM), fragmented packets or ATM. GPON 2.448 Gbps in the downstream direction and 1.24416 Gbps in the upstream direction. GPON in GEM mode can achieve ~ 95% efficiency of its usable bandwidth.

Item	GPON Technology	
Line Rate	Downstream	1.24416/2.48832 Gb/s
	Upstream	1.24416 Gb/s
	Bit rate after scrambling line coding	1.24416 Gb/s
Guard time	Laser on-off	25.7 ns
	Premble and delimiter	70.7 ns
Frame size	General encapsulation method (GEM)	≤ 1523 bytes
Overhead for bandwidth allocation	Status report message	2 bytes

Table 5. Specification of GPON Technology.

3.1.6.2 *Metro Ethernet (ME) Access Network Characteristics using EPON*

Using passive optical network, Ethernet in the First Mile PON (EFM PON) build point to multipoint fiber topology that supports a speed of 1 Gbps for up to 20km. Subscriber connected to the dedicated distribution fiber to the site but share Optical Distribution Network (ODN) trunk to the Central Office (CO). [IEEE802.3ah].

EPON is based on Ethernet Standard, unlike GPON which is based on ATM standard. Figure 15 shows Point to Point Ethernet, Curb Switched Ethernet and Ethernet PON.

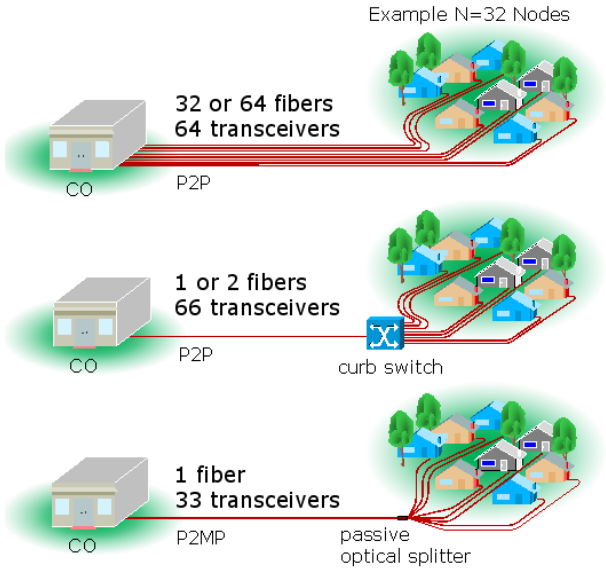


Figure 15. Point to Point Ethernet, Curb Switched, and EPON [Forum05].

EPON System configured in full duplex mode (no CSMA/CD) in single fiber point-to-multipoint (P2MP) topology. Subscriber or ONU, only sees traffic from headend; each subscriber cannot see traffic transmitted by other subscriber. Peer-to-peer communication is done via headend or OLT. Headend allows only one subscriber at a time to transmit using a Time Division Multiple Access (TDMA) protocol.

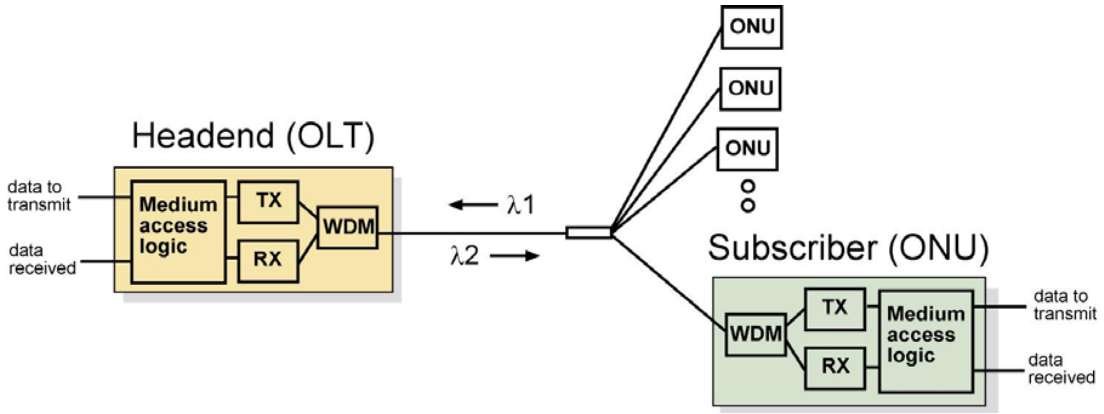


Figure 16. EPON configuration [Forum05].

EPON system uses an optical splitter architecture multiplexing signal with different wavelengths for upstream and downstream as such:

- 1490nm downstream
- 1310nm upstream

EPON can be deployed in Ethernet access platform, with both point to point or point to multi point access card as shown in Figure 17.

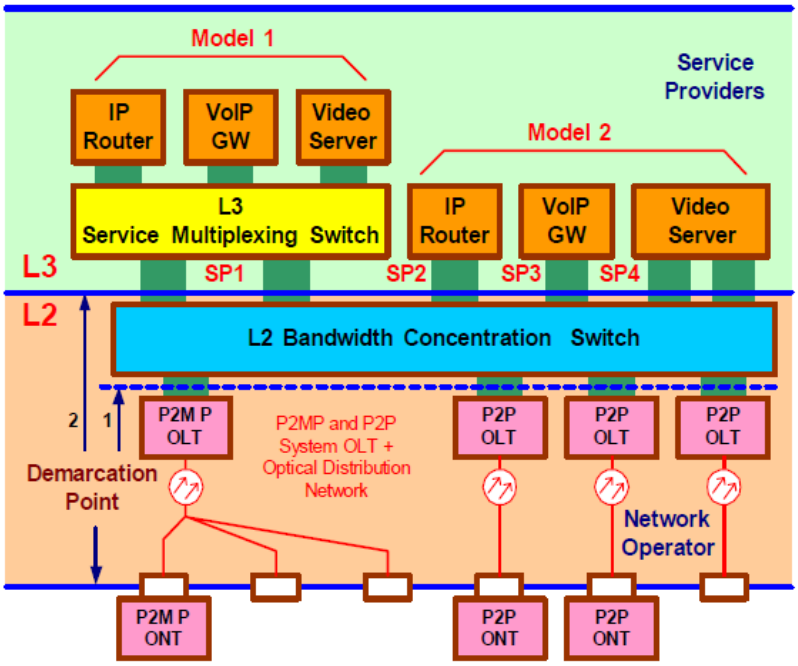


Figure 17. EPON in the Ethernet Access Model [Forum05].

3.1.6.2.1 Link Budget

To support data rate and distance of 10 km and 20 km IEEE802.3ah defines following power budget class:

- *Low power budget class* supports P2MP media channel insertion loss of 20 dB e.g., a PON with the split ratio of at least 1:16 and the distance of at least 10 km.
- *Medium power budget class* supports P2MP media channel insertion loss of ≤ 24 dB e.g., a PON with the split ratio of at least 1:16 and the distance of at least 20 km or a PON with the split ratio of at least 1:32 and the distance of at least 10 km.
- *High power budget class* supports P2MP media channel insertion loss of ≤ 29 dB e.g., a PON with the split ratio of at least 1:32 and the distance of at least 20 km.

Power budget for each class represented by PRX-type and PR-type power budget as follows:

- PRX–type power budget describes asymmetric-rate PHY for PON operating at 10 Gb/s downstream and 1 Gb/s upstream over a single SMF.
- PR–type power budget describes symmetric-rate PHY for PON operating at 10 Gb/s downstream and 10 Gb/s upstream over a single SMF.

3.1.6.2.2 Peak Data Rate

As shown in Table 6 above EPON network with respective power budget support downstream line rate of 10.3125Gbits/s and upstream line rate of 1.25Gbits/s or 10.3125Gbits/s.

3.1.6.2.3 Transmission Delay

Metro Ethernet Forum Carrier Ethernet Class of Service Standard (MEF23) defines maximum Frame Delay for class H Metro Performance Tier as 10ms.

Description	Low Power Budget		Medium Power Budget		High Power Budget		Units
	PRX10	PR10	PRX20	PR20	PRX30	PR30	
Number of fibers	1						-
Nominal downstream line rate	10.3125						GBd
Nominal upstream line rate	1.25	10.3125	1.25	10.3125	1.25	10.3125	GBd
Nominal downstream wavelength	1577						Nm
Downstream wavelength tolerance	-2, +3						Nm
Nominal upstream wavelength	1310	1270	1310	1270	1310	1270	nm
Upstream wavelength tolerance	± 50	± 10	± 50	± 10	± 50	± 10	
Maximum reach	≥10		≥20		≥20		km
Maximum channel insertion loss	20		24		29		dB
Minimum channel insertion loss	5		10		15		dB

Table 6. Primary attributes of all power budget types.

3.1.6.3 *Key Observation and Methodology*

In terms of backhaul, a HeNB is designed to use a broadband IP backhaul which is usually shared IP connection with other broadband users in residential areas. It is contrary to SCeNB backhaul which uses a dedicated leased line in order to guarantee non-blocking configuration. Femtocell performance depends on many aspects including radio channel quality condition and under-lined backhaul quality. As the part of femtocell performance, the existence of congestion in the backhaul link will affect overall femtocell performance.

3.1.6.3.1 Key Observations

The quality of broadband IP connection highly depends on the following issues [Epitiro08]:

- Allocated / available bandwidth per ISP plan (Mbps);
- Additional IP-based activities in the home/office;
- Network load by time-of-day;
- Efficiency of routing from the subscriber premises to the core network.
- ISP traffic management policies (QoS, priority traffic, etc.) according to SLA between ISPs, customer and mobile network operators.

For SCeNB, it assumed that the ISP provides a dedicated link for mobile backhaul. Logically, internet/intranet traffic will not affect the performance of mobile backhaul. There are several backhaul types, which represent last mile-connection for smallcell in Business Scenario 1 (BS1 - Corporate Customer inside Building), Business Scenario 2 (BS2 - Users in Public Places) and Business Scenario 3 (BS3 - Residential User inside their Premises). Except routing efficiency and network load we are considering to observe smallcell performance with related to the above issues.

The objectives of backhaul modeling are:

- Model the impact of specific mobile application services into a backhaul network performance (delay, jitter, packet loss, and throughput).

- Model the relation between specific number of users into a backhaul network performance (delay, jitter, packet loss, throughput) for various scenario including BS1, BS2, as well as BS3.

The backhaul measurement will use Spirent traffic generator in TELKOM Oasis Laboratory. Cloud traffic as main traffic will be generated by this simulator.

3.1.6.3.2 Performance Parameters

As performance indicator, observation will focus on delay, throughput, jitter, and packet loss model over specific test configuration.

The observation considered the reference stated in [Saunders09] and [Epitiro08], which described the requirement to provide good quality voice calls and sufficient support to data services. These references suggested the broadband IP link will provide a minimum performance with:

- Less than 150 ms round-trip delay (more than 200 ms will not be practical for two ways voice conversation);
- Less than 40 ms jitter;
- Less than 3% packet loss; however, packet loss is typically “bursty” by nature, and as such, average rates below 0.25% should be maintained;
- At least 1 Mbps in downlink, ie. from the broadband IP provider network to the HeNB GW;
- At least 256 kbps in uplink, ie. from the HeNB GW to the broadband IP provider network.

3.1.6.4 *Backhaul Observation Scenario*

As can be seen from Test bed configuration, TELKOM test bed consists of GPON link through various access technologies to the BS1, BS2 and BS3. Observation will use Spirent simulator for generating packet and backhaul experiment.

Spirent generates some traffic to show the background traffic and cloud traffic with proportional regards to some network assumption and finding of any papers that show the variant of traffic characteristic based on three kinds of Business Scenario such as residential, public and enterprise areas.

The result of the experiment will show the correlation between backhaul capacity and the Quality of Service of application and information regarding many parameters of backhaul QoS.

3.1.6.5 *Test Parameters of Smallcell using FTTx & xDSL as Backhaul*

Femtocell deployment uses xDSL/MSAN as a backhaul is assumed to be used for residential, which is asymmetric in nature and FTTx/GPON is assumed to be used for indoor office and public areas. GPON is more reliable, higher throughput and symmetrical between downstream/upstream.

Below are the scenarios to be considered in the femtocell over GPON measurement campaign:

- Scenario 0: check initial E2E network performance, without HeNB, in order to capture initial network performance including access part, backbone part and internet exchange.
- Scenario 1: observe minimum bandwidth requirement of a single HeNB considering mix services, various UE types and 4, 6 and 8 simultaneous users accessing it. The applications will be accessed through various UE types including, smartphone and tablet PC. Each user will access generated internet services such as HTTP, FTP, VoIP and Video Service. The traffic flooding a HeNB will be observed through Spirent Analyzer while MSAN is set to 10 Mbps (downstream) and 1 Mbps (upstream) emulating BS3 (residential area).

- Scenario 2: Verify bandwidth requirement of a HeNB taken from Scenario 1, in order to figure out minimum bandwidth requirement for GPON link. The application service will be generated using traffic models available in a measurement tools. In order to verify the minimum bandwidth requirement from a femtocell, we define various mix services as close as the mix traffic as shown in Table 8.
- Scenario 3: verify femtocell performance in the existence of background traffic in GPON backhaul. Background traffic in this case is the traffic generated by other femtocells occupying the same backhaul link. The background will use traffic generated from the spirent measurement connected to the ONT modem. To represent BS1 the traffic from 500 users are generated, while for BS2, the traffic from 2500 users are generated.

Traffic Mix	Traffic Source	URL	File Transfer	Video Streaming	Voice	Notes
Smartphone	Real	Detik.com	4shared.com (5 MB)	Youtube	Skype	Real]NB traffic for scenario 1
Smartphone VoIP Only	Real					Real HeNB traffic for scenario 1
Voip Only	Generated				All G.729	Scenario 2, All FUE access VoIP
Mix 1	Generated	HTTP 1.1	FTP up to 50 MB	H.264, small resolution	G.729	Scenario 2, 3 and 4
Mix 2	Generated	HTTP 1.1	FTP up to 10 MB	H.264, higher res.	G.729	Scenario 2, 3 and 4
PC Background 1	Real	Detik.com	4shared.com (50 MB)	Youtube	Skype	Scenario 3 and 4
PC Background 2	Generated	HTTP 1.1	FTP Upto 50 MB	Youtube	Skype	Scenario 3 and 4

Table 7. Summary of traffic mix of femtocell performance using xDSL & GPON as the backhaul.

In this planning test there are many parameters with three Business Scenarios. Table 8 shows the parameters and values of each Business Scenario.

Parameters	**Residential Business Scenario 3 (BS3)**	**Office Business Scenario 1 (BS1)**	**Public Business Scenario 2 (BS2)**
Service Load: [Sandvine13] • HTTP 1.1 • FTP • RTSP • SIP	 57.6 % 7 % 10 % 3 %	 37.6 % 9 % 46.8 % 6.3 %	 37.6 % 9 % 46.8 % 6.3 %
Number of users/HeNB	4-8	4-50	50-100
Number of HeNB served by a single backhaul	1-4 HeNBs	1-4 HeNBs	1-8 HeNBs
Traffic Model	Random	Random	Random
Technology	MSAN/ME	GPON/ME	GPON/ME
Backhaul bandwidth	1 Gbps	1 Gbps	1 Gbps
Shared backhaul bandwidth	Yes, Home Internet	Yes, Corporate Intranet or Internet	No

Table 8. Test Parameters.

3.1.7 Measurement Scenarios

The study of femtocell backhaul characteristics will benefit to the following activities:

- Minimum bandwidth requirement to support a HeNB should be used to justify the optimal bandwidth for femtocell. This information may be used as later reference to calculate cost structure of femtocell deployment.
- End-to-end femtocell performance will be observed and the pattern of background traffic for both GPON and MSAN will be modeled.

3.1.7.1 *Test Plan Configuration*

3.1.7.1.1 General Test Architecture

Figure 18 shows the general test architecture. The test architecture uses traffic generator to simulate server for background traffic (cloud service, best effort data service, and real time application service), client traffic and running test scenarios.

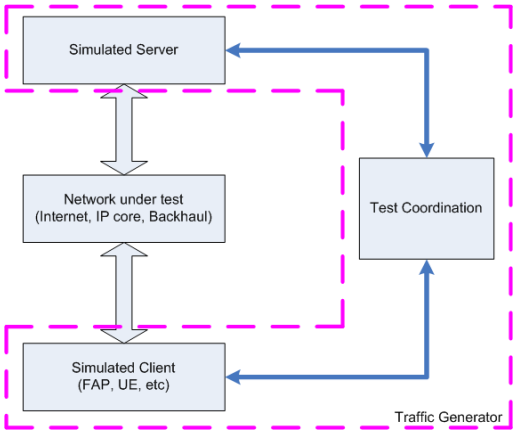


Figure 18. General test architecture.

3.1.7.1.2 Test Configuration for Business Scenario 1 (BS1)

Figure 19 shows test configuration for BS1. Traffic Generator generates traffic for server (cloud service, real time application, best effort data) and client (all HeNB clients) as described in BS1 scenario. Traffic Generator ports configuration:

1. Connected to IP-core to simulate traffic to and from server.
2. Connected to ONU to simulate traffic from and to client.

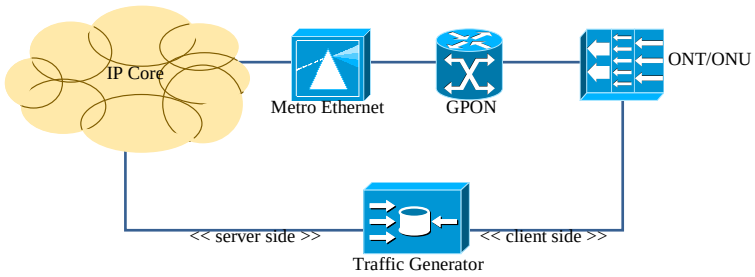


Figure 19. Test configuration for Business Scenario 1.

3.1.7.1.3 Test Configuration for Business Scenario 2 (BS2)

Test configuration for BS2 similar with test configuration for BS1, except ONT in BS2 represents an area being served in public facilities, therefore it is expected to handle larger users (2500 users for instance).

3.1.7.1.4 Test Configuration for Business Scenario 3 (BS3)

Figure 20 shows the test configuration for BS3. In the BS3 network, Modem represents an area in customer home.

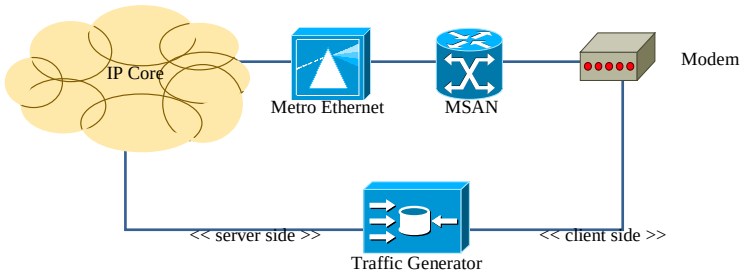


Figure 20. Test configuration for Business Scenario 3.

3.1.7.2 *Test Scenario (TS)*

The purpose of this scenario is to measure backhaul network performance (delay, jitter, and packet loss) in effect of number of user connected to femtocell. The number of user connected to the femto access point varied from single user to maximum number of user per business scenario, and total connected HeNB varied from single femto to maximum number of femtocell per business scenario.

Traffic Generator (port) connected to the IP Core to simulate server traffic and to the HeNB (ONT or MSAN in this case) to simulate customer traffic. Traffic generator was programmed to simulate traffic for desired number of users with traffic characteristic followed service load in Table 8.

For BS1 (corporate customers inside a building), the selected number of users is 500 with reference taken from small-cell deployment for Small Enterprise Customers in [Derrick01].

Whereas for BS2 (users in a public place), 2,500 users are taken with reference from Large Enterprise Customers in [Derrick01] as well.

The BS3 scenario (residential users inside their premise) uses 8, 6, and 4 users as test parameter, considering common number of users per access point (AP).

Business scenario	Scenario	Test scenario	Measured parameter
Corporate customer inside building (BS1)	Number of users	Perform 1-hour test with 2,500 users to get the backhaul network parameter models.	delay, jitter, packet loss
Users in public places (BS2)	Number of users	Perform 1-hour test with 500 users to get the backhaul network parameter models.	delay, jitter, packet loss
Residential users inside their premises (BS3)	Number of users	Step 1: Perform 1-hour test with varying traffic generated by traffic generator to reflect the number of users, from 1 to 40 users. Step 2: Perform 1-hour test with 8, 6, and 4 users to get the backhaul network parameter models.	delay, jitter, packet loss

Table 9. Summary of test scenario.

3.1.8 **Measurement Results and Backhaul Modeling**

The backhaul modeling was derived from the measurement result conducted inside TELKOM network testbed with scenarios and parameters described in the previous sections.

3.1.8.1 *ADSL-based Network*

ADSL-based backhaul is applicable for BS1 scenario. Backhaul throughput for 1-hour test can be shown in the below graphs. It was affected by the maximum throughput in the modem to MSAN which was set at 1 Mbps uplink / 10 Mbps downlink (asymmetric mode).

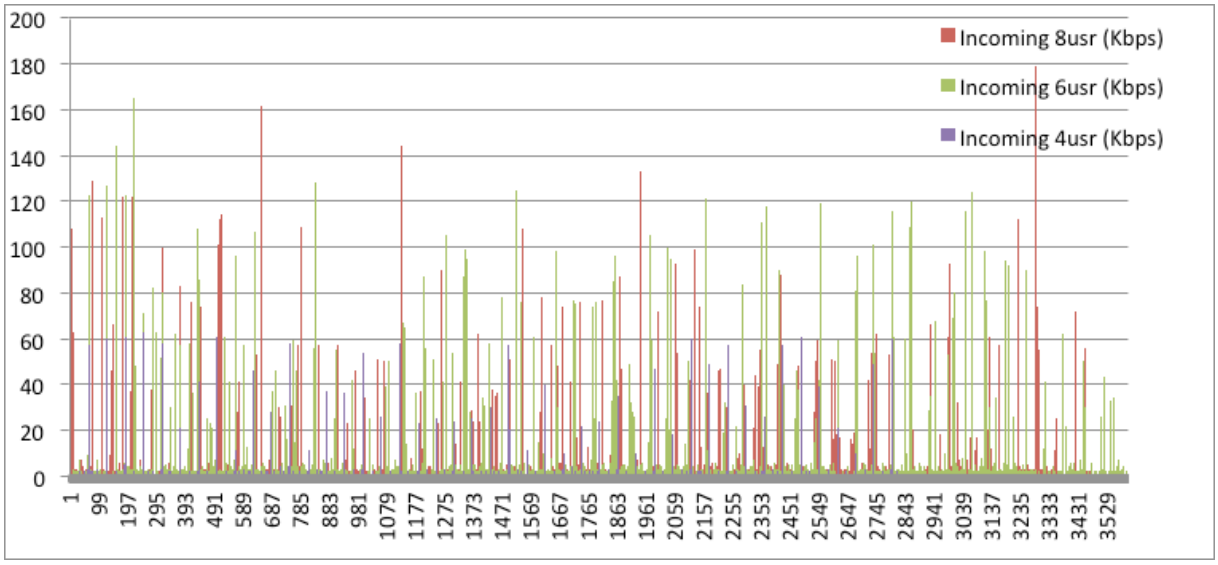


Figure 21. Incoming traffic throughput to server in backhaul during 3,600 sec. test.

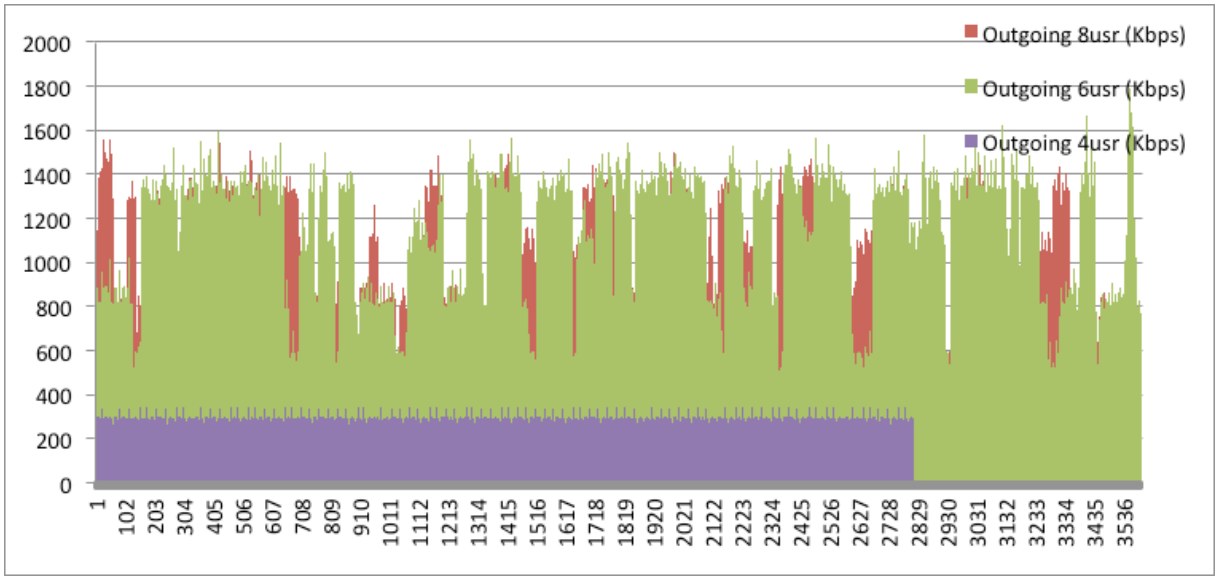


Figure 22. Outgoing traffic throughput from server in backhaul during 3,600 sec. test.

3.1.8.1.1 Delay

Measurement result for delay is shown in figure below. This graph is still deterministic in nature and will be processed further with 8, 6, and 4 users for stochastic modeling approach.

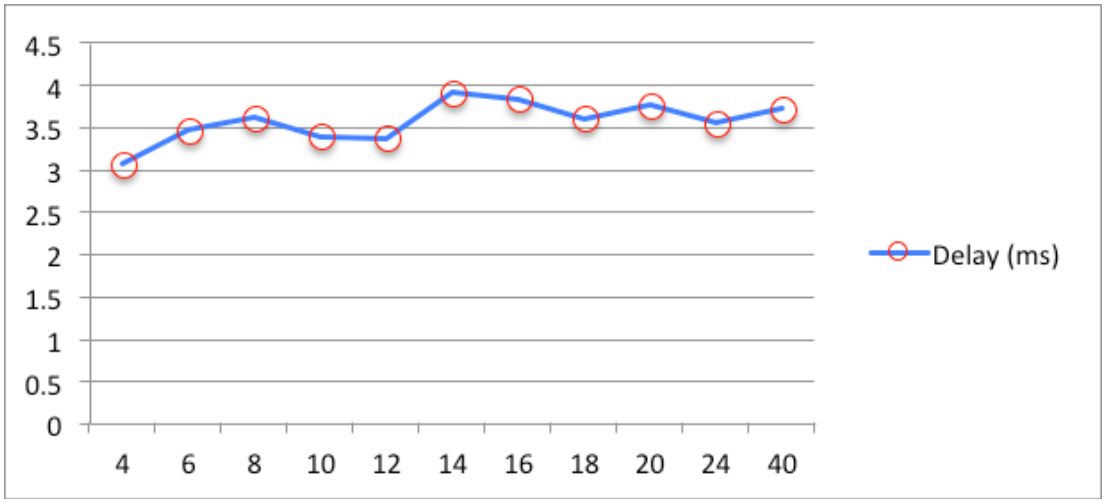


Figure 23. Delay measurement results for ADSL-based backhaul network.

Number of users	Delay (ms)
4	3.061
6	3.471
8	3.611
10	3.387
12	3.374
14	3.903
16	3.836
18	3.603
20	3.76
24	3.553
40	3.714

Table 10. Number of users vs delay for ADSL-based backhaul network.

One hour duration tests with 8, 6, and 4 users were conducted to obtain the statistical parameters for delay modeling. The models were compared with Generalized Pareto Distribution as suggested by [Vodrazka01]. Assuming t_{RTT} is the round trip delay from the client side, this delay is a pseudo-random value with probability density function given by the below equation:

$$f_{(\xi,\mu,\sigma)}(t_{RTT}) = \frac{\sigma^{\frac{1}{\xi}}}{(\sigma + \xi(t_{RTT} - \mu))^{\frac{1}{\xi}+1}} \quad (6)$$

where ξ, μ, σ are parameters of Generalized Pareto Distribution.

The tests found that Gumbel Maximum Value Type 1 Distribution was closer to the delay model for MSAN based backhaul network, which have the following equation:

$$f(t_{RTT}) = \frac{1}{\sigma} exp(-z - exp(-z))$$
$$z \equiv \frac{t_{RTT} - u}{\sigma} \quad (7)$$

The comparison can be depicted in the following chart for 8 users test. Figure 24 shows the result from Generalized Pareto Distribution, while Figure 25 shows the result of Gumbel Maximum Value Type 1 Distribution.

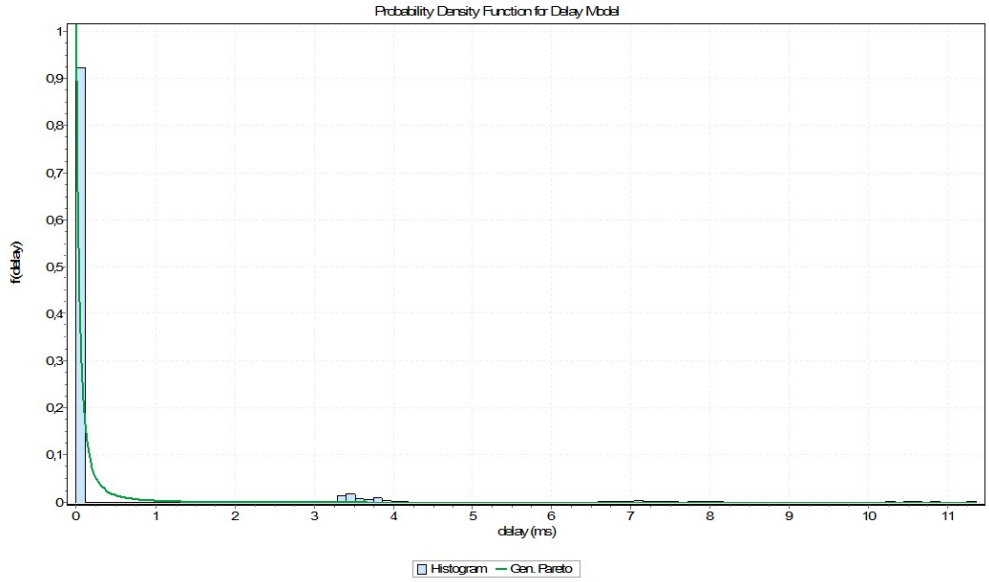


Figure 24. Generalized Pareto Distribution for 8 users delay model.

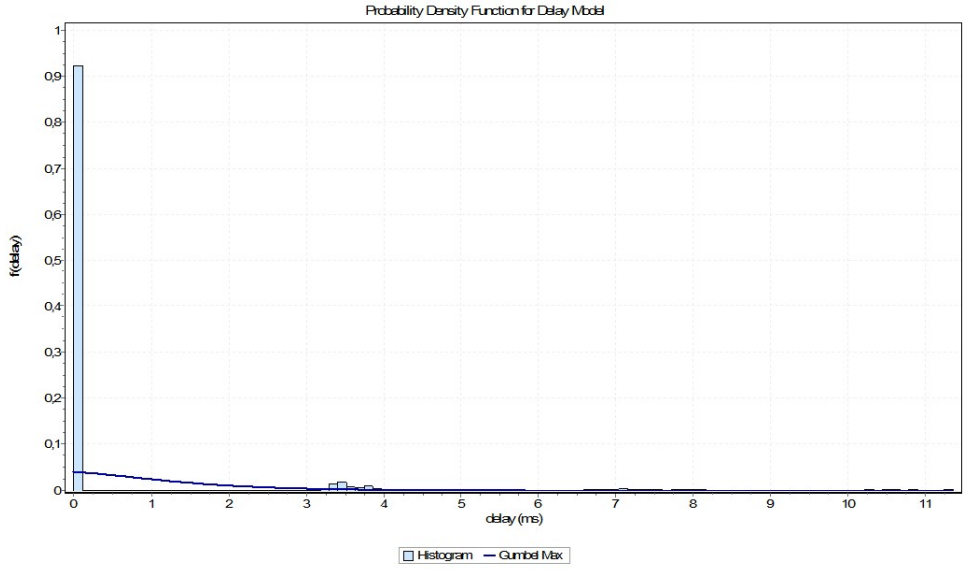


Figure 25. Gumbel Max Value Type I distribution for 8 users delay model.

Using Kolmogorov-Smirnov or K-S goodness-of-fit test, the test obtained the following parameters for both distribution.

Parameter	Number of users	K-S Stat	ξ	μ	σ
Generalized Pareto Distribution					
Value	8 users	0.61388	0.86747	0.04828	-0.02108
Value	6 users	0.61874	0.88152	-0.02414	0.05425
Value	4 users	0.65093	0.95386	-0.00189	0.00396
Gumbel Maximum Value Type I Distribution					
Value	8 users	0.4693	N/A	-0.2378	1.0066
Value	6 users	0.47319	N/A	-0.32828	1.3202
Value	4 users	0.50404	N/A	-0.15957	0.42192

Table 12. Parameters for delay model in MSAN-based backhaul.

3.1.8.1.2 Jitter

Graph in Figure 26 shows the 60-second jitter obtained from deterministic tests.

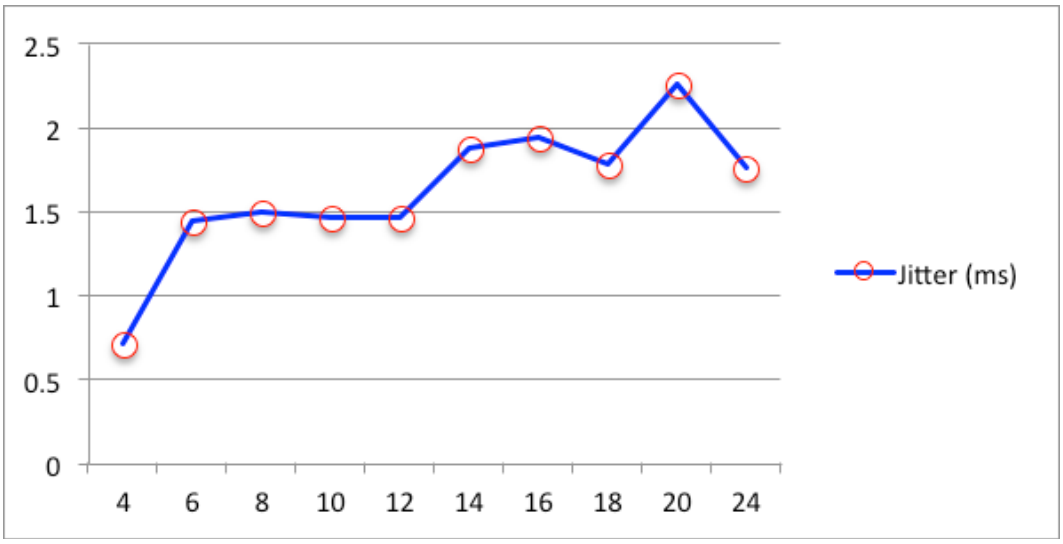


Figure 26. Jitter measurement results for MSAN-based backhaul network.

Number of users	Jitter (ms)
4	0.707
6	1.441
8	1.493
10	1.462
12	1.463
14	1.87
16	1.936
18	1.775
20	2.25
24	1.757

Table 11. Number of users vs jitter for MSAN-based backhaul network.

Further statistical tests for 8, 6, and 4 users found that Weibull Distribution was relatively closer to the 60-second jitter model for MSAN based backhaul network, which follows the below equation:

$$f(j_{RTT}) = \frac{\alpha}{\beta}\left(\frac{j_{RTT}}{\beta}\right)^{\alpha-1} exp\left(-\left(\frac{j_{RTT}}{\beta}\right)^{\alpha}\right) \quad (8)$$

the following graph models the probability density function (PDF) for jitter in MSAN-based backhaul.

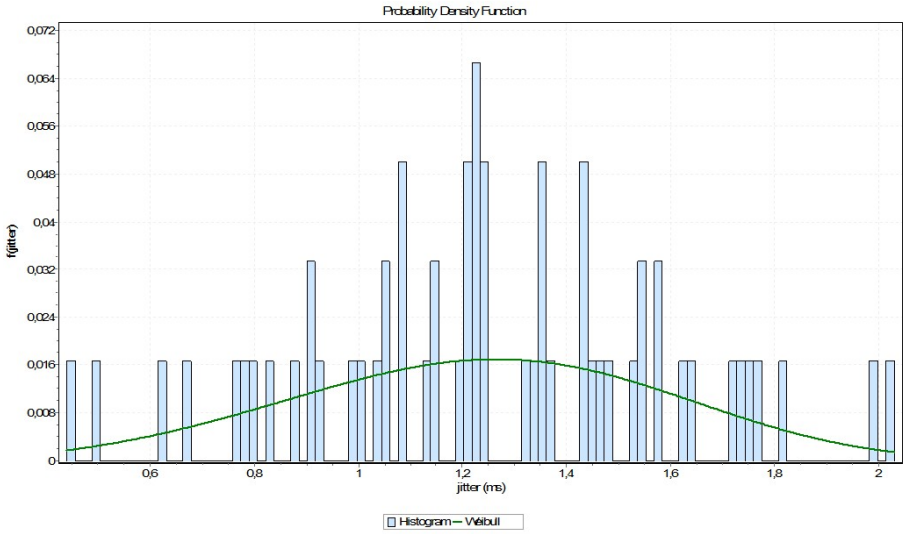


Figure 27. Probability density function chart for 8 users jitter model.

The following parameters are applicable for the above jitter model:

Parameter	Number of users	K-S Stat	α	β
Weibull Distribution				
Value	8 users	0.06443	3.8293	1.3683
Value	6 users	0.11718	6.117	1.7808
Value	4 users	0.2824	4.853	1.8177

Table 12. Parameters for jitter model in MSAN-based backhaul.

3.1.8.1.3 Packet Loss

Measurement result for packet loss is shown in Figure 28.

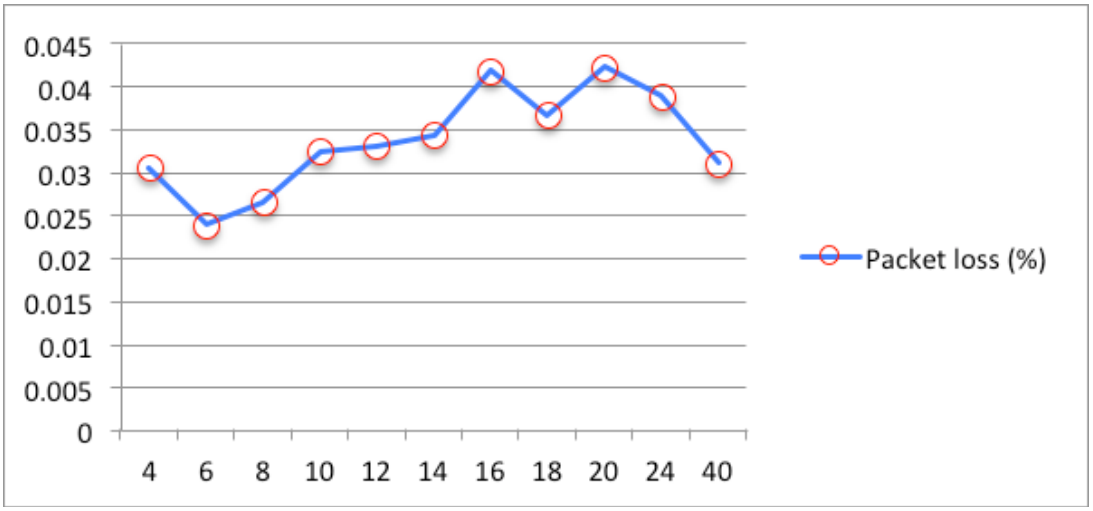


Figure 28. Packet loss measurement results for MSAN-based backhaul network.

Number of users	Packet loss (%)
4	3.05%
6	2.39%
8	2.66%
10	3.24%
12	3.31%
14	3.43%
16	4.18%
18	3.67%
20	4.22%
24	3.89%
40	3.11%

Table 13. Number of users vs packet loss for MSAN-based backhaul network.

Gumbel Maximum Value Type 1 Distribution was relatively closer to the packet loss model for MSAN based backhaul network, which basically has the same equation principle as previous one for delay model. The symbol $l_{svr\text{-}cli}$ is used to show packet loss parameter from server to client:

$$f(l_{svr-cli}) = \frac{1}{\sigma}\exp\big(-z-\exp(-z)\big)$$
$$z \equiv \frac{l_{svr-cli}-\mu}{\sigma} \qquad (9)$$

The following graph illustrates the probability density function (PDF) for downstream packet loss (ie from server to client) in MSAN-based backhaul. The symbol 0.1, 0.2, 0.3 and so forth in the x-axis below denotes 10%, 20%, 30% packet loss respectively.

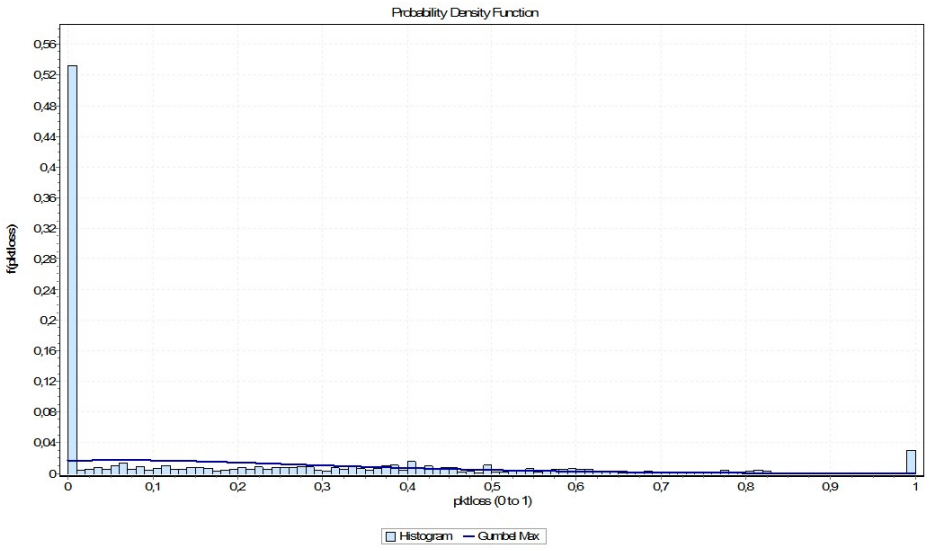


Figure 29. Probability density function chart for 8 users downstream packet loss model.

The following parameters are applicable for the downstream packet loss model:

Parameter	Number of users	K-S Stat	μ	σ
Gumbel Maximum Value Type 1 Distribution				
Value	8 users	0.27348	0.06210	0.20486
Value	6 users	0.39307	-0.00678	0.09888
Value	4 users	0.35208	0.02892	0.28998

Table 14. Parameters for downstream packet loss model in MSAN-based backhaul.

For upstream packet loss (i.e. from client to server), the same distribution function applies to the data. The table below lists all parameters for upstream packet loss model:

Parameter	Number of users	K-S Stat	μ	σ
Gumbel Maximum Value Type 1 Distribution				
Value	8 users	0.3919	0.11651	0.3377
Value	6 users	0.38438	0.11751	0.32759
Value	4 users	0.45643	0.01239	0.30093

Table 15. Parameters for upstream packet loss model in MSAN-based backhaul.

3.1.8.1.4 Throughput Model

Results from one-hour test are shown in the graphs and tables below. The traffic from client was randomly generated by the traffic generator in TELKOM's premise. It is referred to as upstream direction. The traffic from server side is referred to as downstream direction.

The below traffic (upstream to server and downstream from server) was measured in the server side.

From the experiment, Generalized Pareto Distribution with the below function was found to be relatively closer to the data. If t is defined as throughput for upstream traffic, then:

$$f_{(\xi,\mu,\sigma)}(t) = \frac{\sigma^{\frac{1}{\xi}}}{(\sigma + \xi(t-\mu))^{\frac{1}{\xi}+1}} \tag{10}$$

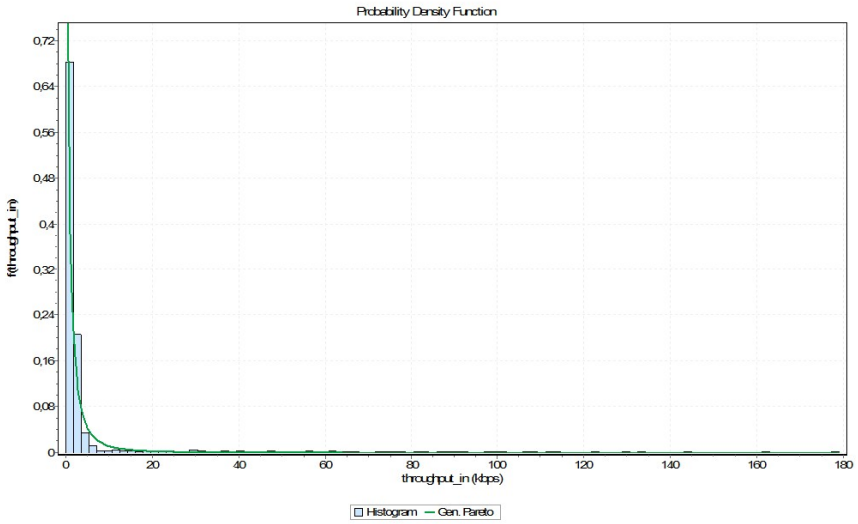


Figure 30. Probability density function chart for 8 users upstream throughput model.

The following parameters are applicable for the upstream throughput model in MSAN-based backhaul:

Parameter	Number of users	K-S Stat	ξ	μ	σ
Generalized Pareto Distribution					
Value	8 users	0.24566	0.76004	-0.15266	0.95793
Value	6 users	0.21423	0.76051	-0.13851	1.1614
Value	4 users	0.46801	0.85334	-0.0616	0.17393

Table 16. Parameters for upstream throughput model in MSAN-based backhaul.

Also from the related experiment, Generalized Extreme Value Distribution was known to be relatively closer to the data. If *t* is defined as throughput for downstream traffic, then:

$$f(t;\mu,\sigma,\xi)=\frac{1}{\sigma}\left[1+\xi\left(\frac{t-\mu}{\sigma}\right)\right]^{(-1/\xi)-1}\exp\left\{-\left[1+\xi\left(\frac{t-\mu}{\sigma}\right)\right]^{-1/\xi}\right\} \quad (11)$$

for $x<\mu+\sigma/\xi$ in the case $\xi > 0$;
for $x<\mu+\sigma/(-\xi)$ in the case $\xi < 0$.

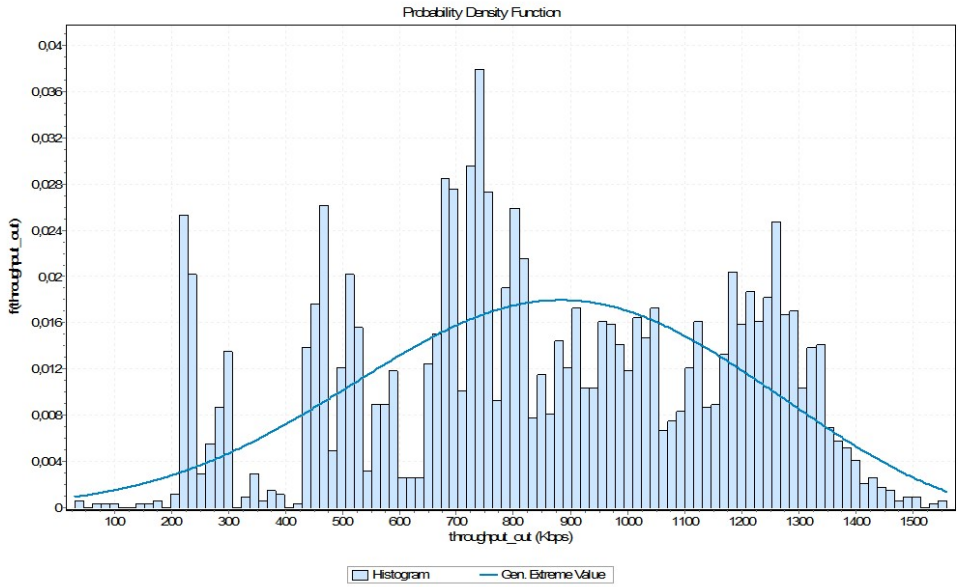


Figure 31. Probability density function chart for 8 users downstream throughput model.

The parameters for the above downstream throughput model in MSAN-based backhaul are as follows:

Parameter	Number of users	K-S Stat	ξ	μ	σ
Generalized Extreme Value Distribution					
Value	8 users	0.05308	-0.35212	747.8	335.92
Value	6 users	0.08877	-0.61395	989.51	318.63
Value	4 users	0.26165	-0.05221	233.32	29.925

Table 17. Parameters for downstream throughput model in MSAN-based backhaul.

3.1.8.2 *GPON-based Backhaul Network*

The backhaul throughput for 1-hour test in GPON-based backhaul can be shown in the below graphs. The test revealed that incoming throughput in the network under test was:

- Around 2 to 100 Mbps (incoming to server with 2,500 users).
- Around 100 to 400 Mbps (outgoing from server with 2,500 users).
- Around 2 Mbps (incoming to server with 500 users).
- Around 50 to 150 Mbps (outgoing from server with 500 users).

GPON-based backhaul is applicable for modeling in BS1 and BS2 scenarios.

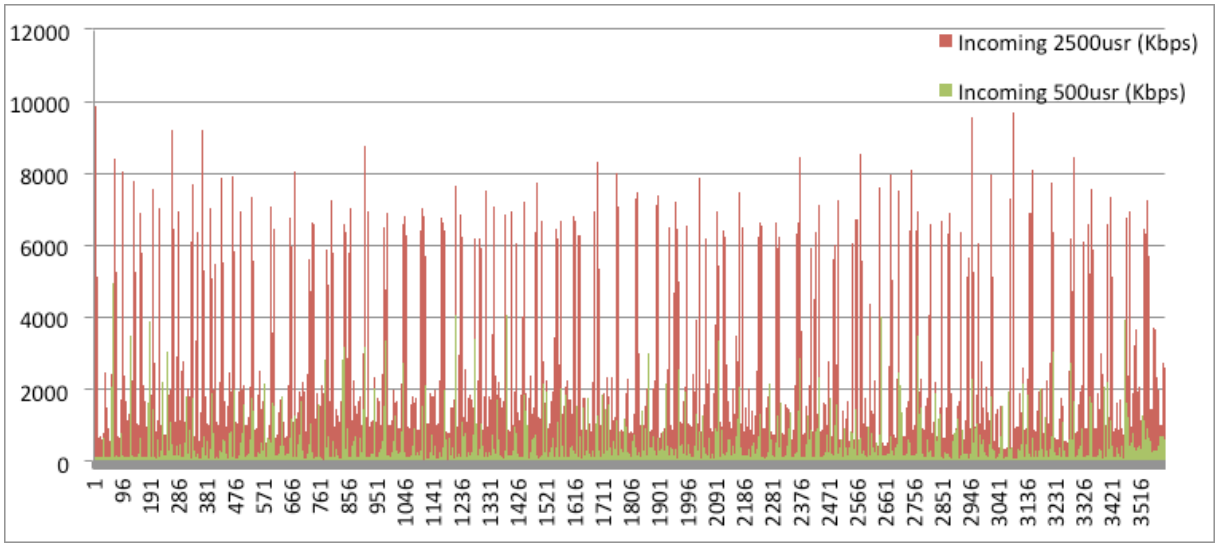


Figure 32. Incoming traffic throughput to server in GPON backhaul (3,600 sec. test).

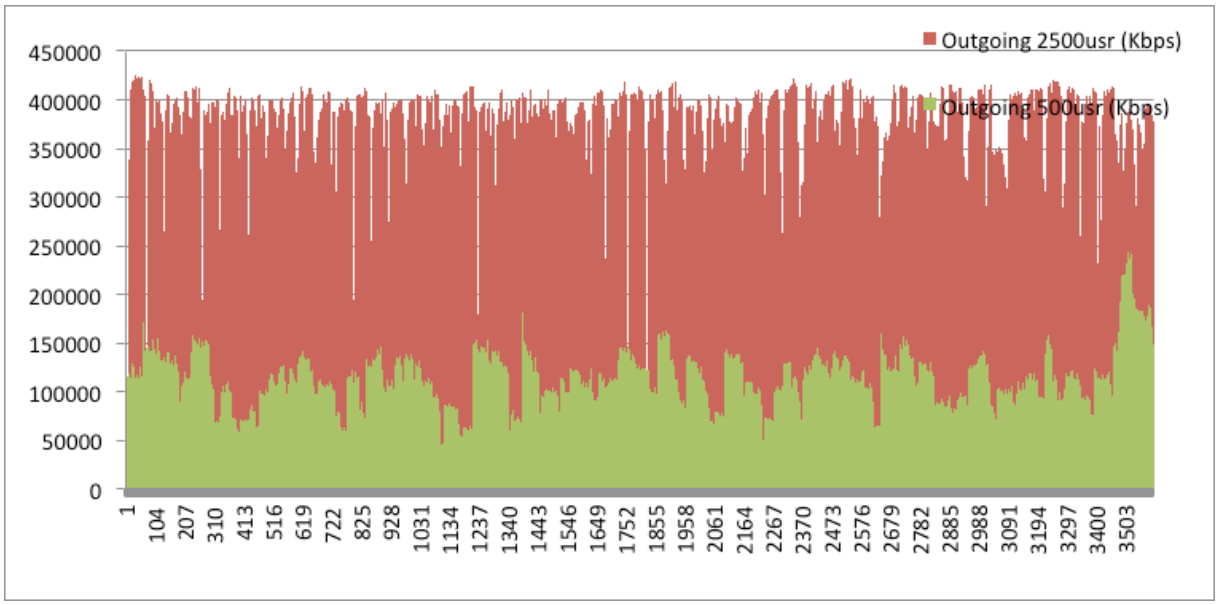


Figure 33. Outgoing traffic throughput from server in GPON backhaul (3,600 sec. test).

3.1.8.2.1 Delay

One hour duration tests with 2,500 and 500 users were conducted to obtain the statistical parameters for delay modeling in GPON-based backhaul. Assuming t_{RTT} is the round trip delay from the client side, this delay is a pseudo-random value with probability density function given by the below Gumbel Maximum Value Type 1 equation:

$$f(t_{RTT}) = \frac{1}{\sigma}\exp(-z-\exp(-z))$$
$$z \equiv \frac{t_{RTT}-u}{\sigma} \quad (12)$$

The following chart for 2,500 users test illustrates delay model in GPON-based backhaul, which also applicable for 500 users test.

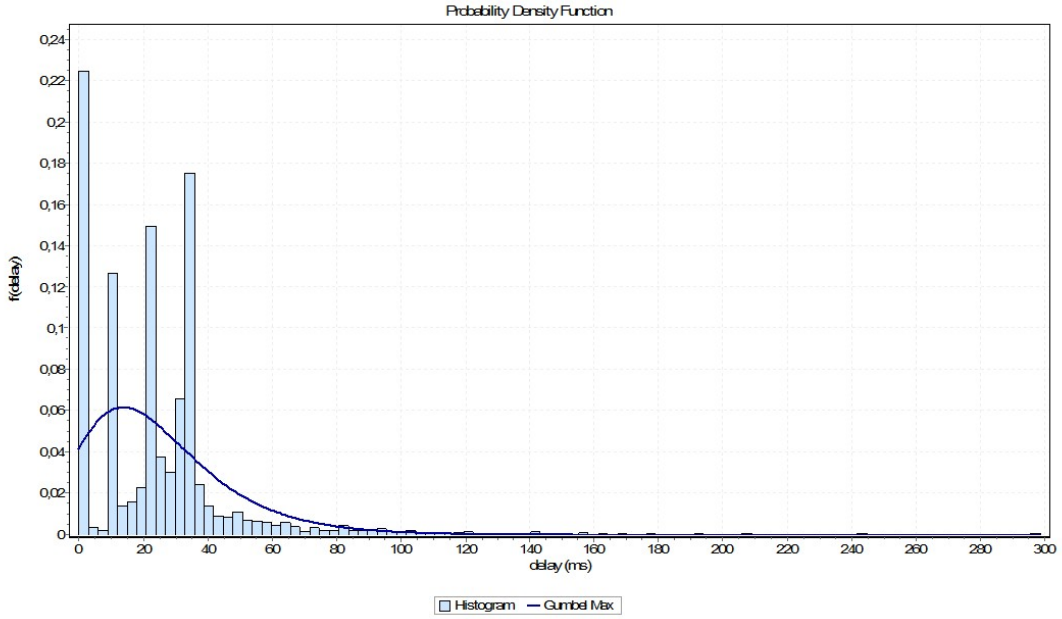


Figure 34. Probability density function chart for 2,500 users delay model.

The test obtained the following parameters for corresponding distribution function.

Parameter	Number of users	K-S Stat	μ	σ
Gumbel Maximum Value Type 1 Distribution				
Value	2,500 users	0.11743	13.826	17.816
Value	500 users	0.26544	1.8477	6.5418

Table 18. Parameters for delay model in GPON-based backhaul.

3.1.8.2.2 Jitter

Jitter model for GPON-based backhaul network is different to the one in MSAN-based backhaul. The tests obtained the following 60-second delay variation or jitter model for 2,500 and 500 users, which follows a Four-Parameter Generalized Gamma Distribution. The equation is as follows, with j_{RTT} denotes 60-second jitter:

$$f(j_{RTT}) = \frac{k(j_{RTT}-\gamma)^{k\alpha-1}}{\beta^{k\alpha}\Gamma(\alpha)}\exp\left(-((j_{RTT}-\gamma)/\beta)^{k}\right) \quad (13)$$

The following chart for 2,500 users test illustrates jitter model in GPON-based backhaul, which also applicable for 500 users test.

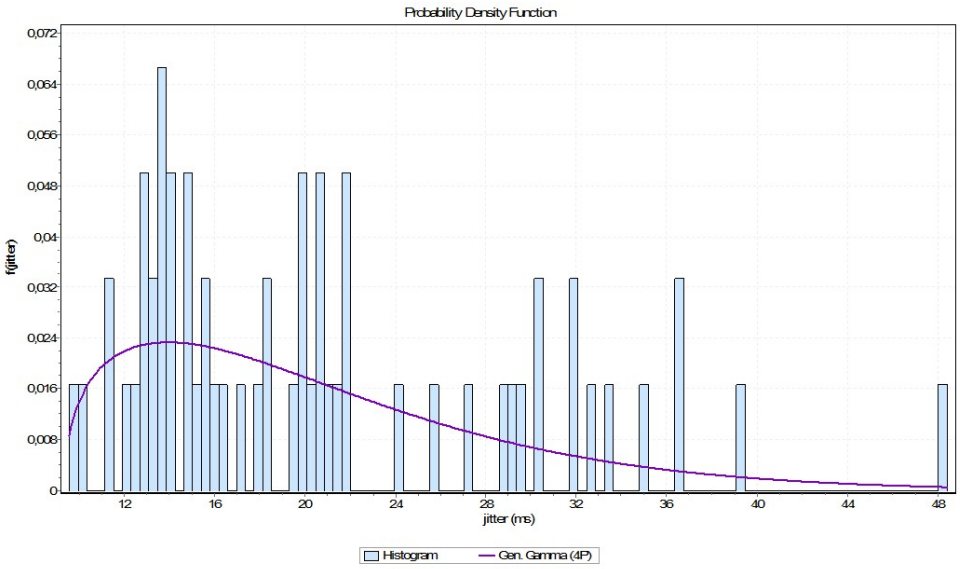


Figure 35. Probability density function chart for 2,500 users jitter model.

The test obtained the following parameters for corresponding distribution function.

Parameter	Number of users	K-S Stat	k	α	β	γ
Four-Parameter Generalized Gamma Distribution						
Value	2,500 users	0.07368	1.2895	1.0744	11.509	9.4338
Value	500 users	0.09054	0.79297	13.268	0.36302	-3.0223

Table 19. Parameters for jitter model in GPON-based backhaul.

3.1.8.2.3 Packet Loss

Gumbel Maximum Value Type 1 Distribution was also relatively closer to the packet loss model for GPON based backhaul network (both downstream and upstream). The symbol $l_{svr\text{-}cli}$ is used to show packet loss parameter from server to client:

$$f(l_{svr-cli}) = \frac{1}{\sigma}\exp\big(-z-\exp(-z)\big)$$
$$z \equiv \frac{l_{svr-cli}-\mu}{\sigma} \qquad (14)$$

The following graph illustrates the probability density function (PDF) for downstream packet loss (ie from server to client) in GPON-based backhaul. The symbol 0.04, 0.08, 0.12 and so forth in the x-axis below denotes 4%, 8%, 12% packet loss respectively.

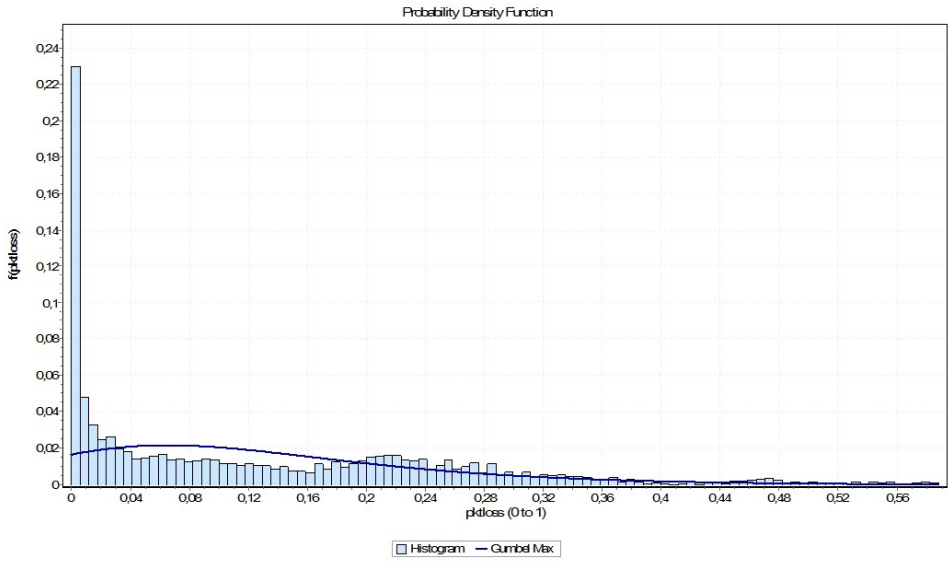


Figure 36. Probability density function chart for 500 users down. packet loss model.

The following parameters are applicable for both packet loss model (Table 20 for downstream direction and Table 21 for upstream direction):

Parameter	Number of users	K-S Stat	μ	σ
Gumbel Maximum Value Type 1 Distribution				
Value	2,500 users	0.41364	0.53649	0.18214
Value	500 users	0.14506	0.06541	0.09943

Table 20. Parameters for downstream packet loss model in GPON-based backhaul.

Parameter	Number of users	K-S Stat	μ	σ
Gumbel Maximum Value Type 1 Distribution				
Value	2,500 users	0.40231	-0.01195	0.12754
Value	500 users	0.34278	0.00443	0.06484

Table 21. Parameters for upstream packet loss model in GPON-based backhaul.

3.1.8.2.4 Throughput Model

Traffic from client was randomly generated by traffic generator in TELK premise during one hour period and referred to as upstream direction. Traffic from server is referred to as downstream direction.

Similar to the one in MSAN based backhaul, the below traffic (upstream to server and downstream from server) was measured in the server side.

From the experiment, Generalized Extreme Value Distribution with the below function was found to be relatively closer to the data. If t is defined as throughput for upstream traffic in GPON-based backhaul, then:

$$f(t;\mu,\sigma,\xi)=\frac{1}{\sigma}\left[1+\xi\left(\frac{t-\mu}{\sigma}\right)\right]^{(-1/\xi)-1}\exp\left\{-\left[1+\xi\left(\frac{t-\mu}{\sigma}\right)\right]^{-1/\xi}\right\} \quad (15)$$

for $x<\mu+\sigma/\xi$ in the case $\xi > 0$;
for $x<\mu+\sigma/(-\xi)$ in the case $\xi < 0$.

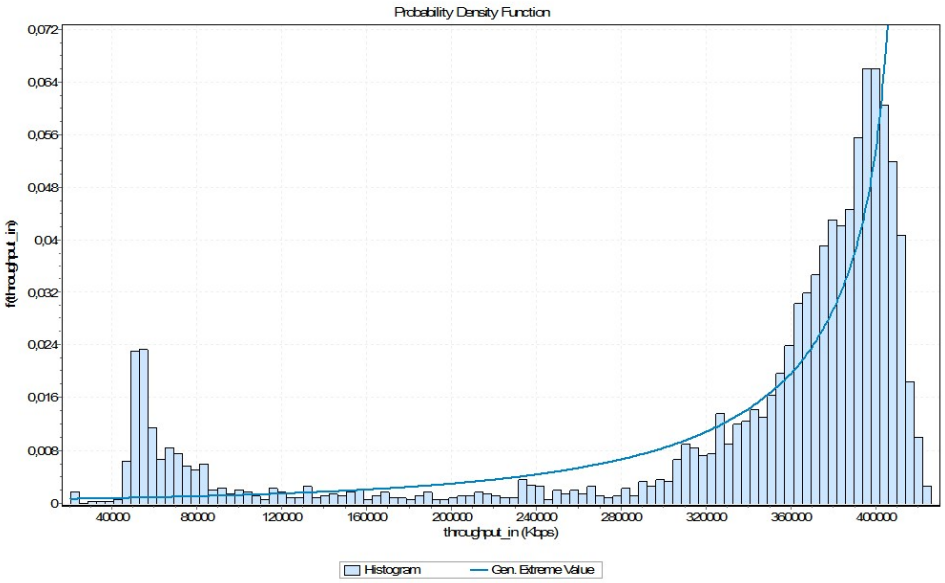


Figure 37. Probability density function chart for 2500 users upstream throughput model.

The following parameters are applicable for the upstream throughput model in GPON-based backhaul:

Parameter	Number of users	K-S Stat	ξ	μ	σ
Generalized Extreme Value Distribution					
Value	2,500 users	0.06822	-1.438	3.4890E+5	90563.0
Value	500 users	0.06195	0.62448	111.34	60.657

Table 22. Parameters for upstream throughput model in GPON-based backhaul.

For downstream traffic, Generalized Extreme Value Distribution was also relatively closer to the data distribution.

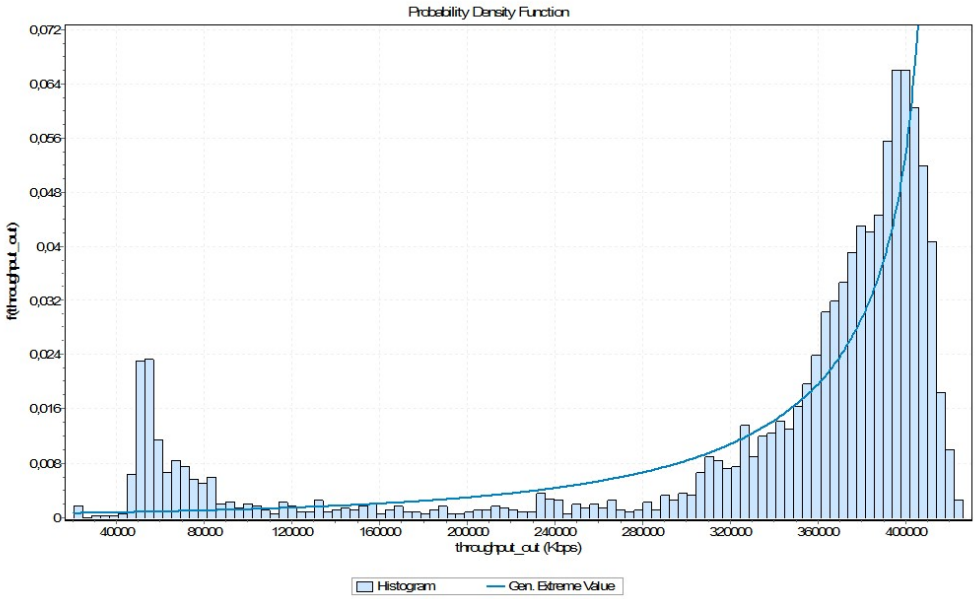


Figure 38. Probability density function chart for 8 users downstream throughput model.

The parameters for the above downstream throughput model in GPON-based backhaul are as follows:

Parameter	Number of users	K-S Stat	ξ	μ	σ
Generalized Extreme Value Distribution					
Value	2,500 users	0.06822	-1.438	3.4890E+5	90563.0
Value	500 users	0.03995	-0.18887	96305.0	25753.0

Table 23. Parameters for downstream throughput model in GPON-based backhaul.

3.2 *PHY layer model*

WP3 addressed the design of transmission strategies for optimizing the **trade-off between the latency and the energy** spent in the **communication**. The latency refers to the time required to transmit a block of bits of a given size. The energy, on the other hand, includes not only the radiated energy, but also the energy spent in the base band processing, coding/decoding and RF stages of the transmitter and receiver. This section provides abstraction models for the PHY layer based on the work done within WP3 and presented in [Munoz13b]. Given the channel, the UE parameters and the number of antennas, any communication strategy will provide a different curve that relates the energy spent by the UE to send or receive a block of bits and the time required for it. All the information regarding PHY techniques and consumption of the UE is summarized in these curves that will be the interface between the optimization techniques in WP5 and the PHY techniques in WP3.

The following figure shows the relationship between the input parameters of the communication abstraction models (propagation channel, UE parameters, and antenna configuration), the physical layer techniques derived in WP3, the user mobile terminal consumption models described in WP2, and the curves that model the communication stage. In this sense, four different curves are considered. These figures provide:

- the energy spent in the uplink (UL), e_{UL}, versus the communication time allocated for the UL, t_{UL}, and the number of bits to be sent in the UL, s_{UL} (this is represented by $e_{UL}(t_{UL}, s_{UL})$);

- the time required in the uplink (UL), t_{UL}, versus the budget for the energy consumption in the UL, e_{UL}, and the number of bits to be sent in the UL, s_{UL} (this is represented by $t_{UL}(e_{UL}, s_{UL})$);
- the energy spent in the downlink (DL), e_{DL}, versus the communication time allocated for the DL, t_{DL}, and the number of bits to be sent in the DL, s_{DL} (this is represented by $e_{DL}(t_{DL}, s_{DL})$);
- the time required in the downlink (DL), t_{DL}, versus the budget for the energy consumption in the DL, e_{DL}, and the number of bits to be sent in the DL, s_{DL} (this is represented by $t_{DL}(e_{DL}, s_{DL})$).

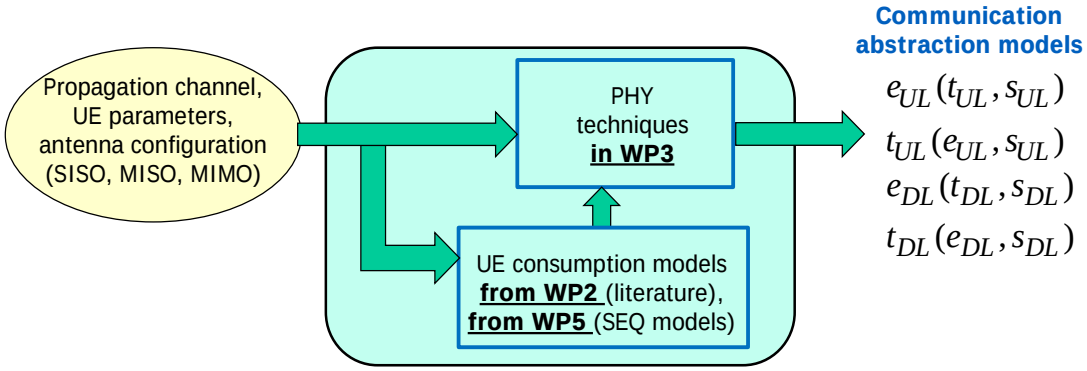


Figure 39. Inputs and outputs for communication abstraction models.

Note that, from the point of view of the tradeoff between the latency and the energy, knowing the curve $e_{UL}(t_{UL}, s_{UL})$ is equivalent to know $t_{UL}(e_{UL}, s_{UL})$. The same thing happens in the case of the DL transmission. Therefore, we focus in the following on the characterization of $e_{UL}(t_{UL}, s_{UL})$ and $e_{DL}(t_{DL}, s_{DL})$.

Once the relationship between the energy, time, and number of communicated bits has been characterized, the communication stage can be treated as a black box within the general master allocation problem that includes communication and computation.

Note that any communication strategy will provide a different curve. In WP5, we will focus primarily on those strategies developed in WP3 to optimize the trade-off between the latency and the energy spent in the communication. Using such techniques, for a given communication time, the minimum communication energy is obtained or vice versa.

3.2.1 Energy-latency tradeoff in the SISO case

As an example, in the SISO case, the curve that relates the energy, time and packet size for the communication in the uplink and downlink are given by eq. (16) and (17) respectively:

$$e_{UL}(t_{UL}, s_{UL}) \approx k_{tx,1} t_{UL} + k_{tx,2} t_{UL} \frac{2^{\frac{s_{UL}}{W_{UL} \cdot t_{UL}}} - 1}{\gamma_{UL}} \quad (16)$$

$$e_{DL}(t_{DL}, s_{DL}) \approx k_{rx,1} t_{DL} + k_{rx,2} s_{DL} \quad (17)$$

where $k_{tx,1}$, $k_{tx,2}$, $k_{rx,1}$ and $k_{rx,2}$ are constants (the numerical values of these constants can be set to adjust, for example, the estimated power consumption with the measurements provided in [Jensen12] or with the measurements described in [TROPIC_D311]. γ_{UL} represents the equivalent channel power gain in linear units for the UL transmission. When $k_{tx,1}$ is set to 0, the constant baseline energy which is consumed just for having the transmitter chain switched on is not considered. When $k_{rx,1}$ is set to 0, the constant baseline energy which is consumed just for having the receiver chain switched on is not considered.

Figure 40 and Figure 41 show the estimated energy consumption according to eq. (16) versus the time allocated for the UL transmission when transmitting a given number of bits. Figure 40 takes $k_{tx,1} = 0$. Therefore, only the part coming from the radiated energy is considered in the estimation of the energy consumption. On the other hand, Figure 41 takes $k_{tx,1} = 0.4$ W.

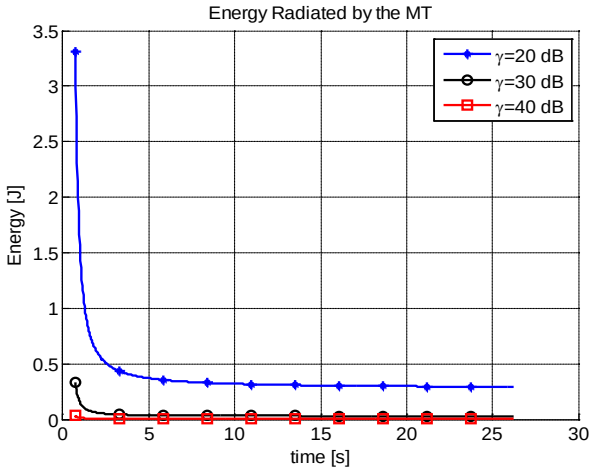


Figure 40. Energy consumption estimation according to eq. (16) considering the following parameters $k_{tx,1} = 0,\ k_{tx,2} = 18,\ W_{UL} = 10MHz,\ s_{UL} = 5Mbyte$.

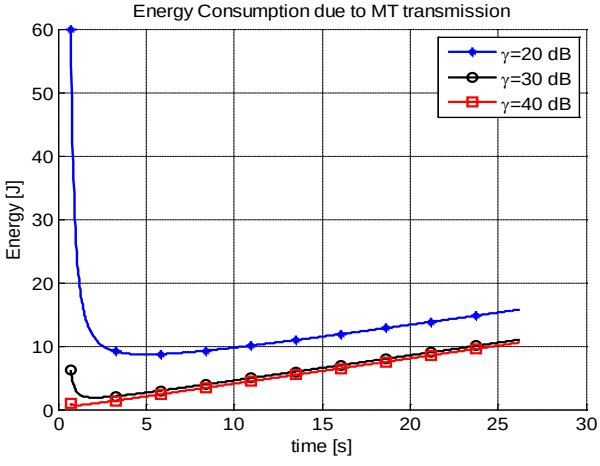


Figure 41. Energy consumption estimation according to eq. (16) considering the following parameters $k_{tx,1} = 0.4,\ k_{tx,2} = 18,\ W_{UL} = 10MHz,\ s_{UL} = 5Mbyte$.

It is important to remark that, when taking the offloading decision (that will be considered later in this deliverable), the optimum transmission time for the UL will not be necessarily be the one corresponding to the minimum UL communication energy since there will be other constraints in the global problem that may potentially prevent it.

3.2.2 Energy-latency tradeoff in the MIMO case

The expressions provided in the previous subsection for the SISO case can be generalized to a setup where both the UE and the serving SC have multiple antennas configuring a MIMO channel. According to this, in this subsection we present the energy-latency tradeoffs for this general setup. The complete derivations for these expressions with the corresponding proofs can be found in deliverable D311 and in [Munoz13b] and are not reproduced here for the sake of clarity and brevity.

Uplink (UL) transmission

Given the UL transmission time, t_{UL}, and the number of bits to be transmitted, s_{UL}, it can be proved that the optimum UL transmission strategy in terms of energy consumption is to transmit over the eigenmodes of the equivalent UL channel matrix **H**, i.e., using a set of beam vectors corresponding to the eigenvectors of the matrix $\mathbf{H}^H\mathbf{H}$ with a proper power allocation among them. According to this result, the final expression of the energy-latency tradeoff in UL is given by

$$e_{UL}\left(t_{UL}, s_{UL}\right) \approx k_{tx,1}t_{UL} + k_{tx,2}t_{UL}\sum_{i=1}^{K(t_{UL},s_{UL})}\left(c(t_{UL},s_{UL}) - \frac{1}{\lambda_i}\right)$$
$$c(t_{UL},s_{UL}) = \frac{2^{\frac{s_{UL}}{W_{UL}t_{UL}K(t_{UL},s_{UL})}}}{\left(\prod_{k=1}^{K(t_{UL},s_{UL})}\lambda_k\right)^{\frac{1}{K(t_{UL},s_{UL})}}} \tag{18}$$

where $K(t_{UL},s_{UL}) \le rank\left(\mathbf{H}^H\mathbf{H}\right)$ is the number of active eigenmodes of the channel through which the transmission is carried out. Such number of active eigenmodes is calculated as the discrete value *K* satisfying the following conditions:

$$\frac{2^{\frac{s_{UL}}{W_{UL}t_{UL}K}}}{\left(\prod_{k=1}^{K}\lambda_k\right)^{\frac{1}{K}}} > \frac{1}{\lambda_K} \quad \text{and} \quad \frac{2^{\frac{s_{UL}}{W_{UL}t_{UL}K}}}{\left(\prod_{k=1}^{K}\lambda_k\right)^{\frac{1}{K}}} \le \frac{1}{\lambda_{K+1}}, \qquad 1 \le K < rank\left(\mathbf{H}^H\mathbf{H}\right)$$
$$\frac{2^{\frac{s_{UL}}{W_{UL}t_{UL}K}}}{\left(\prod_{k=1}^{K}\lambda_k\right)^{\frac{1}{K}}} > \frac{1}{\lambda_{rank\left(\mathbf{H}^H\mathbf{H}\right)}}, \qquad K = rank\left(\mathbf{H}^H\mathbf{H}\right) \tag{19}$$

It is important to remark that both the number of active eigenmodes $K(t_{UL},s_{UL})$ and the water-level $c(t_{UL},s_{UL})$ in (18) depend on the UL transmission time and the number of bits through the UL transmission rate $r_{UL} = \frac{s_{UL}}{t_{UL}}$.

As an illustrative example, the following figure represents the number of active eigenmodes $K(t_{UL}, s_{UL})$ as a function of the UL transmission rate normalized by the bandwidth, i.e., as a function of $\frac{s_{UL}}{t_{UL} W_{UL}}$, for a single realization of a random 4x4 channel. As it can be observed, as the UL rate increases, the number of active modes also increases.

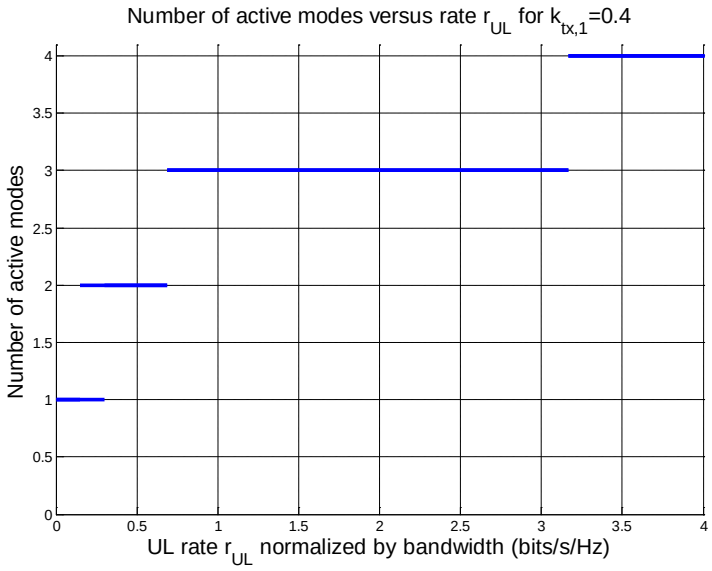


Figure 42. Number of active eigenmodes $K(t_{UL}, s_{UL})$ vs the UL rate normalized by the bandwidth $\frac{s_{UL}}{t_{UL} W_{UL}}$ for a single realization of a 4x4 random MIMO channel.

Downlink (DL) transmission

In the DL case, and according to (17), for a given number of bits to be transmitted, the energy spent by the UE when receiving is minimized when the required transmission time t_{DL} is also minimized or, in other words, when the DL transmission rate, defined as $r_{DL} = \frac{s_{DL}}{t_{DL}}$, is maximum. Such transmission rate is upper-bounded by the channel capacity $R_{DL}^{\max}$, i.e., $r_{DL} \le R_{DL}^{\max}$, which in turn depends on the maximum available transmission power at the serving SC $P_{tx,SC}$. The calculation of $R_{DL}^{\max}$ is found as the optimum value of the cost function within the following optimization problem:

$$\begin{aligned} &\text{minimize} && W_{DL} \log_2 \left| \mathbf{I} + \mathbf{HQH}^H \right| \\ &\text{s.t.} && \text{C1: } Tr(\mathbf{Q}) \le P_{tx,SC}, \\ & && \text{C2: } \mathbf{Q} \ge \mathbf{0}. \end{aligned} \tag{20}$$

where **H** represents the equivalent response of the DL MIMO channel and **Q** is the transmit power covariance matrix at the transmitting serving SC (that should be positive semi definite according to constraint C2). The optimum solution to this problem is well known and consists in transmitting through the eigenmodes of the DL MIMO channel matrix with a power allocation policy based on the water-filling approach. Summarizing, the DL transmission should always be done at the maximum rate denoted by $R_{DL}^{\max}$ and that depends solely on the channel state and the transmission power at the

serving SC. In this sense, the tradeoff between energy and time in the DL is simpler than the one describing the UL transmission.

3.2.3 PHY abstraction procedure considered in TROPIC

In TROPIC, two ways of abstraction modeling of PHY are considered. The first one is a simplified model using simple mapping between SINR and MCS. For the cases when more accurate modelling is required and for WP6 purposes, the modelling defined by *LENA-ns3* simulator is selected.

3.2.3.1 *Less complex PHY abstraction for WP5*

When all the subcarriers allocated to one receiver are modulated using the same modulation and coding scheme (MCS) as in LTE, a compression function to map the instantaneous values of SNIRs to one single MCS value is required. Despite there are different possibilities, the EESM (Exponential Effective SINR Mapping) has shown to yield an accurate estimation of the AWGN-equivalent SINR (usually referred to as 'effective SINR') for frequency selective channels. The EESM method estimates the effective SINR using the following formula:

$$SINR_{eff} = EESM(\gamma, \beta) = -\beta \ln\left(\frac{1}{M}\sum_{i=1}^{M} e^{\frac{-\lambda_i}{\beta}}\right) \quad (21)$$

where the λ_i's are the per sub-carrier SINR values, and β is the parameter to be determined for each MCS level. This value is used to match the actual BLER and the predicted BLER from the effective SINR in the AWGN channel. For the simulation results the MCSs available in LTE-A (15 values ranging from 0.1523 to 5.5547), along with the effective SINR and β values provided in [LTE simulator], obtained through extensive simulations using the LTE codes can be considered. For each MCS value, the effective SINR is the effective SINR required for a BLER less than 10% when using the MCS value. As shown in Figure 43, the relationship between the MCS and the required effective SINR (red points) can be approximated by the following empirical SINR-to-MCS mapping function (solid blue line), with k=1.2213:

$$MCS(SINR_{eff}) \cong k \cdot ln(1 + SINR_{eff}) \quad (22)$$

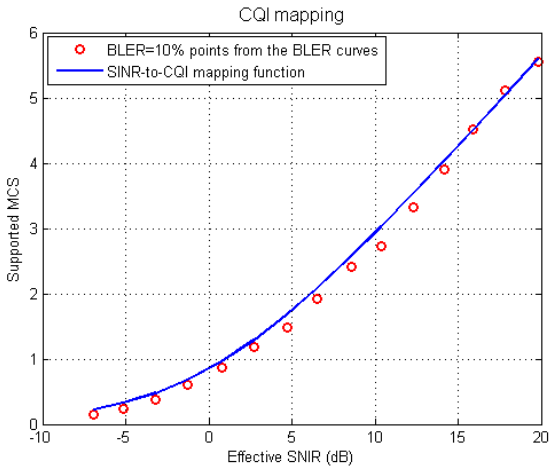


Figure 43. MCS mapping.

CQI	bits/symbol	beta	SINR threshold
1	0,1523	1	-6,934
2	0,2344	1,44	-5,147
3	0,377	1,4	-3,18
4	0,6016	1,48	-1,254
5	0,877	1,5	0,761
6	1,1758	1,62	2,7
7	1,4766	3,1	4,697
8	1,9141	4,32	6,528
9	2,4063	5,37	8,576
10	2,7305	7,71	10,37
11	3,3223	15,5	12,3
12	3,9023	19,6	14,18
13	4,5234	24,7	15,89
14	5,1152	27,6	17,82
15	5,5547	28	19,83

Table 24. SINR thresholds for MCS mapping.

3.2.3.2 *Abstraction modelling for WP6*

The PHY abstraction model adopted for the simulations in TROPIC is the model already implemented in the *LENA-ns3* simulator, selected for the WP6 of the project. It should be noted that this model will be used for the simulations with the standard LTE communication. When other more "complex" communication techniques are involved, the suitable abstraction models will be introduced.

The link abstraction model is based solely on the knowledge of the SINR and the modulation and coding scheme. The model combines Mutual Information-based multi-carrier compression metrics with Link-Level performance curves matching, to obtain lookup tables that express the dependency of the Block Error Rate on the SINR values and on the modulation and coding scheme being used.

The data at the MAC layer of the LTE protocol stack (right above the LTE PHY) is arranged in Transport Blocks (TB), whose size depends on the specific configuration of the underlying PHY. TBs are split into a number of CBs which are independently encoded by the turbo encoder at the PHY layer. Each CB is then encoded and transmitted over the channel exploiting the number of RBs allotted to the user.

Basically, the abstraction model produces the BLER of the Transport Blocks (system-level) from the BLER of the Code Blocks (link-level) for several CB sizes and MCS values.
Link-level simulations are used to obtain the PHY layer performance in terms of BLER as function of SINR over AWGN channels, accounting for the configuration of the PHY layer turbo encoder in terms of the Code Block (CB) length and the selected MCS. The CB size highly impacts the actual BLER performance for a given MCS.

Given the following list of symbols, below the steps of the abstraction procedure.

- N the number of RBs allocated to a UE to transmit a CB over the channel;
- $SNIR_i$ the SNIR of the RB_i and $SNIR_N$ the n-vector of SNIRs of the $\boldsymbol{N}$ RBs;
- C the number of CBs contained in a TB;
- *CBLER* and *TBLER* the BLER of a CB and a TB respectively;

Effective SINR Mapping

Using the *Effective SINR Mapping* (ESM) technique, the instantaneous vector $SNIR_N$, containing the SNIR of the N RBs used to transmit a CB, is mapped onto a single scalar value as follows:

$$eSINR = \alpha_1 I^{-1}\left(\frac{1}{N}\sum_{n=1}^{N} I\left(\frac{SINR_n}{\alpha_2}\right)\right) \quad (23)$$

where *I(.)* represents the *information measure function*, whereas α_1 and α_2 are two scaling parameters that are tuned as a function of the selected MCS.
Based on literature results, the *Mutual Information* (MI) metric (MIESM) is adopted as mapping approach, in particular the *MIB*, defined as the mutual information between the bit input belonging to a specific constellation (MCS), and the corresponding log-likelihood ratio (LLR) output at the receiver.

The *Mean Mutual Information per coded Bit* (MMIB) is expressed as:

$$MMIB = \frac{1}{N}\sum_{n=1}^{N} I_m(SINR_n) \quad (24)$$

with $I_m(.)$ obtained as a mixture of Bessel function, using a different combination for each possible modulation scheme *m* (QPSK, 16-QAM and 64-QAM).

This MMIB corresponds to an equivalent SINR for the transmission of a CB over the allotted RBs and hence it is a time-varying compressed representation of the channel quality as perceived by any given user at any given time.

BLER prediction
To obtain the residual error rate of CB, its MMIB have to be mapped into CBLER. The approximation based on the *Gaussian cumulative model* is used to produce the estimated BLER curves as a function of MMIB. This method provides an analytic and parameterized form of the curves obtained with the link-level simulations.
The estimated BLER curves as a function of MMIB are parameterized as follows:

$$CBLER_i = \frac{1}{2}\left[1 - erf\left(\frac{x - b_{S,M}}{\sqrt{2}c_{S,M}}\right)\right] \quad (25)$$

where $b_{S,M}$ and $c_{S,M}$ are the mean and the standard deviation of the Gaussian cumulative distribution, respectively, and *x* is the MMIB associated with CB *i*. *S* is the code block size and *M* is the MCS, which dictates the actual transmission rate.

In order to reduce the computational complexity and the memory space, only a subset of CB sizes is considered, with unbalanced quantization of the size, because the BLER performance is critical for small code blocks, in order to get more accuracy in such critical zone (small code blocks).
The CB sizes is limited to some representative values (i.e., 40, 140, 160, 256, 512, 1024, 2048, 4032, 6144), while for the others the worst approximation of the real value is used (i.e., the smaller CB size value available respect to the real one).

The error rate at TB level (overall TBLER) is then computed as a product of the $CBLER_i$ of all the *C* Code Blocks belonging to a Transport Block:

$$TBLER_i = 1 - \prod_{i=1}^{C}(1 - CBLER_i) \quad (26)$$

The following figure shows the set of curves corresponding to several block sizes for a single MCS value, reporting both the off-line simulated curves and the estimated ones.

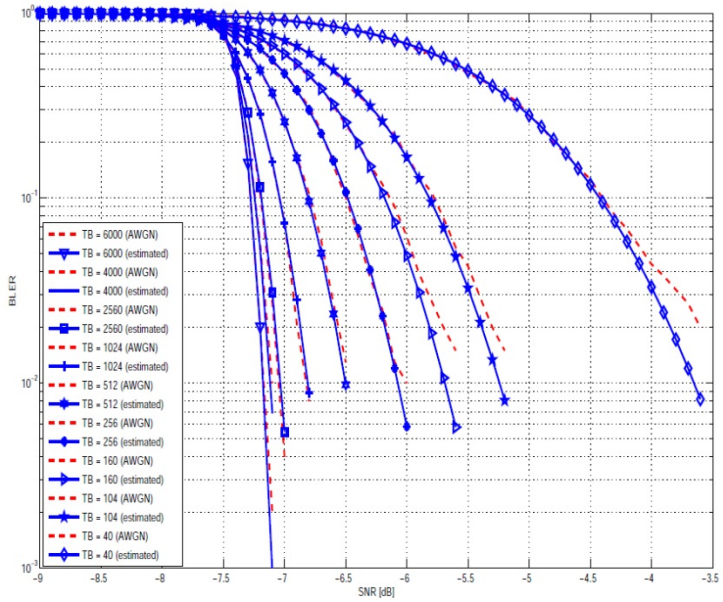


Figure 44. BLER vs SNR example curve for MCS 1 (taken over from [ns3]).

Link adaptation

The AMC mechanism is based on the *Error Model* approach. It relies on the link abstraction model, selecting the MCS that better complies with a given transport block error rate (TBLER) for the connection.

The process starts from the most aggressive transmission scheme (MCS 29), and evaluates the TBLER using the equations described above, taking as input the vector of SINRs for the selected user. If the TBLER is larger than or equal to the target BLER defined by 3GPP (i.e., 0.1), the algorithm keeps on searching for a better (less aggressive) MCS; otherwise, the procedure stops.

The *Error Model* approach outperforms the *Spectral efficiency*-based approach commonly frequently used in the LTE simulators; at all SNR levels the *Spectral efficiency* method tends to be too conservative. However also the latter method is implemented and can be selected.

Remarks

It important to mention that:

- the simulator make use of the *TU Wien LTE Link-level Simulator* to obtain the PHY error model used to model the effect of the physical layer at system-level.

- the simulators take into account the HARQ mechanism of LTE in the link-to-system abstraction. HARQ is part of the LTE MAC layer and provides retransmission capabilities of erroneously-received TBs, aimed at improving link reliability. LENA-ns3 simulator has integrated new sets of curves to the standard one used for the original MCS in order to consider the effect of HARQ retransmissions.

3.3 *Application model*

3.3.1 Task graphs

Given the complexity and diversity of modern applications which span different software architectures (monolithic, centralized, distributed, client server, peer to peer, thin client, thick client, SOA, web applications, app, etc.), run on device with different hardware characteristics (single core, multiple core, no cache, unified cache, shared cache, I/O subsystems, *etc*.) and operating systems features (dedicated and/or shared resources, RTOS, virtualized solutions, work conserving, non work conserving, preemptable, non preemtable scheduling, *etc*.), it is not easy to derive a model of an

application behavior/performance. In general, an application model should be able to support a wide range of analysis techniques from abstract mathematical model to simulation while being concise and efficient to capture the essential features of the system under study and its dependence on the key features/parameters without having to deal with low level application/system details.

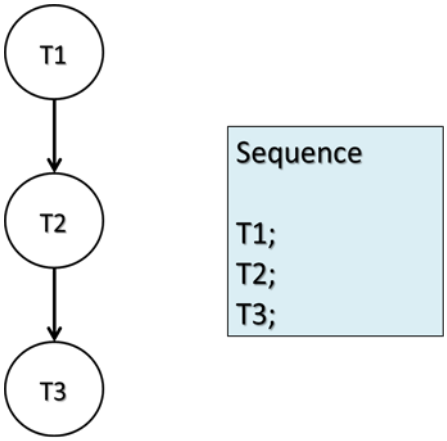


Figure 45. Task sequence.

In TROPIC we will model an application with a *task graph*, which is the most widely known application model. In this model, a *task* is the unit of computation: at the coarsest level, a task can correspond to an entire application, while at the finest level it can correspond to a function, e.g., an image compression, or even a simpler operation. In a task graph, tasks are represented by nodes, while directed edges capture the execution order dependencies among tasks: the edge (T_i,T_j) implies that task T_i must be completed for task T_j to be executed. Moreover, as shown below, in a task graph, nodes represent not only tasks but also programming language constructs, *e.g.*, selection and loops.

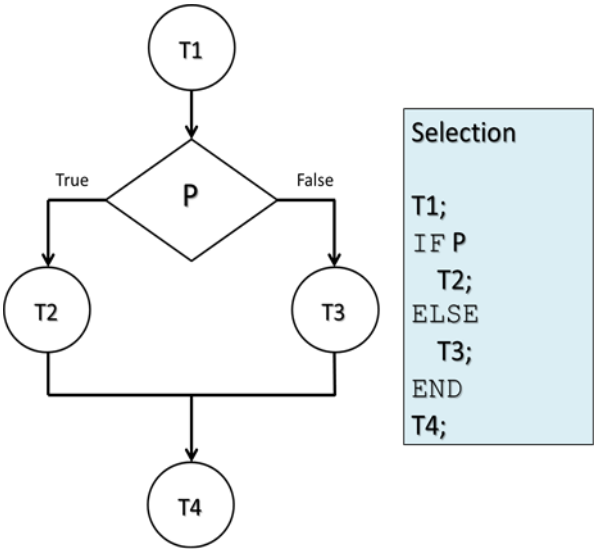


Figure 46. Task Selection (if then else).

In the following, without lack of generality, we will restrict ourselves to structured task graph, i.e., tasks graphs which correspond to structured applications, i.e., applications which are written by combining instructions, tasks, modules only through sequencing (Figure 45), selection (Figure 46) and iteration/loops (Figure 47). The Böhm-Jacopini Theorem (the structured program theorem) ensures that sequencing, selection, and iteration are sufficient to express any computable function. In addition,

we will also consider the parallel construct to cater for the concurrent execution of tasks and/or task partitioning (Figure 48).

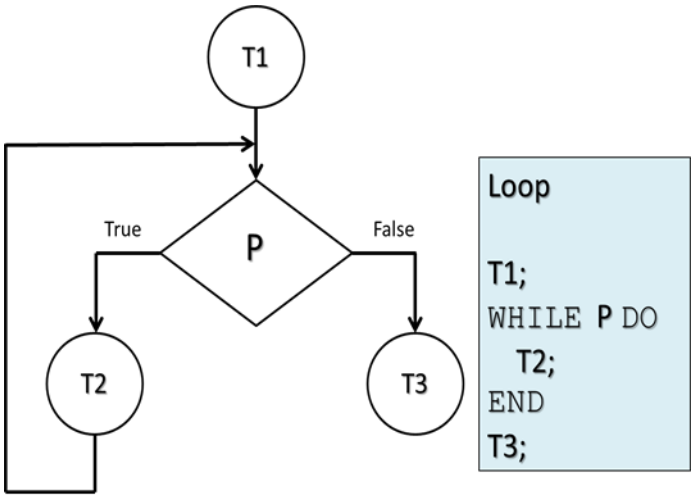


Figure 47. Task Iteration (loop).

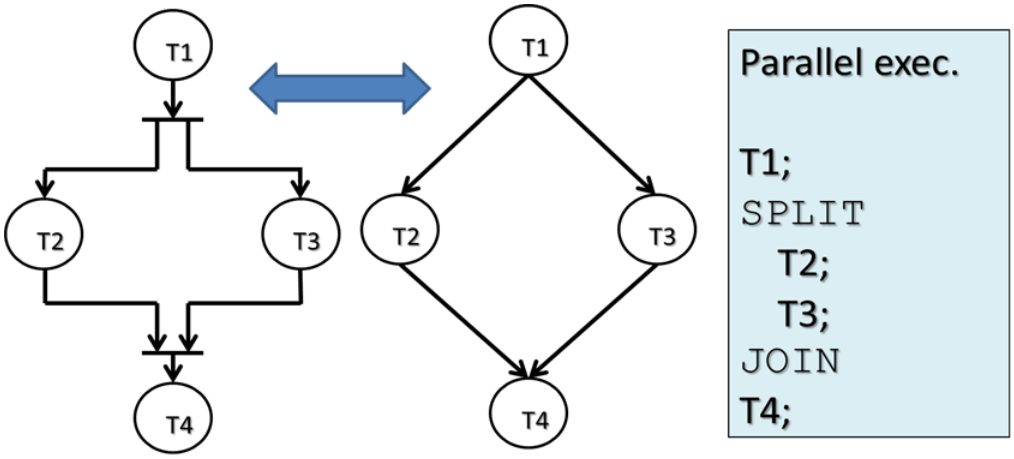


Figure 48. Task parallel execution: standard representation (left) and alternative (right).

3.3.1.1 *Relationship between a task, task graphs and threads*

As mentioned above, a task is an abstract unit of computation ranging from a single instruction to an entire application whereas the task graph represents the entire application. It is important to stress the relationship between a task, which basically represents a portion of code to be executed, a task graph, which related the order of execution, i.e, their precedence order, of all the different portions of code and how an actual system executed them.

In current operating systems, and hence in today virtual machines, applications are executed in the simplest case as a single thread or, in case of parallelized execution, as multiple concurrent threads; often only some part of the execution is actually parallelized and hence the application is run as a single threaded application except for some portion of it which is executed in parallel with multiple thread.

In case of a single threaded application, a single thread executes, following the order dictated by the task graph, the different portions of the application code. In case of a multiple threaded application, we can have, in the most general case, multiple task graphs run in parallel. More often, though, as mentioned above only a portion of the application is parallelized. In these cases, the application structure follows the structure depicted in Figure 33, where tasks T2 and T3, which are executed in parallel, are associated to two threads. Observe that, depending on how the application is actually written, which might depend on the programmer style, we can have different way to associate the task graph execution to actual threads. Considering again the simple example of Figure 33, we can have:

1. a single thread first executes the code associated to T1; then, it launches two threads which execute T2 and T3, and waits for them to run and complete. When the two threads are done, the original thread executes the code associated to T4;
2. a single thread executes the code associated to T1; when the code associated to T1 is completed, it launches another thread to execute T3 (T2) while it executes T2 (T3). When the original thread terminates the execution of T2 (T3), it waits the other thread to complete T3 (T2) and then it executes T4.
3. The application is arranged as a single master thread which in order: 1) launches a child thread to execute T1 and waits for it to terminate; 2) when T1 is completed, it then launches, two child threads, one executing T2 and the other T3 and waits for the two of them to complete; 4) it launches another thread to execute T4.
4. finally, the application can be executed as a single threaded application which executes T1, then T2 and T3 in any order; when it has completed the execution of both T2 and T3, it then executes T4.

Hence, in general we have no unique mapping between tasks and threads since the first represent a passive entity, code to be executed, while the latter represent an *active* entity, which executes the code itself. Nevertheless, it is important to observe that the execution of the application will take about the same time in 1-3 above (barring the time to launch threads which might make option 2) faster then 1) and 3) which should be negligible) where T2 and T3 are executed in parallel and possibly longer time in case 4) since T2 and T3 are serialized. In the following we will always assume that code/tasks that can be executed in parallel is indeed executed by multiple threads (the latter case should would void the benefit of having parallel code in the application and would represent the serial execution of an otherwise parallelized portion of an application).

3.3.2 **Task Characterization**

Our task characterization is focused on performance evaluation. Hence, we are interested in capturing those parameters which mostly affect performance, i.e., response time and energy consumption. They are sketched in Figure 49, and detailed below.

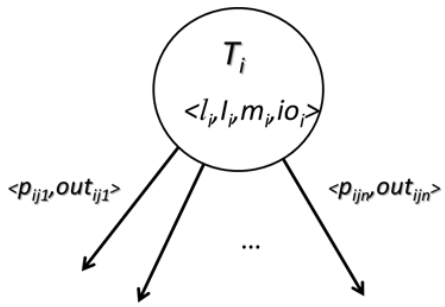


Figure 49. Task parameters.

A task T_i is characterized by the following set of parameters:

- Code length l_i (bytes): the code length affects the time required to upload a task for remote computation;
- Number of instruction to be executed I_i: the number of instruction to be executed, and its statistics (as detailed in later subsections) affects the time required to execute the task on a computing platform;
- Memory requirement m_i (bytes): the amount of memory required by a task/application determines a requirement on the minimum amount of memory a VM need to run the task. Large memory requirement and/or OS memory management via swapping and its impact on task execution will not be dealt with in TROPIC;
- Number of disk I/O operation io_i: the number of I/O operations performed during a task execution. Depending on the type of task I/O operation can have significant impact on the execution time.
- Output size out_{ij} (bytes): the number of bytes that are generated/passed by task T_i and used as input by the subsequent task(s). Since an application can have multiple next tasks (as in the case of the parallel construct), with possibly different data to be exchanged, the output is associated to the departing edges (Figure 46);
- Probability of following a given edge p_{ij}: because of selection and loops, tasks graph have branch points corresponding to multiple edges departing from one node. We associate to the edge (T_i,T_j) the probability p_{ij} of executing T_j after task T_i. Edges corresponding to parallel executions (see Figure 48) are different in that all departing paths are executed in parallel. Implicitly all such edges have probability 1.

3.3.3 **Sample Task Graphs**

In this section we provide sample task graphs from the literature on mobile cloud computing which can be used in the analysis and simulations in WP5 and WP6. In D511, we will provide the task graph model of the prototype application developed in 4A3 and which will be detailed in D43. The sample set will be expanded as new applications will appear in the literature.

The task graphs characterizing mobile computing applications reflect the two dominant approaches in application granularity partitioning which has been summarized in M411. At one extreme, the offloading decision can be taken at the coarsest level, the application level, where except the user interface, which can only be executed on the mobile, the entire application can be offloaded as a whole on a remote server, typically encapsulated in a Virtual Machine. At the other extreme, offloading decisions can be taken at the finest level, at the module, function or even at the instruction level. However, from a practical point of view, too fine grained partitions turn to be rather ineffective because of the involved overhead which, de facto, limits the actual partitioning granularity and/or the number of partitioning options which can be considered in practice. For these reasons, in most of the proposed solutions partitioning is typically at the level of large application components, with two or three main components: the core computation is either entirely offloaded or locally executed (This also eases the computation of the offloading strategy which might negatively impact the overall performance). As a consequence, the tasks graphs are rather simple.

In the following we briefly describe the applications and the associated tasks graph parameters recently appeared in [HaPi13] which are representative of image/audio processing and games applications. These applications have been developed using the Cloudlet approach [Sa09], which adopts the Virtual Machine based offloading which we are considering for TROPIC, and are basically characterized by a GUI running on the mobile node plus a computational intensive component.

Face recognition (FACE)
The Face recognition application detects and attempts to identify faces in image from a populated atabases. The application described in [HaPi13] use Haar Cascades of classifiers, the Eigeinface method for identification and the Open Computer Vision Libraries (OpenCV) [OpenCV].

Speech recognition (SPEECH)
The speech recognition application performs speech to text conversion of spoken English using Hidden Markov Model.

Object recognition (OBJECT)
This application identifies objects and their position in an image using the Scale-invariant feature transform.

Augmented Reality (AR)
This application identifies and labels buildings and landmarks in a photo captured by a mobile device using the OpenCV.

Fluid (FLUID)
This is an interactive fluid dynamics simulation based on phone accelerometer input which renders a liquid sloshing in a container. The computational intensive part is a smoothed particles hydrodynamics simulation generating up to 50 frames per second. This application behavior can be considered representative of interactive real-time games.

The applications components and their interaction is exemplified in Figure 50. Figure 51 and Figure 52 illustrate the corresponding task graphs in the case of a single use, e.g., the user tries to identify a single object in a photo, or repeated use, e.g., the user runs for an interval of time the Fluid application which keeps sending accelerometer information to the hydrodynamics simulation component.

Figure 50. Sample Applications Structure.

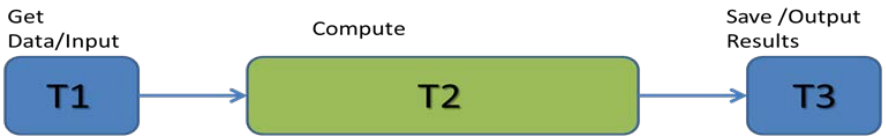


Figure 51. Applications Task Graph (single invocation).

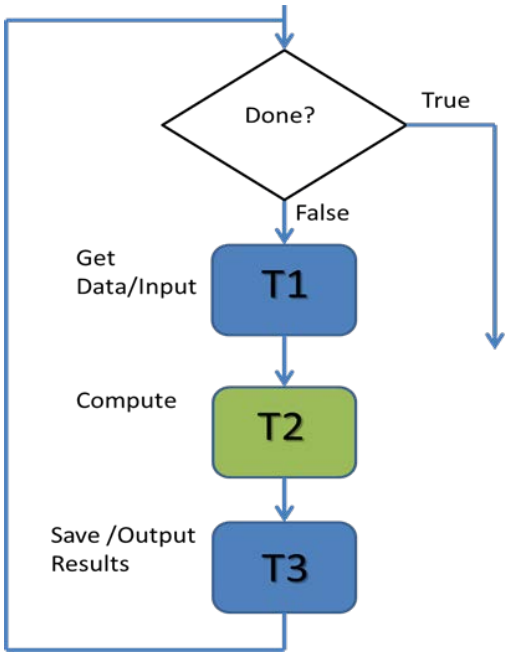


Figure 52. Sample Applications Task Graph (iterated use).

The parameters for the computation intensive task (from [HaPi13]) are summarized in Table 25. The number of instructions have been extrapolated from Figure 9-13 in [HaPi13] and taking into account the performance of mobile node processor ATOM N550 used in [HaPi13] for the experiments.

Application	Code length *l* (MB)	Number of executed Instructions *I*	Memory size *m* (MB)	Input size *in* (KB)	Output size *out* (bytes)
FACE	8.3	$3.664 \cdot 10^6$	99.2	62	60
SPEECH	64.8	$5.445 \cdot 10^6$	111.5	243	50
OBJECT	39.5	$23.289 \cdot 10^6$	113.3	73	50
AR	97.5	$0.931 \cdot 10^6$	287.9	26	20
FLUID	0.5	$0.508 \cdot 10^6$	14.1	0.016	25.000

Table 25. Task parameters for sample applications (from [HaPi13]).

In the literature there are also many other examples of applications (also characterized by more tasks, typically associated to single functions, see, e.g., [Gi09] where examples with up to 20 tasks are considered) but unfortunately the available information is not sufficient for a complete characterization.

3.3.4 Performance Indices

In this subsection we will show to compute energy consumption, the response time and network load of an application given the model described in the previous sections. Power consumption model and network transmission rates are defined in other section of this document.

3.3.4.1 *Energy Consumption*

As we will with the other indices, energy consumption is heavily affected by whether a task is executed locally on the mobile node or remotely in the cloud. In the former case power consumption is only due to local computation; in the latter case, power consumption is instead due to the energy spent to upload the task code and input and to download the task output.

First we will derive the expression for the energy required to execute a single task. For task T_i, let us denote by x_i the binary (decision) variable indicating whether task T_i is executed locally or remotely: we will let x_i=0 to denote that the task is executed locally and x_i=*1* to denote that the task is execute remotely. The energy spent E_{Ti} for the execution of task T_i is then:

$$E_{T_i} = \frac{I_i}{f_{I,M}} P_{cpu,M}(1 - x_i) + \left(\frac{l_i + in_i}{R_{up}} P_{up} + \frac{out_i}{R_{dl}} P_{dl}\right) x_i \quad (27)$$

where:

- $f_{I,M}$ is the mobile node instruction execution rate. We will detail later how it is related to the mobile node CPU characteristics;
- $P_{cpu,M}$ is the mobile node computing power consumption;
- R_{up}(R_{dl}) and P_{up}(P_{dl}) are the upload (download) transmission rate and the associated power consumption.

The (expected) energy to execute the application is then

$$E = \sum_i V_i E_{T_i} \quad (28)$$

Where V_i denote the number of times (or its expected value) task T_i is executed during the application execution.

3.3.4.2 *Response Time Computation*

The response time is also primarily affected by the offloading decision: if executed locally the response time is determined by the local computing resource; otherwise, if executed remotely, the response time depends by the femto computing speed (and the number of users offloading code at the same time on the same femto) and the time required to upload the code and the input and to download the output. We can easily derive the expression for the response time T_{Ti} of a single task T_i:

$$T_{T_i} = \frac{I_i}{f_{I,M}}(1 - x_i) + \left(\frac{l_i + in_i}{R_{up}} + \frac{I_i}{E[f_{I,VM}]} + \frac{out_i}{R_{dl}}\right) x_i \quad (29)$$

where we denote by $f_{I,VM}$ the VM instruction execution rate and by $E[f_{I,VM}]$ its expected value. For the sake of simplicity, in the expression above we omitted the impact of I/O operations.

In order to compute the response time of the application, we cannot proceed as we did for the energy computation by simple considering a weighted sum of the execution times, since we need to account for the fact that the response time of a tasks run in parallel is given by the largest response time among the component tasks (see, e.g., [Lop07]). In this case we derive an expression for the response time *T* recursively. To this end, we find convenient to represent a structured task graph computation by its corresponding syntax tree *G=(V,E)*, which captures the nesting relationship among tasks. In the tree *G*, leaf nodes are the tasks of the task graph, the internal nodes the language constructs and the edges reflect the nesting relationship (see Figure 53).

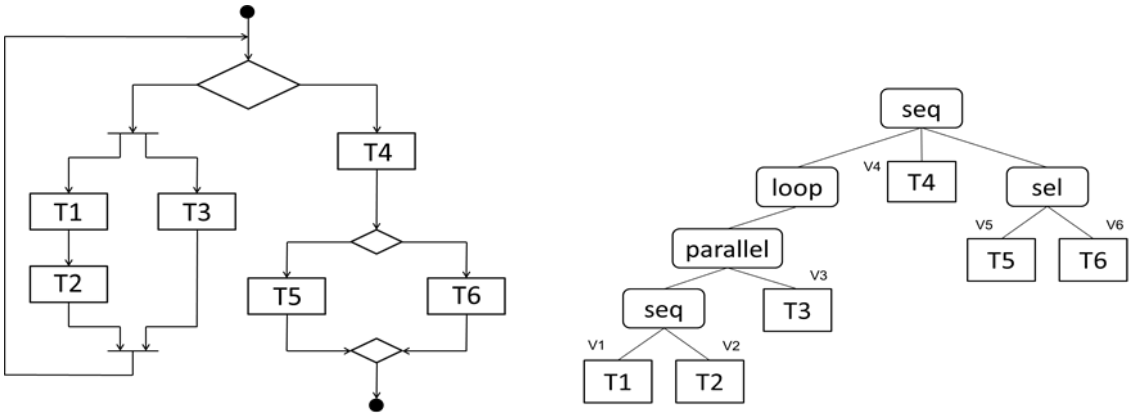


Figure 53. A simple task graph (left) and the corresponding syntax tree.

We will write $i<l$ if node i is a descendant of node l. We will say that a node l is a *direct descendant* of l', denoted by $l<_{dd} l'$, if $l<l'$ and for any other node l'', $l< l''< l'$, l'' is not a parallel construct, *i.e.*, node l is a *direct descendant* of l' if there is no node labeled *parallel* in the path from l to l' in the tree.

We can now state the following result which provides the response time T of a task graph. Let $\mathcal{F}\subset V$ denote the set of non leaf nodes corresponding parallel nodes and let 0 denote the tree root node.
Theorem 1 For $l\in V$, the T_l be recursively defined as follows:

$$T_l = \begin{cases} T_{T_l} & l \text{ is a leaf} \\ max_{n\in d(l)} T_n & l \in \mathcal{F} \\ \sum_{i<_{dd} l} \frac{V_i}{V_l} T_i & l \notin \mathcal{F} \end{cases} \tag{30}$$

The task graph response time is then $T=T_0$.

Theorem 1 provides the response time T recursively. For leaf nodes, which corresponds to tasks, $T_l=T_{Tl}$. For internal nodes, corresponding to portion of the subgraphs, T_l is the response time of that part of the computation. Its expression is the given by the sum of: the overall response time V_iT_i of the tasks which do not appear within a parallel construct (implied by the fact that the sum is over nodes $i<_{dd} l$); and, the response times T_i of the *outer* parallel construct, *i.e.*, parallel constructs which are not nested within other parallel constructs within i. These expressions can be easily derived by visiting the tree in postorder and properly aggregating the response time of the child nodes $d(l)$ to derive the response time of the parent node l.

Single Task Execution Model

Previously, we have considered a simple expression, which consider the number of instruction to be executed and the device instruction execution rate. In the following, we provide more detailed account on how to model the performance a single task running on a computing device and the determining factors.

A task execution time depends on:

- The task characteristics
  - number of instructions I to be executed (Observe that here we refer to number of machine language instructions since these are the instructions actually executed by the CPU);
  - number of I/O operations O to be executed.

- The Hardware/OS/External load
  - clock cycle T_c;
  - CPI - clock cycles per instruction;
  - cost of an I/O operation T_{IO}
  - OS overhead (handling interrupts, system level tasks)
  - background load (CPU-I/O time consumed by other processes)

In the literature, MIPS (million of instructions per second) is often used in place of the more detailed T_c, CPI, which allows to account for different type of processors of the same family (differing for low level architectural characteristics, i.e., pipeline depth). The two are simply related by MIPS=1/T_cCPI. We will briefly discuss the different components/parameters below.

3.3.4.3 *Task Characteristics*

Regarding the number of instructions, we can follow two approaches: either we express directly the number of instruction through some statistics as

- average number of instructions $E[I]$; or better;
- the distribution of the number of instructions $F_I(i) = P[I \leq i]$.

which can be measured/estimated directly; in alternative, if we can consider the computational complexity of the implemented algorithm

- the algorithm complexity $f(n)$, where n is the size of the problem; and
- the distribution $F_n(i)$ of n (which we need to characterize).

Similar type of information is required to characterize number of I/O operations O and its statistics. This, depending on the type of applications could be closely related to the amount of instructions to be executed and/or the data to be transmitted or being completely independent.

3.3.4.4 *Hardware/OS/Background load*

Given the knowledge of the number of instructions I and the number of I/O operation IO, to compute the execution time we need information on the hardware, OS and other user processes behavior.

- The clock speed is a constant on a desktop, but if we consider mobile devices, the CPU speed can be adjusted to save energy thus impacting the execution time on a device.
- The CPI, number of clock cycles per instruction, depends on both the hardware (CPU architecture (pipelined or not), instruction set) and the software (cache misses may result on higher CPI). Hence the CPI is not a CPU dependent constant, but also depends on the tasks memory usage patterns (spatial/temporal locality). We should assume it is a constant.
- The cost of an I/O operation T_{IO} is more complicate to compute: read are faster than write operations and the actual time depends on the OS I/O scheduling, disk caching, etc. A precise characterization is not feasible for our study where we should just assume T_{IO} as a constant which capture average cost.
- Task execution time must also account for the OS overhead: contexts switching, interrupt handling, *etc.*, which reduces the amount of CPU cycles available for user processes execution.
- Finally, last but not least we need also to account for system load represented by the multiple user processes which shares the CPU and OS services.

A simple formula for a task execution time T which accounts for all these factors we could consider is the following:

$$T = I \cdot CPI \cdot T_c \cdot S_{c,OS} + IO \cdot T_{IO} \cdot S_{IO} \quad (31)$$

where $S_c \geq 1$ and $S_{IO} \geq 1$ represent the CPU and the IO slowdown factors, respectively, which capture the relative computing speed reduction due to the OS ($S_{c,OS}$) and background load ($S_{c,load}$). In order to account for the OS and External load separately, we rewrite them as follows:

$$S_c = S_{c,OS} \cdot (1 + S_{c,load}) \quad (32)$$

and

$$S_{IO} = S_{IO,OS} \cdot (1 + S_{IO,load}) \quad (33)$$

where the subscripts *OS* and *load* denote the impact of the OS and of the external load, respectively. Consider S_c: $S_{c,load} \geq 0$ takes into account the presence of other users' processes which share the CPU. We can approximate its value with the average number of running user processes during our task execution. $S_{c,OS}$, instead accounts for the OS overhead which affects all running processes. Similar arguments apply to the I/O slowdown S_{IO}.

3.3.4.5 ***Baseline model***

In the following, we will consider a simple task with no I/O. The expressions can be easily generalized. Its running time - assuming it is taken when no other user process is active, *i.e.*, when $S_{c,load} = 0$ is simply

$$T = I \cdot CPI \cdot T_c \cdot S_{c,OS} \quad (34)$$

We find convenient rewrite it as follows:

$$T = \frac{I}{f_I} \quad (35)$$

where we define

$$f_I = \frac{1}{CPI \cdot T_c \cdot S_{c,OS}} \quad (36)$$

as the (system unloaded) average instruction execution rate. This can be easily generalized to account for load and I/O.

Mobile node execution time

For a mobile node, we can consider the model just described, where we characterize the device with its instruction execution rate $f_{I,M}$. The execution time is then just

$$T_M = \frac{I}{f_{I,M}} \quad (37)$$

VM execution time

We now address the problem of characterizing the execution time T_{VM} of a task running on a VM. In the following, we will denote by $f_{I,f}$ the instruction rate of the femto. For simplicity, we will assume the femto has only one CPU/core. The results can be generalized to the case of multiple CPUs/cores.

We have two different scenarios based on whether or not the physical CPU is scheduled or not according a work-conserving discipline.

3.3.4.6 *Non work conserving discipline*

This is the simplest case. A non work conserving discipline amounts to assign a constant rate to each VM which cannot be exceed even if spare capacity is available. This solution provides maximum isolation among the different VMs. Let $f_{I,VM}$ be the fixed instruction rate associated to a VM. Then we can write

$$f_{I,VM} = \frac{f_{I,f}}{N_{VM}(t)} \tag{38}$$

In general, since different VM can be characterized by different weights, as detailed in Section 4.3, we have

$$f_{I,VM_i} = \frac{f_{I,f} \cdot VM_{weight,i}}{\sum_k VM_{weight,k}}$$

where $VM_{weight,i}$ denotes the weight associated to VM i.

3.3.4.7 *Work conserving discipline*

CPU work conserving disciplines make the best of available resources by never leaving the CPU idle if a VM is active. In this setting, we assume that VMs share the computational capacity of the femto according to the Processor Sharing discipline (this corresponds for instance to the Xen based system using a credit based system). Hence at a given instant the capacity of femto is equally shared among the active (non idle) VMs. We can thus define the instantaneous VM user instruction execution rate as

$$f_{I,VM} = \frac{f_{I,f}}{N_{VM}(t)} \tag{39}$$

where $N_{VM}(t)$ denote the number of active (non idle) VMs at time t. Since $N_{VM}(t)$ is not a constant, so is $f_{I,VM}(t)$. For the following analysis, for simplicity we will assume a discrete time model. Let

$$I_{VM}(t) = \sum_{i=1}^{t} f_{I,VM}(t) \tag{40}$$

denote the number of instructions executed in the interval $\{1,\ldots,t\}$. The task execution time T_{VM} corresponds to the first time t such that $I_{VM}(t) \geq I$, basically a *first passage time*, and $T_{VM} = T_{VM}(I)$ (to stress its dependence on the number of instruction to be executed I) can be written as:

$$T_{VM}(I) = \min\{t \mid \sum_{i=1}^{t} f_{I,VM}(i) \geq I\} \tag{41}$$

$T_{VM}(I)$ is clearly a random variable. Its value depends on what the other VMs are doing during the execution of a given task.

In the rest of the section, we show how we can compute bounds, an approximate expected value, and, if we collect some statistical information on the femto usage over time, the distribution of the $R_{VM}(I)$.

3.3.4.8 *Lower Bound*

A lower bound can be easily computed under the optimistic assumption that only one VM is running on the femto. We readily obtain:

$$T_{VM}^{min}(I) = \frac{I}{f_{I,f}} \tag{42}$$

3.3.4.9 *Upper Bound*

An upper bound on the execution time depends instead on the femtocloud service model. For performance reasons, it is reasonable to assume that only a maximum amount of VMs N_{max} are ever allocated on a femto. Hence we have:

$$T_{VM}^{max}(I) = \frac{I}{N_{max} f_{I,f}} \tag{43}$$

3.3.4.10 *Approximate Expected Value*

The expected execution time can be approximated by computing the average number of active VMs $E[N_{VM}]$ (this can be done via monitoring) as follows:

$$E\left[\, T_{VM}(I)\right] = \frac{I}{E[N_{max}] f_{I,f}} \tag{44}$$

3.3.4.11 *Distribution of T_{VM}*

Assume now that we have some statistical information on how $N_{VM}(t)$ (and hence $f_{I,VM}(t)$) varies with time. More precisely, let us model $N_{VM}(t)$ as a Markov Chain (we can estimate the one-step probabilities directly by taking measurements on the femtos). We can associate to this Markov chain a Markov Reward Model (MRM) by associating a reward equal to the VM instruction rate corresponding to each femto state.

It is easy to realize that the number of instruction executed in the interval $\{1,\ldots,t\}$, $I_{VM}(t)$, is then just the accumulated reward over the interval $\{1,\ldots,t\}$. As a consequence, $T_{VM}(I)$ is then the time it takes to accumulate a reward of at least I. The distribution of $T_{VM}(I)$ can be computed in terms of the distribution of $I_{VM}(t)$ using standard techniques . For instance, by observing that

$$\mathrm{P}[\mathrm{T_{VM}}(\mathrm{I}) \leq \mathrm{t}] = 1 - \mathrm{P}[\mathrm{I_{VM}}(\mathrm{t}) \geq \mathrm{I}] \tag{45}$$

which can be computed numerically using the techniques described in [Tri94].

Network load

We can easily derive the average upload(download) rate induced by a task execution as the ratio of the amount of data uploaded(downloaded) by a task over its execution time.

$$Nrate_{i,UP} = \frac{(l_i + in_i)x_i}{\frac{I_i}{f_{I,M}}(1-x_i) + \left(\frac{l_i+in_i}{R_{up}} + \frac{I_i}{E[f_{I,VM}]} + \frac{out_i}{R_{dl}}\right)x_i} \tag{46}$$

$$Nrate_{i,DL} = \frac{out_i x_i}{\frac{I_i}{f_{I,M}}(1-x_i)+\left(\frac{l_i+in_i}{R_{up}}+\frac{I_i}{E[f_{I,VM}]}+\frac{out_i}{R_{dl}}\right)x_i} \quad (47)$$

We can readily obtain similar expression for the entire application.

3.4 *Model of SCeNBce Real System (cloud side)*

Modeling the small-cell cloud system requires the analysis of the real system, so that appropriate abstractions can be made to accurately represent the system architecture and behavior. This section studies the physical aspects of the SCeNBces along with the virtualization aspects that include the hypervisor functionality and virtual machines.

3.4.1 **Physical SCeNBce**

As described in [TROPIC-D41] the physical SCeNBce is comprises the following parts:

- CPU
- Memory
- HDD
- Network Interface
- Baseband processor
- Modulator / demodulator

The relevant components for the cloud part that are to be modeled for the simulation are the CPU, memory, HDD and network interface. The others can be considered to belong to the radio dimension.

3.4.2 **Virtualized SCeNBce**

As stated in [TROPIC-D41], the hypervisor for the SCeNBce virtualization is Xen. This section analyses the relevant aspects of the virtualization that are to be modeled for the simulation.

3.4.2.1 *CPU scheduling*

Xen is unique among VM platforms because it allows users to choose among different CPU schedulers. Over the last three years, three different CPU schedulers were introduced, all allowing users to specify CPU allocation via CPU shares (weights). Some examples of schedulers are Borrowed Virtual Time (BVT), Simple Earliest Deadline First (SEDF) and Credit Scheduler [Cher07].

The default scheduler for Xen is Credit Scheduler. The SEDF and BVT schedulers are still optionally available but the plan of record is for them to be phased out and eventually removed [CreditScheduler]. Therefore, for the SCeNBce modeling, Credit Scheduler is considered.

3.4.2.1.1 Credit-based CPU scheduler

The credit scheduler is a proportional fair share CPU scheduler built from the ground up to be work conserving[1] on SMP[2] hosts [CreditScheduler].

[1] When a CPU finishes executing, the released computation power is distributed among the other executing CPUs, that is, preventing computing power from going unused.
[2]Symmetric multiprocessing (SMP) involves a multiprocessor computer hardware and software architecture where two or more identical processors are connected to a single shared main memory, have full access to all I/O devices, and are

Each domain (including Host OS) is assigned a ***weight*** and a ***cap***. A domain with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended host. Legal weights range from 1 to 65535 and the default is 256. The cap optionally fixes the maximum amount of CPU a domain will be able to consume, even if the host system has free CPU cycles. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc... The default, 0, means there is no upper cap.

The credit scheduler **automatically load balances guest VCPUs across all available physical CPUs** on an SMP host. The administrator does not need to manually pin VCPUs to load balance the system. However, he can restrict which CPUs a particular VCPU may run on.

Due to performance issues, a parameter called ***ratelimit*** was introduced in Xen 4.2 [CreditScheduler]. The *ratelimit* is a value in microseconds. It is a minimum amount of time which a VM is allowed to run without being preempted. The default value is 1000 (that is, 1ms). So if a VM starts running, and another VM with higher ***priority*** wakes up, if the first VM has run for less than 1ms, it is allowed to continue to run until its 1ms is up; only after that will the higher-priority VM be allowed to run. This feature can be disabled by setting the *ratelimit* to 0.

The ***timeslice*** (time a **VCPU** receives before being preempted to run another) for the credit scheduler by default is fixed at 30ms. This is actually a fairly long time. It's great for computationally-intensive workloads, but not so good for latency-sensitive workloads, particularly those involving network traffic or audio. Xen 4.2 introduces the *tslice_ms* parameter, which sets the *timeslice* of the scheduler in milliseconds and can be set at will by the user.

Credit Scheduler algorithm:

Each CPU manages a local ***run queue*** of runnable VCPUs. This queue is sorted by VCPU ***priority***. A VCPU's priority can be one of two values: **over** or **under** representing whether this VCPU has or hasn't yet exceeded its fair share of CPU resource in the ongoing accounting period. When inserting a VCPU onto a run queue, it is put after all other VCPUs of equal priority to it.

As a VCPU runs, it consumes **credits**. Every so often, a system-wide accounting thread re-computes how many credits each active VM has earned and bumps the credits. Negative credits imply a priority of *over*. Until a VCPU consumes its allotted credits, its priority is *under*.

On each CPU, at every scheduling decision (when a VCPU blocks, yields, completes its time slice, or is awaken), the next VCPU to run is picked off the head of the run queue. The scheduling decision is the common path of the scheduler and is therefore designed to be light weight and efficient. No accounting takes place in this code path.

When a CPU doesn't find a VCPU of priority *under* on its local run queue, it will look on other CPUs for one. This load balancing guarantees each VM receives its fair share of CPU resources system-wide. Before a CPU goes free, it will look on other CPUs to find any runnable VCPU. This guarantees that no CPU is free when there is runnable work in the system.

3.4.2.2 *Memory scheduling*

The hypervisor must allocate all memory used by the domains, but it only deals with physical memory and the page table (which associate chunks of physical memory with pages of virtual memory). The guest OSs handles all other memory management functions [Takemura09].

---

controlled by a single OS instance, and in which all processors are treated equally, with none being reserved for special purposes.

3.4.2.2.1 Static memory allocation

The basic memory scheduling policy a hypervisor can do is to allocate static physical memory for each guest VM. For example, if the physical machine has 4GB of RAM and we want to run as many 1GB VMs as possible, we could run at most three, because the hypervisor itself and domain 0 also require some physical memory.

3.4.2.2.2 Memory overcommit

Memory overcommit provides the ability for the sum of the physical memory allocated to all active domains to exceed the total physical memory on the system. For example, in a 4 GB RAM physical machine, we could run six, ten or even more 1 GB RAM virtual machines. Xen implements this new memory overcommit feature since version 3.3.

Memory is taken from one domain and given to another using the existing Xen “ballooning” mechanism, first introduced in [Wald02]. A domain that is idle (or nearly so) is probably not using much memory; this memory can be made available to use in another domain, or for a newly created domain. In the following, it is briefly explained how the Xen Dynamic Memory Control (DMC) carries out this mechanism [XenDMC].

In order to add memory to or remove memory from a running guest, DMC relies completely on the action of a balloon driver running within the guest operating system. The balloon driver works by inflating or deflating a memory balloon - a special area of memory within the guest's physical address space.

During normal operation, every physical memory page within the guest is backed up by a physical memory page on the host. In order to reduce a guest's memory allocation, the balloon driver running in the guest "inflates" its memory balloon. The balloon driver achieves this by using an OS-specific technique to allocate physical memory pages from the guest. Initially, these guest physical memory pages will be backed by host physical memory pages, just like any other pages in the guest. However, after acquiring guest physical memory pages, the balloon driver immediately informs the hypervisor that it may recycle the host physical memory pages that back them. The hypervisor immediately revokes the guest's access to these host physical memory pages, and makes them available for use by other guests.

Inflating a balloon measurably increases physical memory pressure within the guest. From the guest's point of view, it can no longer use the memory that's been taken away. The balloon driver appears to be rather like a long-lived process that's using some of the available guest physical memory.

In order to increase a guest's memory allocation, the balloon driver running in the guest is asked to "deflate" its memory balloon. However, in order to do this, the balloon driver must first ask the hypervisor to re-back ballooned-out guest physical pages with host physical pages. The hypervisor may refuse, and will certainly do so if there are no remaining physical pages available on the host.

In [Magen08] it is analyzed the performance of the memory overcommit over Xen through the “self-ballooning” mechanism. It is stated a default overcommit ratio of 7:4(e.g. 7x512MB *loaded* guests, 2GB physical memory). It is also stated an “aggressive” configuration of 15:4 (e.g. 15x512MB *idle* guests, 2GB physical memory). This values do not represent typical values of the memory overcommit ratio since it is a dynamic mechanism managed by the hypervisor, but it gives a hint about how much the memory can be overcommitted.

At the moment this work is being done and as far as we know, there is not any available study about Xen memory overcommit performance.

Other memory management mechanisms that are being implemented in the latest Xen versions, but not fully set yet are:

- *Memory Sharing*. Allow sharing of identical pages between HVM guests. Tech preview.
- *Memory Paging*. Allow pages belonging to HVM guests to be paged to disk.
- *TMEM (Transcendent Memory)*. Claiming underutilized memory in a system and making it available where it is most needed. This is in experimental state.

One important issue for the memory management indicated in the Xen best practices documentation[3] is that it is recommended to dedicate a fixed amount of RAM for Xen dom0 in order to avoid problems when the ballooning mechanism modifies the dom0 allocated memory.

3.4.2.3 *Disk allocation*

Xen virtual machines find their storage by examining the configuration setup upon creation. Xen supports many different storage options, each with its own strengths, weaknesses, and design philosophy. These options broadly fall into the categories of file based and device based. An extensive study of the Xen storage options has been done in [TROPIC-D41]. Regardless the storage option, the virtual disk size of the virtual machine must be indicated upon creation. This virtual disk represents the storage capacity of the virtual machine.

3.4.2.4 *Network bandwidth allocation*

By default, Xen guests can use any bandwidth setting available which your physical network card supports. The physical network card must be mapped to one of virtual machine's virtual network interfaces. However, in some environments it may be required to limit the network bandwidth available to certain guests. This can be used to implement basic Quality of Service on a host running multiple virtual machines. In Xen the "*rate*" parameter part of the VIF (Virtual Interface) entries can throttle guests.

The *rate* option can be added to the VIF entry in a virtual machine configuration file to limit a virtual machine's network bandwidth.

3.4.2.5 *Virtual CPU scheduling*

In the same way the hypervisor schedules the CPU computing power among the executing VCPUs, within a virtual machine, the operating system schedules the computation among its virtual CPUs. The operating system chosen in TROPIC is Linux [TROPIC-D41].

The Linux CPU scheduler is responsible for keeping the CPUs in the system busy. The Linux scheduler implements a number of scheduling policies, which determine when and for how long a thread runs on a particular CPU core.

The default scheduling policy uses the Completely Fair Scheduler (CFS) [CFS] to provide fair access periods for all threads using this policy. The main idea behind the CFS is to maintain balance (fairness) in providing processor time to threads. This means processes should be given a fair amount of the processor. When the time for threads is out of balance (meaning that one or more tasks are not given a fair amount of time relative to others), then those out-of-balance threads should be given time to execute. CFS establishes a dynamic priority list partly based on the niceness value of each process thread. This gives users some indirect level of control over process priority.

[3] http://wiki.xen.org/wiki/Xen_Best_Practices

Linux incorporates process and thread scheduling by treating them as one in the same. A process can be viewed as a single thread, but a process can contain multiple threads that share some number of resources (code and/or data).

3.5 *SCeNBce Simulation Model (cloud side)*

This section describes a possible simulation model of the SCeNBce. This model intends to provide the necessary abstractions in order to simulate the femto-cloud behavior from the cloud point of view. A preliminary cloud model is described in [TROPIC-D21]. The model designed here takes this preliminary model as a basis and extends it modeling all the necessary abstractions to carry out an accurate simulation of the femto-cloud system.

Several aspects are essential and can be identified as indispensable to be modeled.

- Physical SCeNBce features including physical resources within the SCeNBce: CPU, memory, disk and network
- Hypervisor functionality
  - Scheduling policy to allocate computing power to VMs
  - Scheduling policy to allocate memory to VMs
  - Scheduling policy to allocate network bandwidth to VMs
- VM features
  - Virtual resources (CPU, memory, disk, …)

Other aspects such as the VM allocation policies or management algorithms are closely related to the previous ones, but as the section focus only in VMs and SCeNBce modeling and not in the SCM functionality, they fall out of the scope of the section.

In the following these aspects are further explained.

3.5.1 **Physical SCeNBce model**

One basic aspect to take into account when modeling the virtualization on the SCeNBces is the physical resources. The whole SCeNBce includes both radio and cloud sides, but for the VM modeling, only the “cloud” (or computing) side is considered in this section.

The physical SCeNBce from the computing side can be represented by the abstraction of its CPU, memory, disk, and network card.

- ***CPU***: Composed of one or more computing cores, each of them with a certain computing speed. As the SCeNBce is SMP, all the CPU cores will have the same speed. On the SCeNBce the CPU can be modeled as a list of objects, each of them representing one individual core
  - ***Core***: Each CPU core can be represented by its speed and state. The speed is equal among all the cores within the CPU and measured in Millions of Instructions per Second (MIPS). The state can be mainly FREE (waiting for a task to execute) or BUSY (executing a task)
- ***Memory***: The RAM can be represented by its capacity
- ***HDD***: The disk can be represented by its storage capacity
- ***Network card***: it can be represented by the physical network card speed

3.5.2 **Virtualized SCeNBce model**

This section describes the simulation model for the virtualized SCeNBce.

3.5.2.1 ***Virtual machine model***

The parameters that represent each VM are:

- ***Virtual CPUs***: each virtual machine has a number of virtual CPUs. This can be represented by a set of objects, each of them representing an individual VCPU
    - ***VCPU***: each virtual CPU can be represented by its state (FREE, BUSY). No more parameters are needed since the computational capabilities are defined with the *weight* and *cap* parameters of each VM
- ***Virtual Memory***: The virtual allocated memory can be represented by the allocated capacity
- ***Virtual Disk***: In the same way, the virtual storage can be represented by its capacity
- ***Backhaul Bandwidth***: The allocated network bandwidth for the VM can be represented by a value that can range from the maximum physical network card speed to some inferior value, thus limiting the allocated bandwidth
- ***State***: As defined in [TROPIC-D41], the VM state can be RUNNING, SAVED, PAUSED or DEFINED. The UNDEFINED state is not considered in the model, since this state represents the inexistence of the VM
- ***Weight***: VM weight for CPU scheduling
- ***Cap***: Maximum CPU consumption representing the percentage of usage of one CPU
- ***Ratelimit***: Minimum VM execution time
- ***Timeslice***: VM execution time before being preempted

It is important to note that in the model it is not considered the difference between primary and secondary VMs. The goal of the model is to describe a general VM, regardless the application it's running on.

3.5.2.2 ***CPU scheduling model***

This section describes the simulation model for the CPU scheduling mechanisms. That is, how the physical computational resources are shared among virtual resources, and how the virtual CPUs computational capabilities are allocated to the application.

3.5.2.2.1 Scheduling of VCPUs among CPU cores

The assignation of the VCPUs to the physical core that executes them is carried out by the hypervisor based on the proportional fair policies of the credit scheduler. The goal of this scheduler is to share the computational resources among the VCPUs in a fair, work conserving manner.

Furthermore, each VM has a *weight* and a *cap* that define the overall and maximum computational resources they use. The *weight* defines how much CPU usage the VM has with respect to the other VMs, so it's an important parameter to take into account in the CPU scheduling model. The *cap* is not

considered in the model, as in principle, no CPU limitation is to be set to any VM, so every VM takes as much CPU as possible according to the scheduling policies.

In turn, each VCPU belongs to one VM. That implies that VCPUs of a VM with high weight, receive more computational resources than VCPUs belonging to a VM with low weight. This can be modeled dividing the VM weight by its number of ***active*** VCPUs and associating that value to the VCPU.

For example, let's consider two virtual machines with default weight (256) running in the same physical machine. VM1 has one VCPU (VCPU1) and VM2 has two active VCPUs (VCPU2 and VCPU3). As the two VMs have the same weight, each VM receives 50% of the physical CPU usage. In turn, this 50% has to be shared among the VCPUs of the VM, so the VCPU of the VM1 receives 50% and the VCPUs of the VM2 receive 25% each. That is, we can associate the VCPU weight with the values 256 for VCPU1 and 128 for VCPU2 and VCPU3. In this way we can model the CPU usage by the individual VCPUs.

Now, let's suppose VM2 is running just one thread that is running on VCPU2 while VCPU3 remains idle. In this case, VCPU2 receives the 50% of the computational power, and we can associate a weight of 256 to both VCPU1 and VCPU2.

In short, each active VCPU is associated with a dynamic *weight* parameter as the VMs do. This parameter can be derived as follows:

$$VCPU\ weight = \frac{VM\ weight}{Number\ of\ active\ VCPUs\ of\ the\ same\ VM} \tag{48}$$

This weight has to be recalculated for every VCPU within a VM each time a VCPU goes from BUSY to FREE or vice-versa.

All active VCPUs of any VM must run within the physical CPU cores. They share the physical computational resources according to the aforementioned credit scheduler. The goal of the CPU scheduler is to share the computational resources fairly in a work conserving manner. That means that each VCPU receives computational resources according to its VCPU weight and that all the CPU resources are utilized when possible.

An illustrative example can be the following: four VMs with the same weight execute over a physical CPU with two cores. Each VM has only one VCPU. In this case, the scheduler would allocate two VCPUs in the run queue of each core, providing each VCPU with 50% of the physical core. Let's say one VCPU finishes its execution. Now, the scheduler would swap VCPUs between both run queues according to the credit algorithm in order to provide a fair distribution of the computational resources. The result is that each VCPU now receives a computational power equivalent to 66.6% of a physical core.

The upper limit of the computational resources given to a VCPU is 100% of a physical core, since VCPUs can only be executed in one core at the same time. VCPU's execution is not parallelizable and during their timeslice they can only run in one physical core. The fact that the VCPU execution can span among different physical cores sharing the overall computational capacity is due to the fact that the VCPU timeslices can span among the physical cores, but only one physical core can be executing a VCPU timeslice at a time.

The challenge for our simulation model is how to abstract the scheduler functionality. There are two main aspects to take into account: the amount of computational resources a VCPU gets, and the overall execution time of the VCPU, which encompass the execution time in the physical cores and the waiting time in the run queues while other VCPUs are executing.

The computational resources allocated to a VCPU at a given time can be modeled as follows:

- When there is less number of active VCPUs than physical cores: each VCPU receives a 100% of a physical core without waiting times
- Otherwise, if there are more active VCPUs than physical cores, the overall computation capability is shared among the VCPUs taking into account their individual weight. VCPUs are executed in a physical core during a certain *time slice*. Then they are preempted and queued for its next execution

Let $\{VCPU_1, \ldots, VCPU_L\}$ be the set of active VCPUs at a given time and C be the number of physical cores. In this case, the amount of computational resources used by a certain VCPU in terms of percentage of the computational capacity of one physical core can be expressed as:

$$VCPU_i \; core \; utilization \; (\%) = \frac{100 * C * VCPU_i \; weight}{\sum_{k=1}^{L} VCPU_k \; weight} \tag{49}$$

This utilization must be recalculated for all active VCPUs whenever a VCPU goes to idle to active or vice-versa, that is, each time a VCPU enters or exits the aforementioned active VCPU set.

3.5.2.2.2 Simplified fair work-conserving CPU scheduler

In the following, the usage of a simplified fair work-conserving CPU scheduler is proposed as an alternative to the credit scheduler for the simulation model.

This scheduler achieves the same goals than the credit scheduler, that is, fair sharing of computational resources among VCPUs and usage of the whole CPU capacity when possible. The main difference is that it uses one general queue rather than individual queues for each physical core.

The CPU allocation policy is the same than in the credit scheduler: the computational resources are shared among VCPUs fairly depending on the VCPU weight. The work-conserving policy is also a feature of this scheduler: whenever there are available computational resources they are allocated to waiting VCPUs.

A model of this simplified scheduler is depicted in Figure 54.

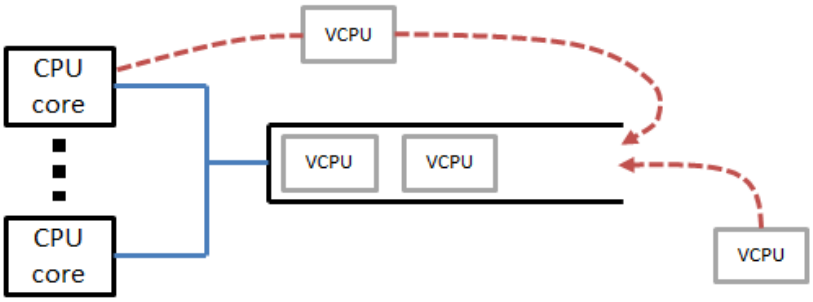


Figure 54. Simplified CPU scheduler model.

The scheduler operation is similar to the credit scheduler. When a VCPU goes from idle to busy it is pushed into the queue. CPU cores execute VCPUs during a certain amount of time known as *timeslice*. When a CPU core is free it takes the next waiting VCPU, executes it during its *timeslice* and queues it again. This process loops until the VCPU finishes its execution, where it is idled and not queued again.

In general, this scheduler can be considered a good approximation to the credit scheduler. The CPU allocation is the same and the overall execution time is similar. The overall execution time differs due to the fact that there is a single queue rather than an individual queue for each core. But note that in the

case in which there is only one physical core per machine, this simplified scheduler is equivalent to the credit scheduler.

In order to model the fair CPU allocation, two main approaches can be followed. One is to have a variable *timeslice* dependent on the VCPU weight and a FIFO queue. The other is to have a fixed *timeslice* and queue VCPUs with priorities taking into account their weight. The first approach is the one followed in the simulation model. The latter is a kind of credit algorithm which increases the complexity of the model with no big different results from the former one, since the fair CPU scheduling is achieved by both approaches and the overall execution time slightly differs since the different VCPU waiting times. The approach considered in the simulation model is a variable VCPU *timeslice*.

The VCPU *timeslice* at a given time can be derived as follows:

$$VCPU_i\ timeslice\ (ms) = \frac{30 * VCPU_i\ weight * L}{\sum_{k=1}^{L} VCPU_k\ weight} \quad (50)$$

Where we take as the basic *timeslice* the default 30 milliseconds and $\{VCPU_1, \ldots, VCPU_L\}$ represents the set of active VCPUs at a given time.

Following this model, a VCPU which weight is the average weight of all the active VCPUs will execute during a time slice of 30 ms, a VCPU which weight doubles the average will execute during a time slice of 60 ms and so on.

3.5.2.3 ***Memory scheduling model***

Two kind of memory allocation can be used. Static memory allocation and memory overcommit.

3.5.2.3.1 Static memory allocation

In case static memory allocation is used the model for the memory scheduling is straightforward. A certain amount of the physical memory is assigned to the VM, and this memory cannot be used for other purposes.

This implies that the number of VMs on a SCeNBce is limited by the amount of available memory. A new VM that demands more memory than physically available cannot be created.

3.5.2.3.2 Memory overcommit

When using memory overcommit, more memory than physically available can be allocated to a VM. This is driven by the Xen ballooning algorithms.

Within TROPIC scope, it is not relevant to model and simulate memory ballooning, since it is intended to optimize resource provisioning rather than improve application performance.

Excessive overload can be avoided when creating new virtual machines in a SCeNBce with high memory overcommit ratio. This is part of the SCM functionality, which decides when and where to deploy new VMs based on the QoS mechanisms.

The memory overcommit ratio can be easily derived by looking into the allocated memory to VMs in a SCeNBce.

3.5.2.4 *Disk allocation model*

Disk is allocated to VMs in a static manner. The allocated disk to VMs must always be less than the total physical disk.

This is an important aspect to take into account specially when migrating VMs, since their disk storage must be migrated with them.

3.5.2.5 *Network allocation model*

The network bandwidth is allocated to VMs in a fair manner. VMs can use any bandwidth available that the network card supports.

In case different VMs are transmitting at the same time, the bandwidth is divided evenly among them, unless some VM has set the *rate* parameter, which limits the maximum bandwidth that can be allocated to that VM.

3.5.2.6 *Virtual CPU scheduling model*

Processes in Linux are composed of threads. These threads are scheduled among the VCPUs by the Linux CPU scheduler. Assuming the default scheduling policy is used (Completely Fair Scheduler, CFS) it can be considered that threads are assigned evenly to VCPUs.

In case the VMs are only configured with one VCPU, this model is rather straightforward. In this case, the VCPU is in busy state as long as any thread is being executed in that VM. Otherwise the VCPU is in idle state. The VCPU usage is shared evenly among the executing threads.

In case a VM is configured with more than one VCPU, the overall computational capacity of the VM is shared evenly among the different threads according to the CFS policies. If only one thread is present, this thread is executed in one VCPU. If more threads are present, they are scheduled evenly among the VCPUs.

4 SMALL CELL CLOUD SYSTEM

The small cell cloud introduces new control entity – the SCM. Besides this entity, also the UE, SCeNB and security gateway must be equipped with additional components and modules in order to support offloading. This section defines new modules and interaction among modules and entities in the network.

4.1 *Functional block scheme of the SCC*

To represent functionality of individual parts of the SCC system, we distinguish four subsystems according to its relation to specific part of the SCC system: Middleware, Security, Management, and Communication. For each subsystem, specific components, encompassing several functional modules, are designed. To easy orientation in the system, we depict the SCC functional scheme using specific symbols as summarized in Table 26.

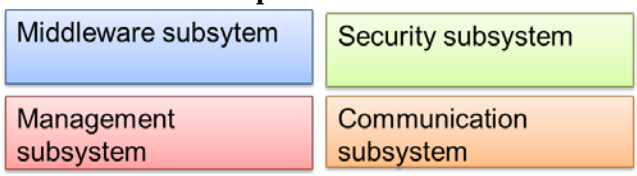
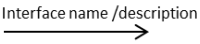
Entity	Representation
Subsystem	
Component	
Module	
Interface between modules within a component	
Interface between modules in different component	

Table 26. Symbols and representation of entities in the SCC system scheme.

The core of the SCC system with all cloud and radio related modules is shown in following figure.

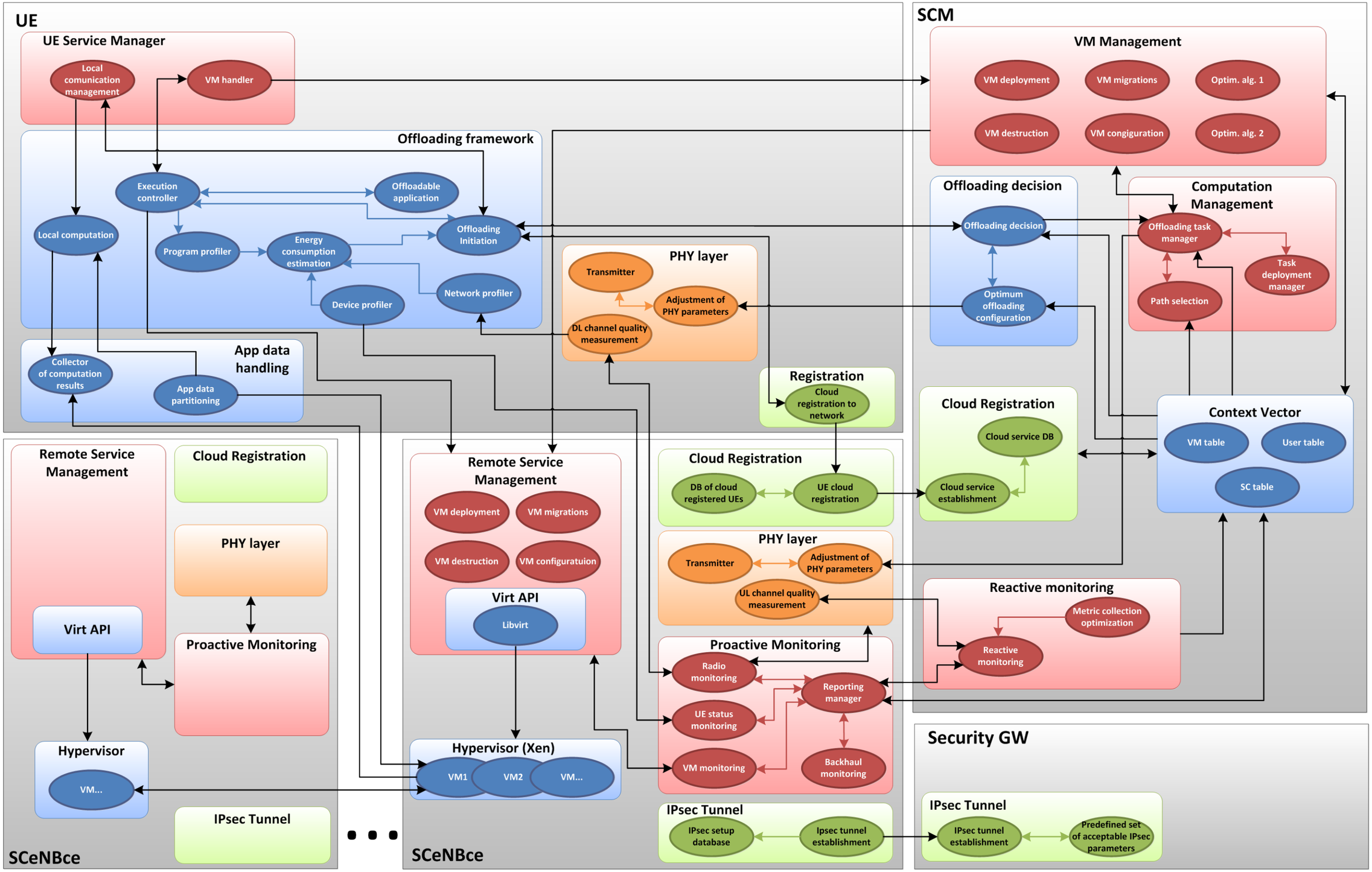


Figure 55. Core of the SCC system.

4.2 *Description of individual modules and components of SCC system*

4.2.1 SCM

The modules located at the SCM are the following:

4.2.1.1 *VM management component*

This component is in charge of the management of the VM lifecycle. That includes VM creation and deployment, migrations, saving, awakening and destruction modules. For this management aspects, optimization algorithms are implemented in order to optimize the overall QoE of the system's users as described in [TROPIC-D52].

4.2.1.2 *Computation management component*

This component defines how to distribute tasks to the available SCeNBces in order to meet user's requirements and type of service into account. Also, potential optimization of the path for data delivery to the VMs and delivery of results from the VMs to the UE is carried out by this component. Distribution and deployment of the computation among small cells is addressed in [TROPIC_D52] while path selection is addressed in Section 7 [TROPIC_D51].

4.2.1.3 *Reactive monitoring component*

This component is in charge of the reactive monitoring carried out by the SCM. The operation of this module is based on polling messages forwarded to the SCeNBces and the reception of the responses containing the monitoring information. Detailed information about this component can be found in [TROPIC-D42] and [TROPIC-D52.]

4.2.1.4 *Context vector*

The context vector is the component in charge of storing the context information of the SCC system for each of the SCeNBCEs, users, VMs and applications. A detailed description can be found in [TROPIC-D42].

4.2.1.5 *Cloud registration*

This component handles the new connections of both SCeNBces and users from the security point of view. It registers the appropriate data in the context vector and sends back to the connection entity the necessary parameters.

4.2.1.6 *Offloading decision*

The offloading decision is made taking into account energy consumption and delay constraints and it is the result of a joint optimization over the radio and computational resources. Then the decision on whether to offload the computation or to perform it locally has to be taken by considering the following issues:

- To minimize energy consumption at the UE under latency constraint (or vice versa)
- To guarantee the stability of the queue of instructions to be executed at the mobile side

4.2.2 SCeNBce

The modules located at the SCeNBces are the following:

4.2.2.1 *Proactive monitoring component*

This module is in charge of the proactive monitoring carried out by the SCeNBces. The operation is based on the proactive forwarding of monitoring messages from the SCeNBces to the SCM without a

previous polling message from the SCM. Detailed information about this module can be found in [TROPIC-D42] and [TROPIC-D52].

4.2.2.2 *Remote service management component*

This component is in charge of the VM lifecycle management in the SCeNBces including VM deployment, migration, destruction and configuration. It interacts with the hypervisor to handle the management actions indicated by the SCM.

4.2.2.3 *PHY Layer*

This component represents physical layer at radio interface (LTE). It enables configuration of the physical layer channel and communication as well as it provides inputs related to radio channels to the context vector.

4.2.2.4 *Cloud registration*

This module is in charge of the following aspects:

- When the SCeNBce switches on, it connects with the SCM registering the SCeNBce in the system
- When a new user attaches to the SCeNBce, this module handles the message passing with the SCM to register the new user

4.2.2.5 *IPsec tunnel*

This module is used to establish IPsec connection between the SCeNBce and Security GW to provide secure communication between both.

4.2.3 **UE**

The modules located at the UE are the following:

4.2.3.1 *UE service manager*

This component carries out the VM lifecycle management from the UE point of view. It mainly handles the primary VMs associated to the user.

4.2.3.2 *Registration*

This component is the one in charge of the connection with the SCC. It connects to its pair in the SCeNBce and the connection message reaches the SCM. The related information for the connection reaches back this module, which updates the appropriate configuration parameters for future connections to the SCC.

4.2.3.3 *App data handling*

This component handles partitioning of applications and data if processing at more than one SCeNBce is assumed. At the same time, results of computation are collected and merged by this module.

4.2.3.4 *Offloading framework*

The offloading framework is responsible of deciding, in runtime, which modules of the TROPIC apps to offload to the SCC. This can be done in two different ways: (a) locally, on the UE-side, or (b) globally, on the SCC side. In the first case (a) (decision on the UE), the decision uses information

locally known at the device and the key performance indicator is whether the user saves battery or time. A detailed description of the offloading framework that operates at UE side is presented in [TROPIC-D43].

Conversely, in the second case (b) (decision on the SCM) also information from the small cell(s) is taken into account and the key performance indicator is whether all the users are globally getting a good service. This type of offloading framework requires several optimization algorithms on both the cloud management side and radio management side (see Section 5).

The two techniques are radically different from the point of view of the networking overhead. The first offloading technique was already implemented and tested within the framework of 4A3, and is now being enhanced with data-oriented partitioning techniques that will be presented later on in this document. We are considering implementing the second offloading technique within the framework of 6A2 and comparing the two approaches experimentally.

The "energy consumption" sub block provides an estimate of the energy consumption associated to each possible offloading strategy. Such estimation includes the energy spent for the communication with the serving SCeNBce and the energy spent for the computation performed at the UE.

The offloading framework is composed by the set of modules that are directly involved in the process of decision taking in relation with the offloading. In that sense, such modules identify the profiles of the application to be offloaded potentially, the user equipment, and the network. In addition, estimation of the energy consumption and computation power are obtained. Finally, this framework also contains the modules to initiate and execute the offloading

4.3 *Description of interfaces between modules*

Z Protocol messages are only intended for communications between different devices. Modules within the same device do not have dedicated Z Protocol messages. The Z Protocol messages are described in [TROPIC – D42].

4.3.1 **VM management – context vector**

The VM management module in the SCM interfaces with the context vector in order to store and query the necessary parameters for the VM lifecycle management.

4.3.2 **VM management – Remote Service Management**

The VM management module in the SCM interfaces with the remote service management in the SCs to communicate and receive the Z Protocol messages intended for the VM lifecycle management. I.e. when the SCM sends a DEPLOY message, it's the remote service management module the one that attends it.

4.3.3 **Proactive monitoring – Context vector**

The proactive monitoring module interfaces with the context vector to store the value of the parameters that are being monitored. This interface goes through other components located in the SCM, such as the server and the process that attend the proactive monitoring messages.

4.3.4 **Reactive monitoring – Context vector**

The reactive monitoring module interfaces with the context vector for two possible actions:

- To retrieve the information about the SCs to poll, and the parameter details
- To store the received monitoring information

4.3.5 **Proactive monitoring – Remote service management**

The proactive monitoring module interfaces with the remote service management module to retrieve the parameters from the VMs, since the only component that handles and has access to the hypervisor and VMs is the Remote Service Management module.

4.3.6 **UE service manager – service management**

During offloading, the Execution Controller component of the offloading framework handles the follow: a) Interrupt the local execution of a module; b) send to the femto-cloud side the necessary input together with the specifics on which application and which module (within the application) to execute; c) wait for the output of the remote execution to end and return the execution control to the application as if the module was executed locally. The UE service manager acts as a middleware between the Execution controller component of the offloading framework and the femto-cloud side. It makes possible to allocate and access the primary VM of the user from the UE point of view, as well as guarantees that the offloading process is done properly.

4.3.7 **Remote service management**

It is the counter-part of the UE service manager. In addition, its task is to take care of the remote execution of a given app module, as specified by the Execution Controller component and communicated by the UE service manager.

4.3.8 **Registration (UE) – Cloud registration (SC)**

The registration module in the UE interfaces with the cloud registration module in the SC to handle the connection of new users to the SC. This involves also the interface with the cloud registration module in the SCM described in the following section.

4.3.9 **Cloud registration (SC) – Cloud registration (SCM)**

The cloud registration module in the SC interfaces with the cloud registration module in the SCM to convey the Z Protocol messages regarding the connection of the SC when it switches on and the messages for the connection of new users. The interchanged Z Protocol messages along with the description of the different connection processes are explained in [TROPIC-D52].

4.4 ***Architecture of SCC***

This section provides summary of the architecture defined by the project. The proposed architecture is based on the version drafted in [TROPIC_D22] but with respect to the original version, it is aligned and updated according to the needs arisen due to the project progress in WP3, WP4, and WP5. Therefore, the architecture presented in this section is also the final architecture delivered to WP6 for implementation.

Based on [TROPIC_D22], we consider centralized approach but we also take into account possible future extension towards advanced decentralized approaches in the design of the final architecture. During the project, two possible options of final architecture have been investigated. They differ to each other in the implementation of communication among entities and translation of protocols. The first option is compatible with existing architecture but it needs enhancements by means of Small Cell Cloud Gateway (SCC-GW) into SCeNBces. The SCC-GW identifies and splits SCC traffic from conventional traffic to avoid forwarding of all traffic to P-GW as in conventional LTE-A network. The second option requires no HW modification of the small cell base station (except extension with computing /storage capabilities) but it changes the way of addressing equipments over radio. Both architectures are addressed in following subsections.

4.4.1 Consolidated SCC architecture with protocol translation

Figure 56 depicts the overall view of the SCC architecture showing the LTE and SCC components and their interfaces. The RAN includes:

- SceNBce, the computationally-enhanced SC
- SCeNB, a regular legacy Small Cell eNB

The Core Network includes:

- S-GW – a mobility anchor which transfers the user traffic from the old serving base station to the new one upon radio handover
- P-GW – a gateway to external Packet Data Networks (PDN)
- MME – entity, which commands the SGW.

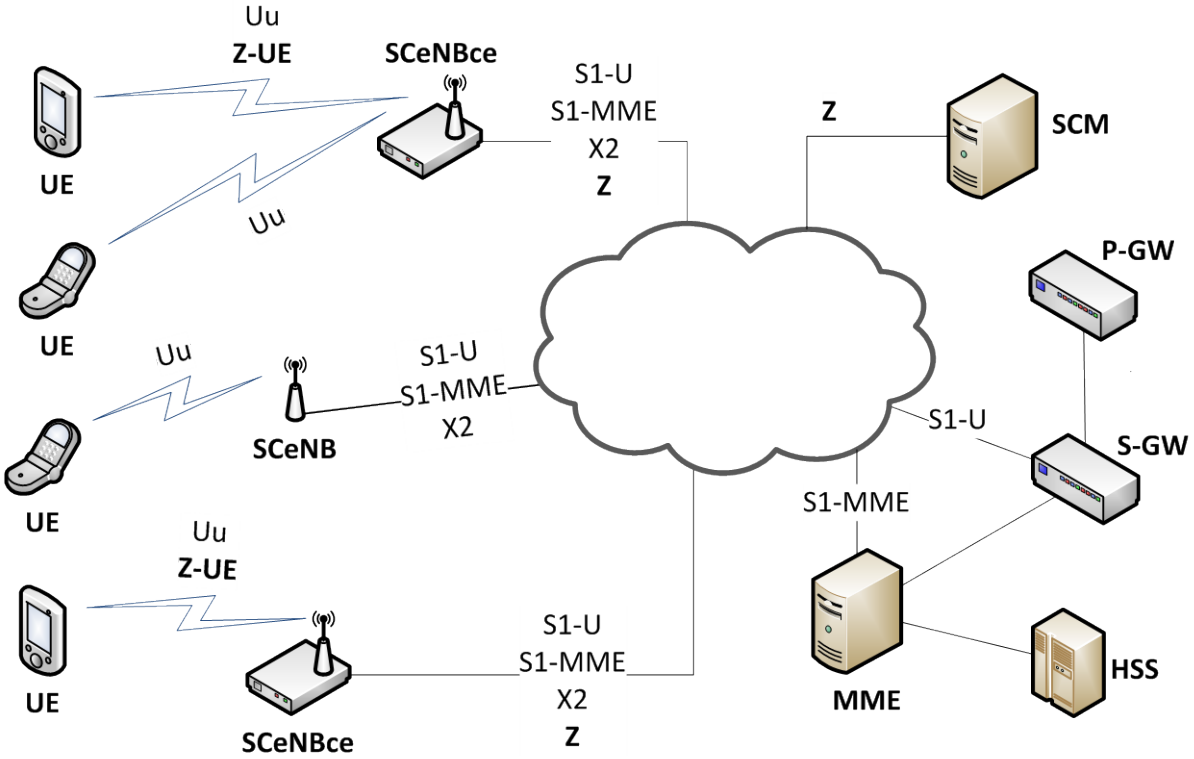


Figure 56. General SCC architecture (defined in [TROPIC_D22]).

The SCeNB and SCeNBce communicate with the MME over the S1-MME control interface using SCTP; and with the S-GW over the S1-U interface, on which the user data is encapsulated over GTP. The SCM communicates with the SCeNBces (and vice-versa) over the Z control interface making use of the Z Protocol. The UE communicates with the base station through the radio channel using the Uu and Z-UE interfaces for the LTE and SCC communications, respectively. Communications between SCeNBces are transmitted over the Internet.

4.4.1.1 *Assumed SCeNBce Hardware Configuration*

The HW configuration of a typical Small Cell (SCeNB) [Lin11] and the HW configuration considered for the SCeNBce are depicted in Figure 57. We have chosen this configuration since we consider it is the most generic for having signal processing and computing capabilities in the same device. The baseband processor (BP), which is in charge of signal processing, is placed as a peripheral following a single system bus architecture (multidrop topology). The BP connects to the signal modulator and demodulator which in turn connects to the antenna. The BP chipset holds its own memory and runs a real-time operating system to cope with the radio signal processing requirements. The DMA (Direct Memory Access) controller allows direct read/write access to the main memory without occupying CPU cycles; i.e. the CPU initiates the data transfer from the BP to the memory, carries out other tasks

while the transfer is in progress, and receives an interrupt from the DMA controller when the transfer is done. This operation is equivalent to the DMA controller of the Network Interface Card (NIC), which connects to the backhaul.

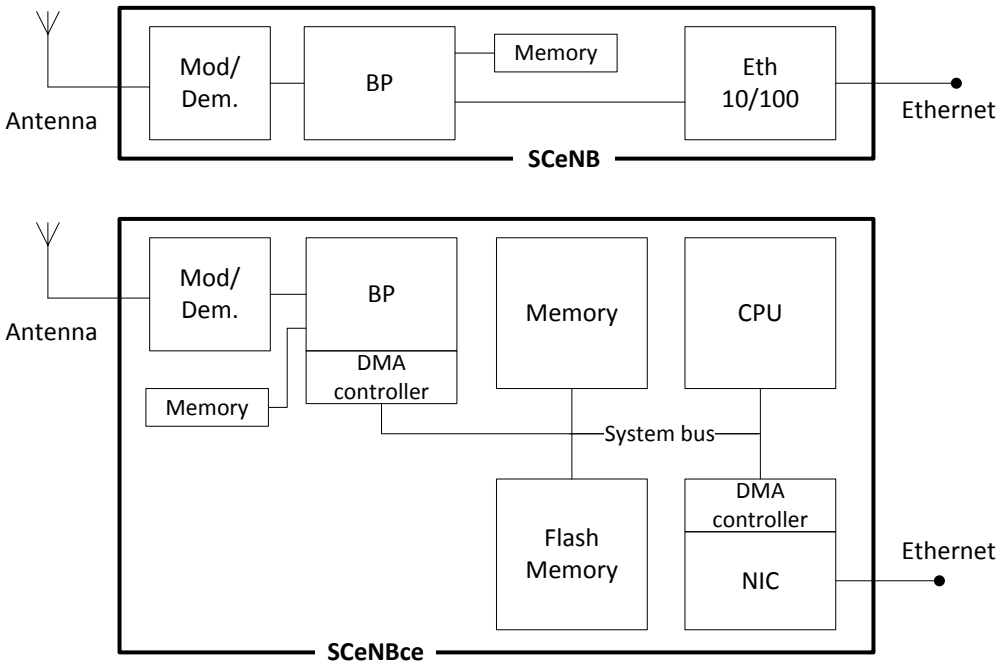


Figure 57. SCeNBce HW configuration.

This new configuration introduces additional functionalities to the base station. Two computing entities, the BP and the CPU, are in charge of the traffic processing load and the SCC related computing tasks, respectively. Particularly, the CPU will be in charge of certain traffic processing tasks that are part of the BP functionality in regular base stations.

Since processing traffic in the CPU involves delay overhead that might not be negligible, it can be argued that the traffic processing load could be done by an additional network processor. We have opted not to consider additional processing hardware due to the following reasons: (i) we assume that a HW architecture dedicated to that purpose will have its operation adapted to minimize the overhead, i.e. attending BP and NIC interrupts with maximum priority (thus handling the traffic as soon as it arrives at the SCeNBce) and using DMA whenever possible (thus minimizing computing resources dedicated to traffic aspects); (ii) introducing additional processing hardware increases cost and complexity; (iii) recent developments of network "softwarization" widely adopted by network operators, such as NFV [NFV12] and SDN [ONF12], call for the usage of general purpose hardware for deploying network functionality and the elimination of physical middle-boxes (NAT, firewalls, etc.), which has clear benefits on scalability, programmability, customizability, costs, Time to Market, service deployment, etc.; (iv) supporting the previous point, companies such as Intel are already thinking of this new scenario and preparing new general-purpose HW processors that have the performance to process packets and run multiple applications simultaneously [ONF12]; (v) Business Systems Support functions (such as billing) may need to see the traffic being transmitted, which will force the traffic to be redirected to where those functions reside, i.e. the system's CPU, or will involve traffic monitoring and therefore additional overhead; and (vi) handling the new traffic by means of additional processing hardware does not eliminate the overhead, and some performance degradation of the LTE traffic when compared to typical SCeNBs will occur anyway.

All this has motivated us to consider the placement of the novel SCC network functionality as software elements executed in the SCeNBce's CPU rather than in dedicated network processors. We do not consider the elimination of the BP and consider it as a necessary separate component due to the tight real-time requirements of the radio signal processing [TROPIC_D53].

4.4.1.2 *SCC-LTE integration*

Two kinds of traffics can be differentiated within the SCC:

- LTE traffic, either internal traversing the S-GW or external traversing both S-GW and P-GW
- SCC traffic among SCeNBces and SCM, which do not traverse the EPC. For the SCC traffic to be offloaded to the Internet without traversing the Core Network a traffic breakout mechanism in the SCeNBces is needed, i.e. a mechanism that identifies and segregates LTE and SCC traffics.

4.4.1.2.1 LTE-SCC traffic segregation

The UE can communicate with an external entity by means of communications through the P-GW, but this would imply a significant delay overhead. Offloading the SCC traffic to the Internet aims to maintain the delay overhead to a minimum while reducing the Core Network's load since no GTP tunnels are needed between EPC components.
Technologies addressing LTE traffic offload are for example LIPA and SIPTO [3GPP TR23.829]. They offload LTE traffic to a local network (LAN) and to the Internet, respectively. Both are based on the collocation of a Local Gateway (L-GW) either on the eNB or separated from it. We have not considered the utilization of these solutions since the inclusion of the L-GW is not transparent to the LTE architecture and additional functionality must be added to the EPC components [3GPP TR36.300], thus impacting the existing LTE architecture.
From a high level point of view, the segregation functionality consists of the following steps: (1) The BP processes the radio signal and de-encapsulates the physical and link layers; (2) the data is inspected to identify whether it's LTE or SCC traffic; (3) the data is encapsulated accordingly for each kind of traffic and (4) handed over the NIC to be delivered through the backhaul. As traffic processing tasks, (2) and (3) can be carried out either by the BP or the CPU, while (4) is launched by the CPU, which handles the connections with the NIC, and then it's completed using DMA.
The overall goal is to encapsulate and transmit the LTE traffic over the S1-U or S1-MME interfaces, thus being transmitted to the EPC; and encapsulate the SCC traffic over the typical Internet stack, thus being offloaded to the Internet.

4.4.1.2.2 Radio channel/UE identification for SCC traffic

The segregation functionality takes place for communications from the UE. In the same way, for communications to the UE, the SCC and LTE traffics (arriving to the base station from different sources and with different protocol stacks) shall be delivered to the corresponding UE appropriately.
In LTE, the traffic is transmitted though pre-established EPS bearers [3GPP TS24.301] between UE and P-GW. Each UE has at least one established *default bearer* with nominal QoS established during the UE Attach process. Additionally, it may have further default bearers (on a PDN connection basis) and *dedicated bearers* (with different QoS requirements). These bearers are not IP based, i.e. the routing of the traffic along the P-GW – S-GW – BS – UE path is not based on the UE IP address, but rather on GTP tunnels from P-GW to BS (S1/S5/S8 bearers) and explicit radio resource allocation from BS to UE (radio bearers and logical channels). The mapping between S1 bearers (GTP tunnels) and radio bearers is done based on the TEID (Tunnel Endpoint ID) transmitted in the GTP-U header. The TEID is used to de-multiplex traffic incoming from remote tunnel endpoints so that it is delivered to the User plane entities in a way that allows multiplexing of different users [3GPP TS 29.281].
In the SCC, the LTE traffic handling does not require additional functionality since the GTP and S1-AP packets are delivered from the SCC-GW directly to the BP, which handles them following the typical operation in current base stations. The data is transmitted though the established EPS bearers like in regular base stations. On the contrary, the SCC traffic needs special treatment. The IP packets arriving to a SCeNBce shall be delivered to a specific UE, but the only information that these packets contain is the UE IP address, which makes impossible for a regular base station to identify the destination UE, since regular base stations do not see nor handle the IP layer. Therefore, a mechanism to map UE IP addresses to radio bearers/logical channels (i.e. to TEID) is needed at the SCeNBce.
All the established bearers have their associate TEID at the base station, but now the question is through which of them send the user SCC traffic. The immediate answer is to send the traffic through the first stablished default bearer because (i) it is always present as long as the UE is connected; (ii) it

minimizes the impact on LTE QoS sensible traffic, since it has nominal QoS constraints and the other EPS bearers transporting QoS sensible traffic such as voice calls, etc. do not become affected; and (iii) the necessary data to carry out the mapping (i.e. UE IP and TEID) can be obtained upon the UE Attach procedure.
Figure 58 and Figure 59 depict the mapping of IDs and addresses for LTE traffic and SCC traffic, respectively.

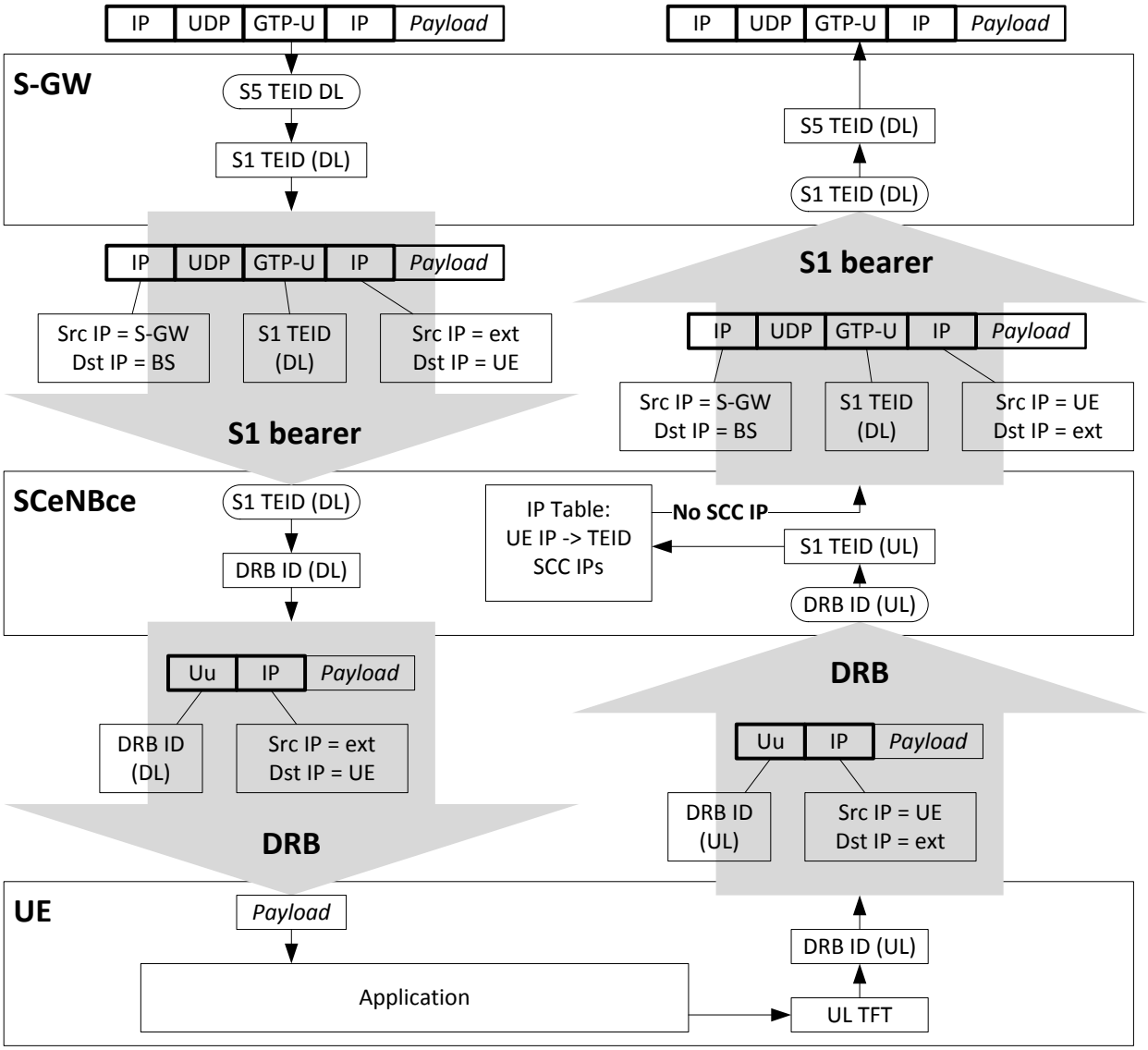


Figure 58. LTE traffic mapping.

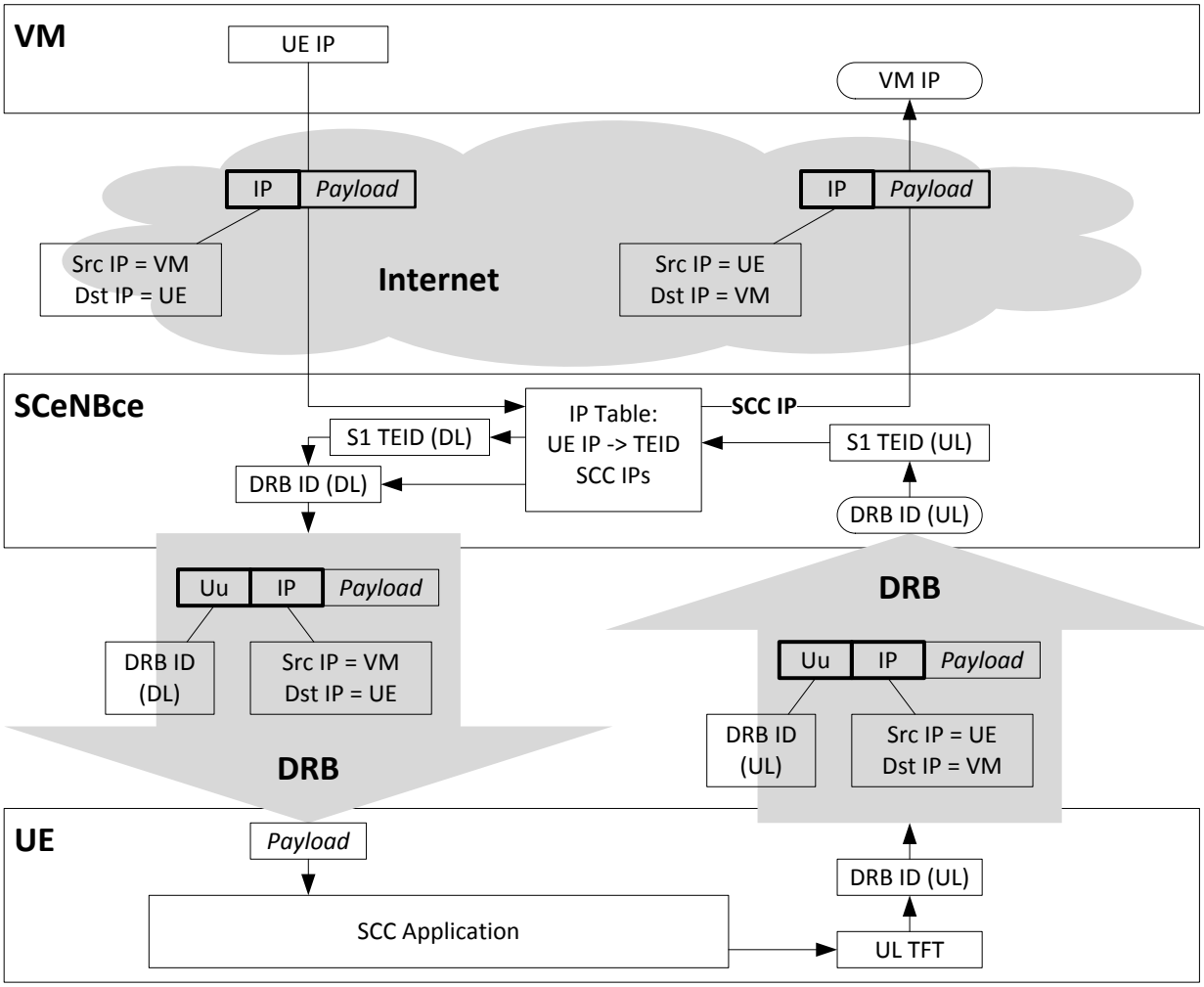


Figure 59. SCC traffic mapping.

4.4.1.3 *SCC protocols and blocks*

Figure 60 depicts the overview of the hardware and software components that form the SCC along with their interfaces and protocol stacks used in the information exchange. The physical nodes, highlighted in light grey, are the UE, SCeNBce and the EPC components MME, S-GW and P-GW. Highlighted in dark grey are the individual components relevant for the SCC operation being the hardware part composed of the BP, modulator/demodulator, antenna and the NIC. The SCM, although presented so far as a standalone component, can be understood as a software entity that can be placed in either a dedicated physical device or a shared one.

The SCC-GW (Small-Cell Cloud Gateway) is the component, which is in charge of executing the traffic segregation functionality, following the aforementioned "network softwarization" tendency. As depicted, it holds an IP Table storing the UE IP-TEID mapping along with the MME IP and the set of IP addresses belonging to the Cloud entities. The SCC-GW and BP exchange S1-AP packets for the LTE-A control plane and GTP-U packets for the LTE-A user plane and cloud traffic. In turn, the BP communicates to the SCC-GW the UE IP-TEID mappings upon the initial default bearer establishment.

The SCC management is carried out by the SCM in cooperation with SCM-BS (SCM Base Station) and SCM-UE (SCM User Equipment):

- The SCM is a part of the operator's network. Located in a standalone way and reachable by its IP address (SCM IP). It manages, monitors and optimizes the system operation. The SCM communicates with the SCM-BS and the SCM-UE (SCM User Equipment).

- The SCM-BS is located in each SCeNBce. It gathers information related to radio environment (UE, multi-cell) and backhaul parameters (latency, jitter, etc.). It includes the interaction with the hypervisor to manage the Virtual Machines operating on the SCeNBce.
- The SCM-UE is located in the UE. It gathers the status of the mobile device (e.g., UE battery), handles QoE monitoring and communicates with the SCM-BS.

The role of both SCM-UE/BS is to deliver of local management functionality at the SCeNBce and UE to improve efficiency and scalability. For the SCC operation the most relevant management action is the update of the IP table with the IP addresses of the cloud entities. To that end, upon a new SCeNBce connection the central SCM distributes the IP address to the SCM-BS in other SCeNBces, which in turn update their IP table locally.

The SCM and SCM-BS communicate through the Z interface while the SCM-BS and SCM-UE communicate through the Z-UE interface. The Z and Z-UE interfaces use the Z Protocol, which is an application protocol that defines the exchange of control messages. Actions described by Z Protocol messages are, for example, deployment/destruction of VMs, monitoring messages, updates of the IP tables and all the other management actions needed for the SCC operation.

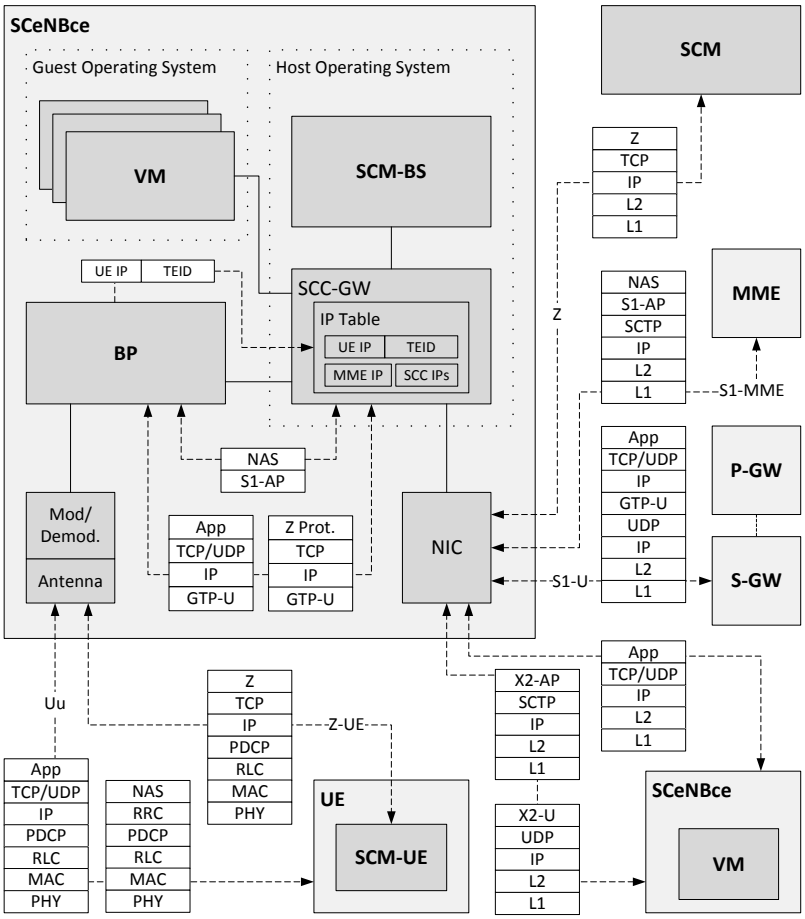


Figure 60. Modules in SCC architecture.

The boxes depicted within the elements indicate the protocols and interfaces they handle. Figure 60 depicts the overview of the hardware and software components that form the SCC along with their interfaces and protocol stacks used in the information exchange. The physical nodes, highlighted in light grey, are the UE, SCeNBce and the EPC components MME, S-GW and P-GW. Highlighted in dark grey are the individual components relevant for the SCC operation being the hardware part composed of the BP, modulator/demodulator, antenna and the NIC. The SCM, although presented so

far as a standalone component, can be understood as a software entity that can be placed in either a dedicated physical device or a shared one.

The SCC-GW (Small-Cell Cloud Gateway) is the component which is in charge of executing the traffic segregation functionality, following the aforementioned “network softwarization” tendency. As depicted, it holds an IP Table storing the UE IP-TEID mapping along with the MME IP and the set of IP addresses belonging to the Cloud entities. The SCC-GW and BP exchange S1-AP packets for the LTE-A control plane and GTP-U packets for the LTE-A user plane and cloud traffic. In turn, the BP communicates to the SCC-GW the UE IP-TEID mappings upon the initial default bearer establishment.

The SCC management is carried out by three entities: the already introduced SCM, the SCM-BS (SCM Base Station) and the SCM-UE (SCM User Equipment). Their role is to perform the already described SCM functionality in a per component basis, allowing the placement of local management functionality at the SCeNBce and UE to improve efficiency and scalability. For the SCC operation the most relevant management action is the update of the IP table with the IP addresses of the cloud entities. To that end, upon a new SCeNBce connection the central SCM distributes the IP address to the SCM-BS in other SCeNBces, which in turn update their IP table locally.

The SCM and SCM-BS communicate through the Z interface while the SCM-BS and SCM-UE communicate through the Z-UE interface. The Z and Z-UE interfaces use the Z Protocol, which is an application protocol that defines the exchange of control messages. Actions described by Z Protocol messages are for example deployment/destruction of VMs, monitoring messages, updates of the IP tables and all the other management actions needed for the SCC operation.

4.4.2 SCC architecture with new addressing

4.4.2.1 *Relevant Entities*

In Figure 61 is shown the detailed 3GPP LTE network architecture, including the protocol stacks.
The architecture includes the following entities which communicate with each-other at different levels SCM-UE, SCM-BS, and SCM as shown in Figure 61.

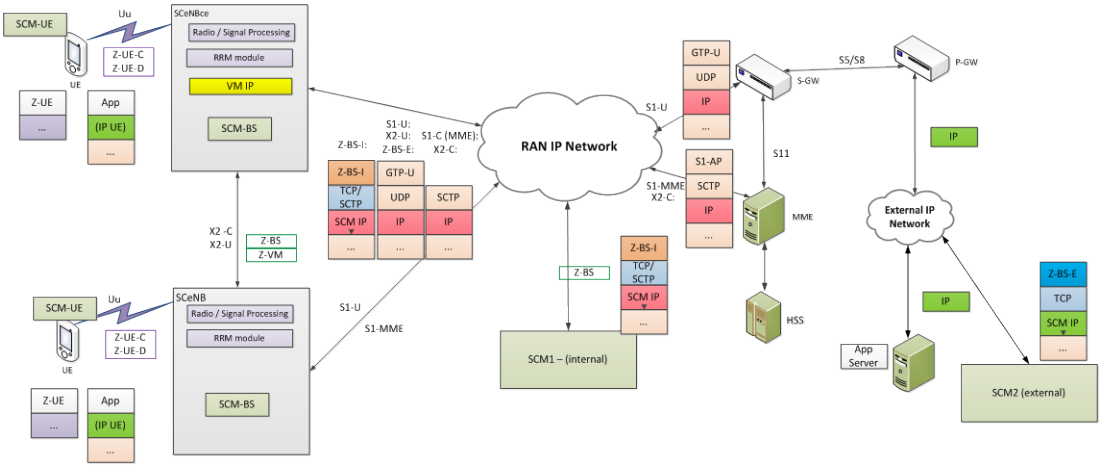


Figure 61. System architecture and protocol stacks.

4.4.2.2 *IP addressing*

A first observation is that there are two IP domains:

1. An external IP domain, where are located in general the application servers or the WWW domain. The access to the external IP domain is through the PDN (Packet Data Network) GW, named also P-GW.
2. An internal IP domain, in which the eNBs communicate based on an IP address allocated within the internal IP domain. The relevant LTE interfaces using the internal IP domain are the S1 and X2 interfaces. The S1 interface is using the GTP-U protocol for tunneling user data from an UE to the S-GW/PDN GW. The tunnel is created between the serving eNB and the PDN GW. This means that an SCM located in the external IP domain can speak with an UE, based on the UE IP address, but an SCM located in an eNB or generally in the internal IP domain cannot speak with an UE.

A second observation is that an eNB does not communicate with an UE based on UE IP address. Currently, for supporting the hand-over process, an eNB can obtain UE-specific data, based on temporary US identifiers, from the serving eNB in two ways:

- Over the X2 interface;
- Through a transparent container over the S1 interface.

4.4.2.2.1 SCM situated in the external IP domain (SCM-E)

From the above situation it results that for having an eNB communicating with an SCM in the external IP domain is needed to include the Z-interface traffic within a pre-defined GTP tunnel, where the Z-interface traffic is multiplexed together with the UE traffic. There are three disadvantages related to this approach:

- The signaling associated with the Z-protocol is transmitted in an un-reliable mode (over UDP);
- The incurred delays are relatively high;
- Does not resolve the UE communication with VMs located on eNBs within RAN.

However an advantage of this approach is that the application and the SCM can communicate in the common external IP domain.

A detailed view of the usage and GTP headers is given in TS29.281. We reproduce some essential definitions and their relevance to the Z-protocol transport.

4.4.2.2.2 Details of GTP usage

GTP is defined for use in cellular networks.

GTP-U (user plane) messages are either user plane messages or signaling messages. User plane messages are used to carry user data packets between GTP-U entities. Signaling messages are sent between network nodes for path management and tunnel management.

A GTP-U tunnel is identified in each node with a TEID, an IP address and a UDP port number. A GTP-U tunnel enables forwarding packets between GTP-U entities.

Common Tunnel Endpoint Identifier (C-TEID) unambiguously identifies a tunnel endpoint in the receiving GTP-U protocol entity for a given UDP/IP endpoint. The sending end side of a GTP tunnel locally assigns the C-TEID value used in the TEID field and signals it to the destination Tunnel Endpoint using a control plane message.

4.4.2.2.3 SCM situated in the RAN internal IP domain (SCM-I)

In this case the SCM function is either embedded within a stand-alone computer within the Operator network or within a base station, which can be also a SCeNBce. This option presents a number of advantages:

- The protocol stack can include reliable transportation, for example by using SCTP, as in the case of the control plane of X2 protocol (SCTP involves multi-hosting functionality) or by simply using TCP;
- SCM placement within the RAN network ensures the lowest delays;
- No encapsulation over GTP is needed.

This option is the preferred one and will be considered in continuation.

4.4.2.3 *Communication paths for offloaded applications*

For generality, let's consider that the UE which has offloaded an application is radio-served by a SCeNB with no computing capabilities. The execution of the application is offloaded to a SCeNBce which has no radio connectivity with the UE. Note that the approach presented below is applicable for any type of cellular base stations or RNC, as well as for Access Points using wireless technologies or by network elements, such as Routers.

For exchanging user data, the application on the UE has to receive/transmit traffic from/to another eNB, i.e. to communicate with an entity within RAN based on the IP Address on that entity. This implies that not all the UE traffic is encapsulated on the existing bearers. Depending on the application status (offloaded or not) some of the traffic is diverted within RAN.

Given that within RAN the UE is not identified based on its IP address, and both on data plane and control plane of non-radio protocols is needed the communication with some entities not belonging to the radio network, it may be needed a differentiation between the UE traffic within the RAN and the traffic to/from S-GW.

4.4.2.4 *New data and control paths within RAN*

As explain before, the executing virtual machine becomes the central point of user traffic for an offloaded application within RAN.
Figure 62 illustrates the new data paths.

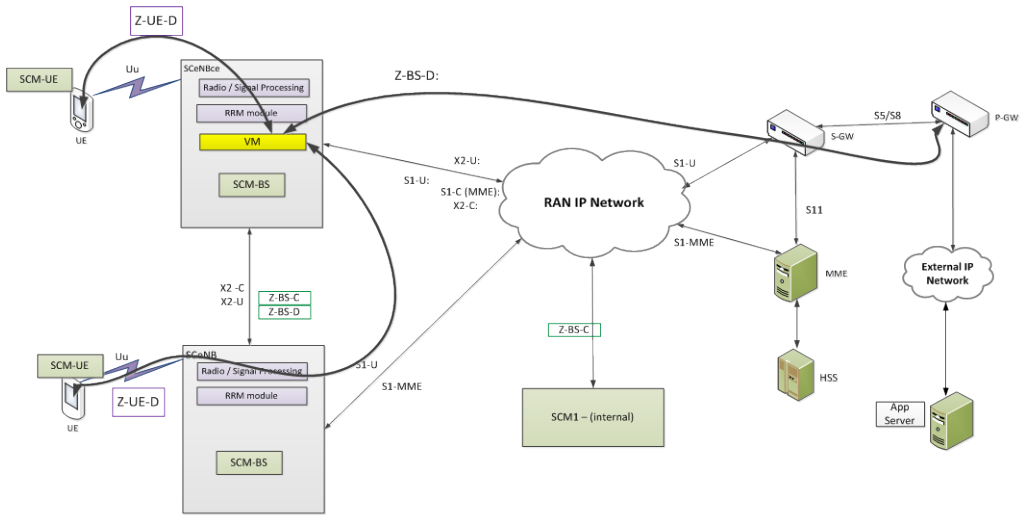


Figure 62. New data paths within RAN.

As discussed above, the eNB or the platform where the VM running the application is placed become the central point of processing user data. The VM communicates on one side with the S-GW, using the protocol interface Z-BS-D, and on the other side with the application on UE using the protocol interface Z-UE-D. This last communication channel goes through the radio serving base station; its IP address and the UE S-TMSI (Temporary Mobile Subscriber Identity including the MME Code) shall

be known by the Virtual Machine on which the application is running. When the UE is identified by its IP address and/or Ethernet address, these can be also used as identifier.

For the control plane, the SCM becomes the central point of communication with the control module in UE (SCM-UE) and the control module in eNB (SCM-BS). The respective protocol interfaces are named Z-UE-C and Z-BS-C.

Based on this new architecture, it results that the radio serving data path and the application serving data path are on different segments and they should be treated separately.

For the control plane, the SCM becomes the central point of communication with the control module in UE (SCM-UE) and the control module in eNB (SCM-BS). The respective protocol interfaces are named Z-UE-C and Z-BS-C. The resulting communication paths are presented in Figure 63.

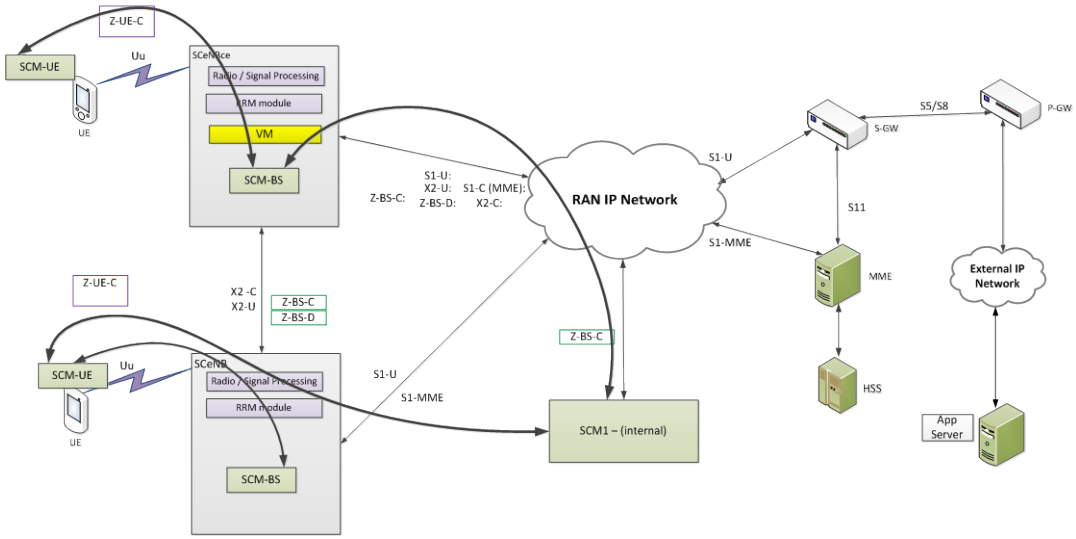


Figure 63. New control paths within RAN.

4.4.2.5 *Mobility aspects*

Mobile users can change their serving eNB during movement or for achieving load balancing. While in the traditional cloud concept the user handover from eNB to eNB is transparent to the cloud server, this is not the case when the application is executed in an eNB, because the MME will command the S-GW to direct the user traffic to the radio serving eNB.

While for the existing architecture the S-GW will direct the traffic to the new radio serving eNB, for the offloaded applications may be preferable to continue the execution on the already allocated Virtual Machines. On one side, this puts a new requirement on the MME on being application and computing platform aware and on the other side the computing platform needs to be aware of the IP address of the new radio-serving eNB.

4.4.2.6 *Multiple applications*

Given that not all the applications will be offloaded, a different treatment of user data shall be in place for offloaded and not-offloaded applications, such that the data of non-offloaded applications will be sent as before to the S/P GW, while the data for offloaded applications will go to the Virtual Machine.

4.4.2.7 *Z-protocol bearer architecture*

The bearer architecture of Z-protocol, as resulted from the above discussion, is shown in the next figure:

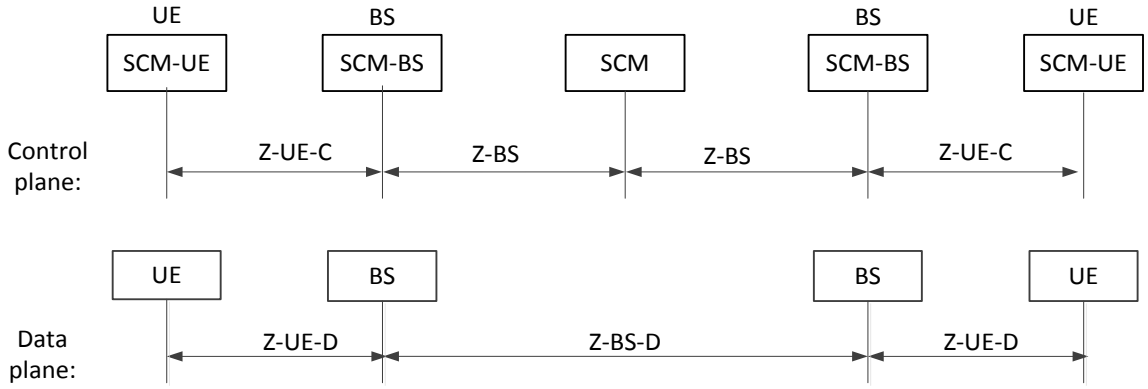


Figure 64. Communication channels supporting Z-protocol.

SCM-UE (Small Cell Manager – UE part) communicates with its pair SCM-BS located in the serving eNB using a Control Logical Channel (Z-UE-C) and a Data Logical Channel (Z-UE-D).

SCM-BS, located within SCeNBce or SCeNB(ce), communicates with a SCM, for exchanging control information, using the Z-SCM interface; this interface can be used in a stand-alone mode or its messages may be integrated partially or totally into the LTE X2 interface. In case that the SCM is integrated in an eNB, for that eNB the communication may be internal without using the IP addressing. SCM interface should rely on TCP or SCTP reliable transmission protocols.

The SCM exchange control information with Hypervisors of the VMs located in an eNB using the Z-HIP high level interface. The messages for this part of the Z-protocol were defined in WP4.

An eNB serving an UE which wants to offload the execution of an application communicates with the assigned VM for the application by using the GTP encapsulation over the Z-VM interface.

4.4.2.8 *Ports*

All Z-protocol communication channels can use IP addressing. It is assumed that a TCP / SCTP port will be allocated for control part and a UDP / TCP port will be allocated for data part.

4.4.2.9 *Transport of the Z-protocol over the Uu interface*

The transport of Z-protocol over the Uu interface involves using a radio bearer (logical channel) which is provisioned or allocated to eNB services, instead of being dynamically allocated by MME or UE. The *priority* of this logical channel should be relatively high, which means a distribution of the existing priority order. The assignment of the priority for Z-protocol communication should be provisioned at eNB and communicated to UEs.

4.4.2.10 *Radio bearers based on 3GPP specifications*

The logical channels for the transport of the Z-protocol shall integrate into the existing radio bearer configuration (Release 12).
From the study of TS36.331, TS.36321 and TS 36.508 it results that there are 5 bits in the MAC header allocated for LCID (section 6.1.2 in TS 36.331), from which are already assigned 11 logical channels, which are allocated for signaling and data as follows:

- LCID 0,1,2 are respectively allocated to SRB0, SRB1, SRB2
- LCID 3-10 are allocated to DRBs (Data Radio Bearers); these LCIDs correspond 1:1 to the E-RABs established by the MME.
- LCD 11-15 are RESERVED for use as identity of logical channel.

The LCIDs > 15 are partially used for commands. In this group, a number of LCID are not yet assigned. The priority of these radio bearers goes from the highest for the signaling bearers to the lowest for the last LCID.

These reserved LCIDs can be used for both Z-UE-D and Z-UE-C channels.

It should be noted that the LCID "0" is allocated for the establishment of the initial RRC (Radio Resource Control) connection between eNB and UE and is not used anymore afterwards.

For understanding the logical channel and bearer concept, we reproduce the Layer 2 architecture, for DL and UL, from TS36.300. It can be observed that for each bearer of the higher layers is assigned a special processing resource and a special Logical Channel.

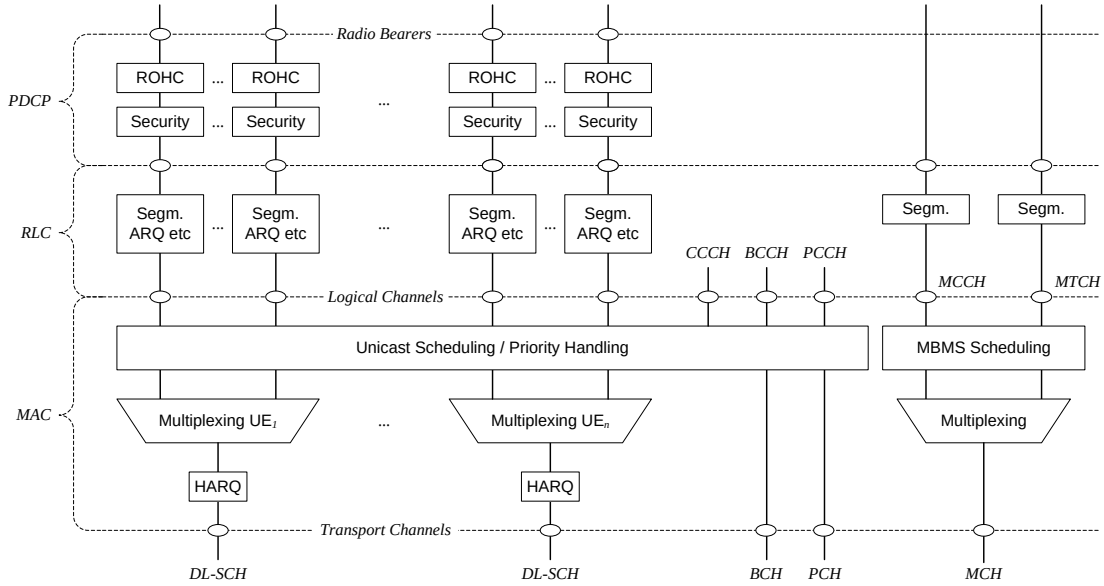


Figure 65. Layer 2 structure for DL (from TS36.300).

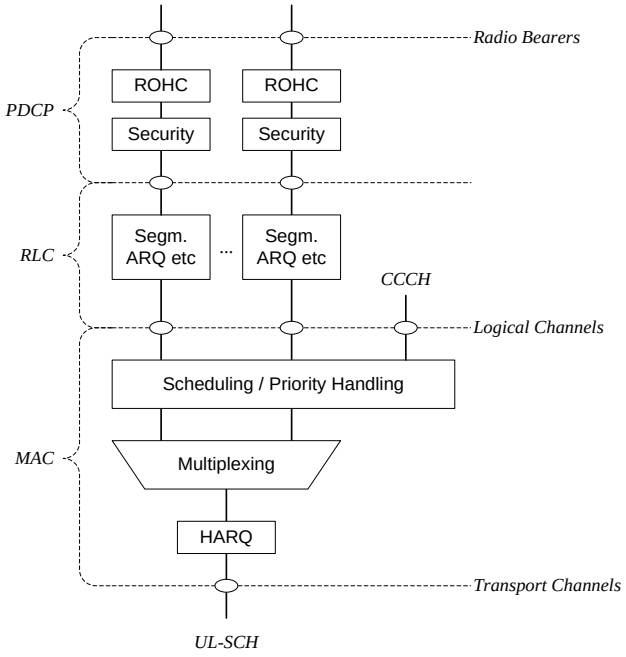


Figure 66. Layer 2 structure for UL (from TS36.300).

We recommend that the initial Logical channel setting (a Layer 2 function) will differentiate between three types of traffic:

1. Signaling to be sent or received to/from MME; this traffic uses LCID 0,1,2, as shown above;
2. Non-3GPP traffic to be encapsulated and to be sent or received to/from S-GW; this traffic uses LCID 3…10
3. Non-3GPP signaling to remain in radio serving eNB; LCID(s) to be assigned in future 3GPP releases. No IP encapsulation of the protocol payload is needed.
4. Non-3GPP signaling to remain in RAN; LCID(s) to be assigned in future 3GPP releases. The payload may be transmitted by the same procedures as those used for X2 interface.
5. Non-3GPP user data to remain in eNB; No IP encapsulation of the protocol payload is mandatory needed, however a header indicating the eNB entity to handle the data is needed. So SCM-BS will be assigned a specific value in this header.
6. Non-3GPP user data to remain in RAN. The payload may be transmitted by the same procedures as those used for X2-U interface, eventually using GTP-U encapsulation.

4.4.2.11 *Z-UE protocol transport support by LTE*

This protocol, running between the SCM-UE and the SCM-BS, belongs to NAS (Non-Access Stratum) family of protocols.

4.4.2.12 *Transport of NAS and non-3GPP information transfer*

NAS (Non-Access Stratum) information includes the signaling between UE and MME and the transfer of non-3GPP information, as control/data information related to Z protocol. Below are detailed the supporting functions in TS36.331.

4.4.2.12.1 DL information transfer

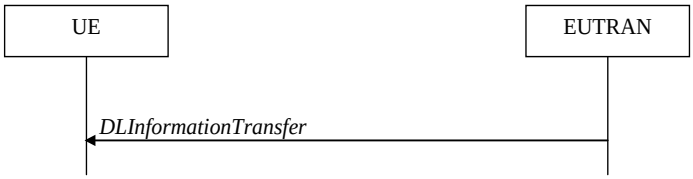


Figure 67. DL Information transfer in TS36.331.

The purpose of this procedure is to transfer NAS or (tunneled) non-3GPP dedicated information from E-UTRAN to a UE in RRC_CONNECTED.

Initiation

E-UTRAN initiates the DL information transfer procedure whenever there is a need to transfer NAS or non-3GPP dedicated information. E-UTRAN initiates the DL information transfer procedure by sending the *DLInformationTransfer* message.

Reception of the DLInformationTransfer by the UE

Upon receiving *DLInformationTransfer* message, the UE shall:

1> if the *dedicatedInfoType* is set to *dedicatedInfoNAS*:

2> forward the *dedicatedInfoNAS* to the NAS upper layers.

4.4.2.12.2 UL information transfer

General

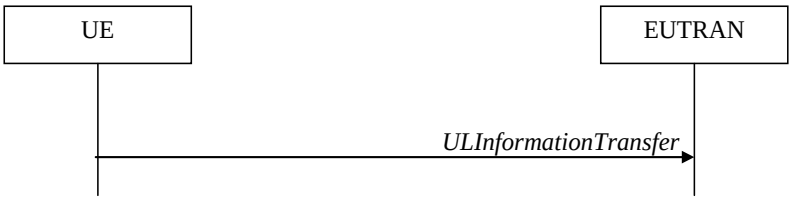


Figure 68. UL Information transfer in TS36.331.

The purpose of this procedure is to transfer NAS or (tunneled) non-3GPP dedicated information from the UE to E-UTRAN.

Initiation

A UE in RRC_CONNECTED initiates the UL information transfer procedure whenever there is a need to transfer NAS or non-3GPP dedicated information, except at RRC connection establishment in which case the NAS information is piggybacked to the *RRCConnectionSetupComplete* message. The UE initiates the UL information transfer procedure by sending the *ULInformationTransfer* message.

Actions related to transmission of ULInformationTransfer message

The UE shall set the contents of the *ULInformationTransfer* message as follows:

1> if there is a need to transfer NAS information:

set the *dedicatedInfoType* to include the *dedicatedInfoNAS*;

submit the *ULInformationTransfer* message to lower layers for transmission, upon which the procedure ends;

Failure to deliver ULInformationTransfer message

The UE shall:

1> if mobility (i.e. handover, RRC connection re-establishment) occurs before the successful delivery of *ULInformationTransfer* messages has been confirmed by lower layers:

2> inform upper layers about the possible failure to deliver the information contained in the concerned *ULInformationTransfer* messages;

4.4.2.13 *Z-BS Data Plane*

The transport layer for data streams over Z-BS-D is an IP based Transport. The following figure shows the transport protocol stacks.

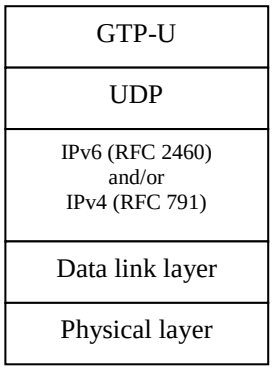


Figure 69. Transport network layer for data streams over Z-BS-U.

The GTP-U (3GPP TS 29.281) protocol over UDP over IP shall be supported as the transport for data streams on the Z-BS-D interface.
There may be zero or one UL data stream and zero or one DL data stream per bearer at the Z-BS-D interface.

- The DL data stream is used for DL data forwarding from the source eNB to the target eNB.
- The UL data stream is used for UL data forwarding from the source eNB to the target eNB.

Each data stream is carried on a dedicated transport bearer.
The identity of a transport bearer of the IP address and the TEID of the corresponding GTP tunnel, allocated by the target eNB (see TS 29.281).

4.4.2.14 *UDP/IP*

The path protocol used shall be UDP (IETF RFC 768 [3]).
The UDP port number for GTP-U should be defined for the Z-protocol.
The eNBs over the Z-BS-D interface shall support fragmentation and assembly of GTP packets at the IP layer.
The eNB shall support IPv6 (IETF RFC 2460) and/or IPv4 (IETF RFC 791).

4.4.2.15 *Diffserv code point marking*

IP Differentiated Services code point marking (IETF RFC 2474) shall be supported. The mapping between traffic categories and Diffserv code points shall be configurable by O&M for based on QoS Class Identifier (QCI)/ Label Characteristics. Traffic categories are implementation-specific and may be determined from the application parameters.

4.5 *Z-Protocol for SCC*

In addition to detailed description of centralized architectures defined in previous subsection, we also present detail structure of Z-protocol and framework for management of advanced architectures. The Z-protocol is split into part related to communication of UE with eNBs (Z-UE protocol) and communication between SCeNBs and core network (Z-BS protocol).

4.5.1 **Initial Z-UE protocol API**

The following Table summarizes the initial API procedures.

Elementary Procedure	**Initiating Message**	**Successful Outcome**	**Unsuccessful Outcome**
Z-UE Setup	Z-UE Setup Request	Z-UE Setup Response	Z-UE Setup Failure
Application Offload	Application Offload Request	Application Offload Response	Application Offload Failure
Transparent Container UE	Transparent Container UE Request	Transparent Container UE Response	Transparent Container UE Failure
UE Status		UE Status Information	

Table 27. The initial API procedures.

4.5.1.1 *Z-UE Setup*

4.5.1.1.1 General

The purpose of the Z-UE Setup procedure is to exchange application level configuration data needed for SCM-UE and the serving eNB to interoperate correctly over the Z-UE interface. This procedure erases any existing application level configuration data in the two nodes and replaces it by the one received. This procedure also resets the Z-UE interface like a Reset procedure would do.

4.5.1.1.2 Successful Operation

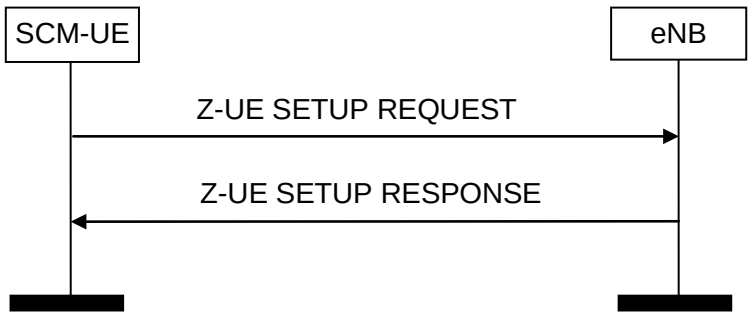


Figure 70. Z-UE Setup, successful operation.

An UE initiates the procedure by sending the Z-UE SETUP REQUEST message to the serving eNB. The serving eNB replies with the Z-UE SETUP RESPONSE message. The initiating UE shall transfer:

- ***Offloading_Capability IE***, providing details on the offloading capabilities of the UE;
- ***Network IP_address_Z_control,*** providing the IP Address, including the port information, of the UE for eNB communication with an SCM placed outside RAN
- ***UE Identifier*** to be used by SCM-BS in the further communication within RAN.

The UE SETUP RESPONSE may include:

- ***Capability_indication***:
  - Deep packet inspection (DPI) available; in this case the Z-UE communication may be done as non-3GPP-traffic.

- ***LCIDs_for_Z-UE***
  - LCIDs to be used for Z-UE signaling in UE Tx and Rx modes
  - LCIDs to be used for Z-UE data in UE Tx and Rx modes.

- ***VM_indication***
  - VM supported on eNB platform

- ***SCM_indication***
  - SCM within RAN or not
  - SCM IP Address

- ***Serving_eNB_IP_Address***
  - Data IP address, including UDP/TCP port
  - Z-signaling IP address, including TCP port.

4.5.1.1.3 Unsuccessful Operation

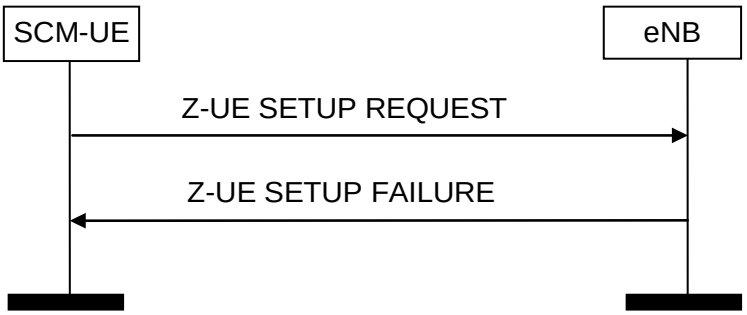


Figure 71. Z-UE Setup, unsuccessful operation.

If the serving eNB cannot accept the setup it shall respond with an Z-UE SETUP FAILURE message with appropriate cause value.
If the Z-UE SETUP FAILURE messages includes the *Time To Wait* IE the initiating UE shall wait at least for the indicated time before reinitiating the Z-UE Setup procedure towards the same eNB.

4.5.1.2 *Application Offload*

4.5.1.2.1 General

The purpose of the Application Offload procedure is to request the offload of the application on an external execution platform.

4.5.1.2.2 Successful Operation

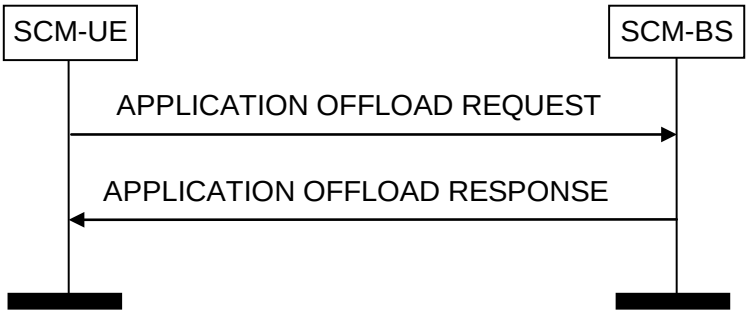


Figure 72. Application offload, successful operation.

An UE initiates the procedure by sending the APP OFFLOAD REQUEST message to the serving eNB. The serving eNB replies with the APP OFFLOAD RESPONSE message will be send by the eNB after asking the SCM to allow the offload and will reflect the SCM response. The initiating UE shall transfer:

- ***Reason IE***, indicating the cause for request, which may be:
  - ***Battery low***
  - ***QoE low***

- ***Application-ID IE,*** providing:
  - IP Address, including the port information, of the UE and the application for communication with an SCM
  - Application_identifier_on_UE
  - Application_program_name

- ***Required_VM_per _Application-ID IE***, including
  - VM performance requirement of the application
  - Capacity of program memory
  - Capacity of data memory.

The APPLICATION OFFLOAD RESPONSE may include:

- ***Acceptance_of_offloading_request***
  - Yes/No

- ***VM_IP_Address*** (in case of Yes)

4.5.1.2.3 Unsuccessful Operation

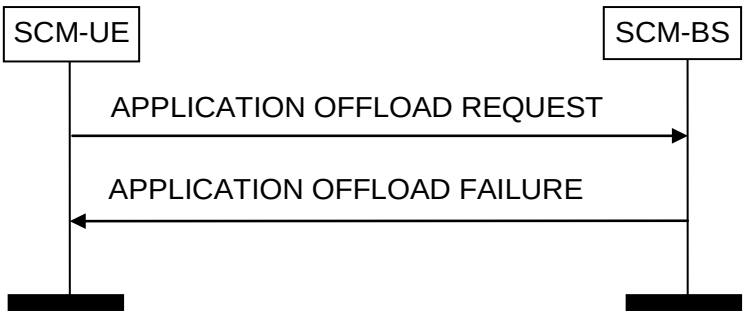


Figure 73. Application offload, unsuccessful operation.

If the serving eNB cannot accept the application offload it shall respond with an APP OFFLOAD FAILURE message with appropriate cause value.
If the APP OFFLOAD FAILURE messages includes the *Time To Wait* IE the initiating UE shall wait at least for the indicated time before reinitiating the APPLICATION OFFLOAD procedure towards the same eNB.

4.5.1.3 *Transparent Container UE*

4.5.1.3.1 General

This procedure allows to send data and eventual commands between UE and a Virtual machine located on the radio serving eNB or on another eNB/ general platform.

4.5.1.3.2 Successful Operation

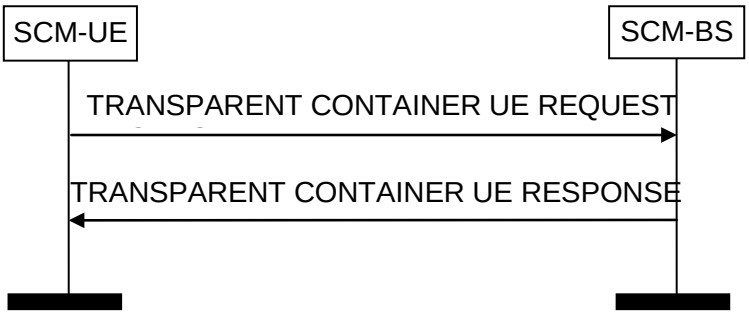


Figure 74. Transparent container, successful operation.

An UE or the serving eNB or a VM located on external platforms (the Container Source) initiates the procedure by sending the TRANSPARENT CONTAINER UE REQUEST message, including the block of transparent commands and data, to the to the Container Recipient. The Container Recipient replies with the TRANSPARENT CONTAINER UE RESPONSE message to acknowledge the correct data reception. The Transparent Container is defined as byte string and its size in bytes:

- ***Container_ID IE*** (an identification number)

- ***Container_Source IE*** (to use only the relevant IEs)**,** providing:
  - IP Address, including the port information, of the Source
  - Application_identifier_on_UE
  - UE_Identifier (if UE is the Source)

- ***Container_Recepient IE*** (to use only the relevant IEs)**,** providing:
  - IP Address, including the port information of the Recipient
  - Application_identifier_on_UE
  - Application_program_name
  - UE_Identifier (if UE is the Recipient)

- ***Container_size*** (in bytes).

TRANSPARENT CONTAINER UE RESPONSE includes the ACK/NAK of the correct reception of a container identified by Container_ID.

- ***Container_ID IE*** (the identification number used in TRANSPARENT CONTAINER UE REQUEST message)
  - UE_Identifier

- Application_identifier_on_UE

- ***ACK_NACK.***

4.5.1.3.3 Unsuccessful Operation

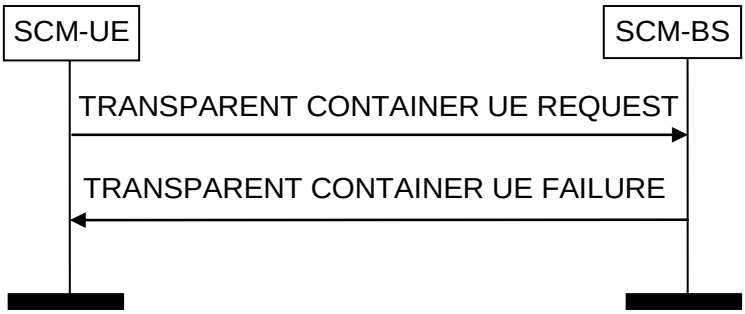


Figure 75. Transparent container, unsuccessful operation.

If the Recipient did not receive the Container content, it shall respond with an TRANSPARENT CONTAINER UE FAILURE message with appropriate cause value.
If the TRANSPARENT CONTAINER UE FAILURE message includes the *Time To Wait* IE the initiating UE shall wait at least for the indicated time before reinitiating the TRANSPARENT CONTAINER UE procedure towards the same Recipient.

4.5.1.4 *UE Status*

4.5.1.4.1 General

The purpose of the UE Status procedure is to provide the eNB with information of the UE parameters, such as UE battery status.

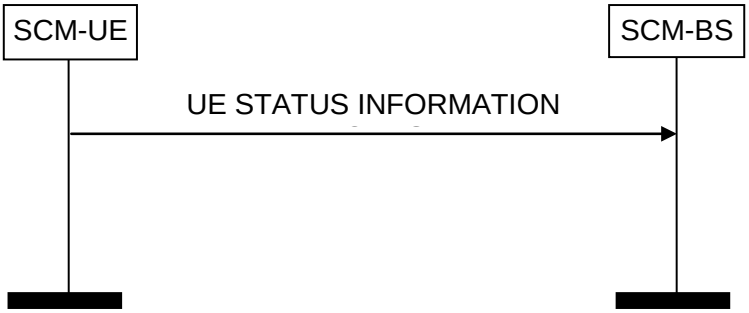


Figure 76. UE STATUS INFORMATION.

The SCM-UE may send the following IEs:

- ***UE_battery_status, indicating*** in the percentage of available battery relative to the full status

- ***UE_mobiliy_indication,***indicating if the UE's GPS indicates a change of position

- ***UE_energy_consumption_per_processor_cycle***

- ***UE_energy_consumption_per_transmitted_bit_per_application***
  - Aplication_ID
  - Energy_per_bit

- ***UE_energy_consumption_per_received_bit_per_application***
  - Aplication_ID
  - Energy_per_bit

4.5.2 Initial Z-BS protocol API

4.5.2.1.1 Z-BS Setup

The following Table summarizes the initial API procedures.

Elementary Procedure	Initiating Message	Successful Outcome	Unsuccessful Outcome
Z-BS Setup	Z-BS Setup Request	Z-BS Setup Response	Z-BS Setup Failure
BS Application Offload	BS Application Offload Request	BS Application Offload Response	BS Application Offload Failure
Backhaul Delay	Backhaul Delay Request	Backhaul Delay Response	
UE Application Throughput	UE Application Throughput Request	UE Application Throughput Response	
Backhaul Throughput per Application	Backhaul Throughput per Application Request	Backhaul Throughput per Application Response	
Transparent Container BS	Transparent Container BS Request	Transparent Container BS Response	Transparent Container BS Failure
Served UE Status		Served UE Status Information	

Table 28. The initial API procedures

4.5.2.1.1.1 *General*

The purpose of the Z-BS Setup procedure is to exchange application level configuration data needed for SCM and the serving eNB to interoperate correctly over the Z-BS interface. This procedure erases any existing application level configuration data in the two nodes and replaces it by the one received. This procedure also resets the Z-BS interface like a Reset procedure would do.

4.5.2.1.1.2 *Successful Operation*

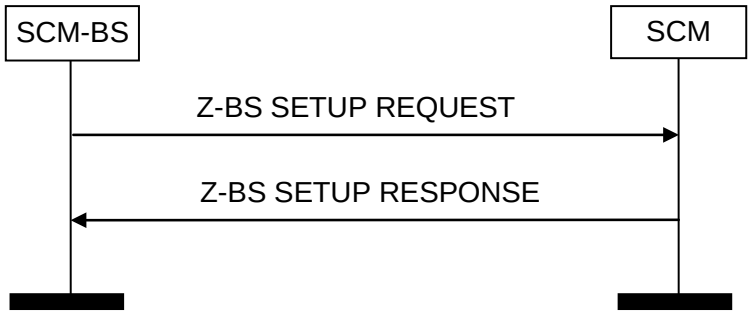


Figure 77. Z-BS Setup, successful operation.

An eNB initiates the procedure by sending the Z-BS SETUP REQUEST message to the pair entity (SCM or another eNB). The pair entity replies with the Z-BS SETUP RESPONSE message.

The initiating eNB_1 will include the following IEs:

- eNB_Identification
  - ECGI
  - IP_Address and port of SCM-BS
- ***SERVED_UE_LIST,*** each element of the list including***:***
  - ***Offloading_Capability IE***, providing details on the offloading capabilities of the UE;
  - ***Network IP_address_Z_control,*** providing the IP Address, including the port information, of the UE
  - ***UE Identifier*** to be used by SCM-BS in the further communication within RAN
- ***Virtual_Machine_capability***
  - Yes/No

The SCM-BS SETUP RESPONSE may include:

- ***VM_indication***
  - VM supported on eNB platform (Yes/No)

4.5.2.1.1.3 *Unsuccessful Operation*

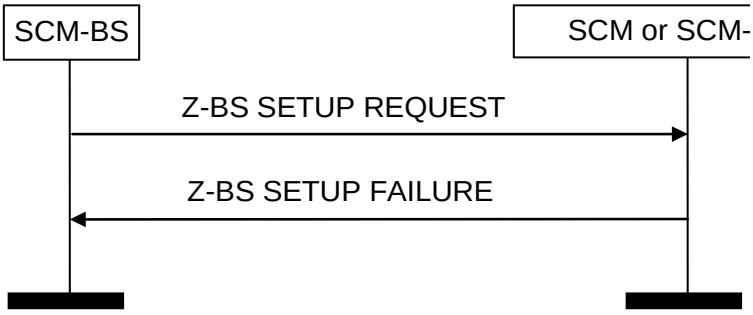


Figure 78. Z-BS Setup, unsuccessful operation.

If the SCM or the SCM-BS on a pair eNB cannot accept the setup it shall respond with an Z-BS SETUP FAILURE message with appropriate cause value.
If the Z-BS SETUP FAILURE messages includes the *Time To Wait* IE the initiating eNB shall wait at least for the indicated time before reinitiating the Z-BS Setup procedure towards the same SCM or SCM-BS.

4.5.2.1.2 BS Application Offload

4.5.2.1.2.1 *General*

The purpose of the BS application offload procedure is to request the SCM or SCM-BS to decide on the offload of an application from an UE served by requesting SCM-BS on an external execution platform.

4.5.2.1.2.2 *Successful Operation*

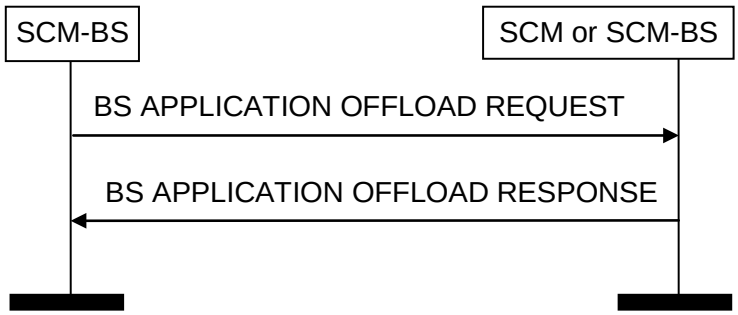


Figure 79. BS application offload, successful operation.

A BS-SCM initiates the procedure by sending the BS APP OFFLOAD REQUEST message to the SCM or pair SCM-BS. The APP OFFLOAD RESPONSE message will be send by the SCM. The initiating SCM-BS shall transfer the information provided by UE over Z-UE interface:

- ***Reason IE***, indicating the cause for request, which may be:
  - Battery low
  - QoE low
- ***Application-ID IE,*** providing:
  - IP Address, including the port information, of the UE and the application for communication with an SCM
  - Application_identifier_on_UE
  - Application_program_name
- ***Application-QoS-Requirements IE***, including:
  - Maximum allowable application delay
- ***Parameters_of_execution offload IE,*** including:
  - Size of the file to be sent to the server for transferring execution
  - Size of the file to be received from the server for transferring execution
- ***Required VM per Application IE***, including
  - VM performance requirement of the application in processor cycles/s
  - Capacity of program memory
  - Capacity of data memory.
- ***Maximum_server_rate*** IE
  - in (CPU cycle/sec)

The APPLICATION OFFLOAD RESPONSE may include:

- ***Acceptance_of_offloading_request***
  - Yes/No
- ***VM IP_Address*** (in case of Yes)
- ***Actual VM performance***
  - VM performance in processor cycles/s
  - Capacity of program memory
  - Capacity of data memory.

4.5.2.1.2.3 *Unsuccessful Operation*

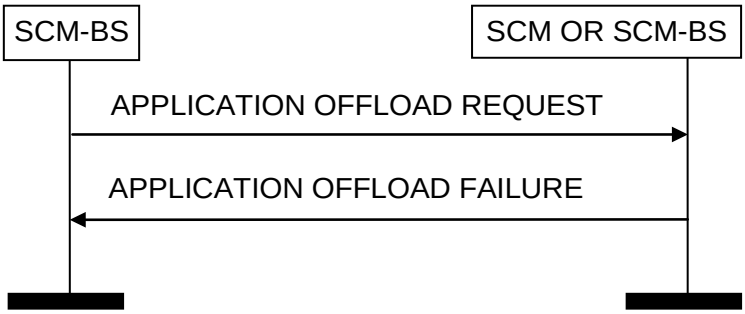


Figure 80. Application offload, unsuccessful operation.

If the serving eNB cannot accept the application offload it shall respond with an APP OFFLOAD FAILURE message with appropriate cause value.

If the APP OFFLOAD FAILURE messages includes the *Time To Wait* IE the initiating UE shall wait at least for the indicated time before reinitiating the APPLICATION OFFLOAD procedure towards the same eNB.

4.5.2.1.3 Backhaul Delay

4.5.2.1.3.1 *General*

The purpose of the Backhaul delay procedure is for eNB1 to provide the SCM or the pair SCM-BS with an IE containing the time indication of its transmission and the indication if the absolute time was used. The pair entity will either determine the one way propagation time in case that is able to determine the absolute time when the IE was received and respond with an IE indicating the one-way delay, or return the packet to transmitter which will be able to determine the round-trip delay.

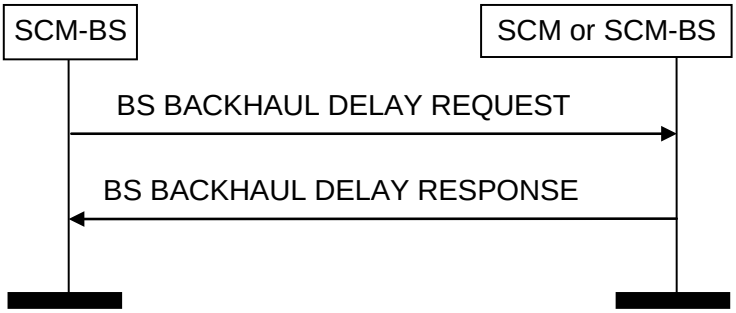


Figure 81. Backhaul delay, successful operation.

4.5.2.1.3.2 *Successful Operation*

An SCM-BS initiates the procedure by sending the BACKHAUL DELAY REQUEST message to the SCM or the pair SCM-BS. The pair SCM replies with the BACKHAUL DELAY RESPONSE message.

- ***Time_tag IE***, indicating the time at which this IE was sent
- ***Clock_generation IE,*** indicating:
  - Clock generated by GPS or equivalent
  - Clock generated locally

The BACKHAUL DELAY RESPONSE may include:

- ***Time_delay IE***, indicating the measured one-way delay
- ***Processing_time IE,*** indicating the processing delay at reving SCM

- ***Relative_time_tag IE,*** indicating the time at which the initiator of the procedure has sent the Time_tag IE.

4.5.2.1.4 UE Application Throughput

4.5.2.1.4.1 *General*

The purpose of the UE application throughput procedure is for SCM-BS2 to provide the SCM or the pair SCM-BS, at the moment of request, with an IE containing the assessed or reserved UE throughput over the Uu interface per a specific application, in bits/sec. This information will be based on the priority of the bearer or of the IP packets carrying the application data and on the serving eNB RRM assessment.

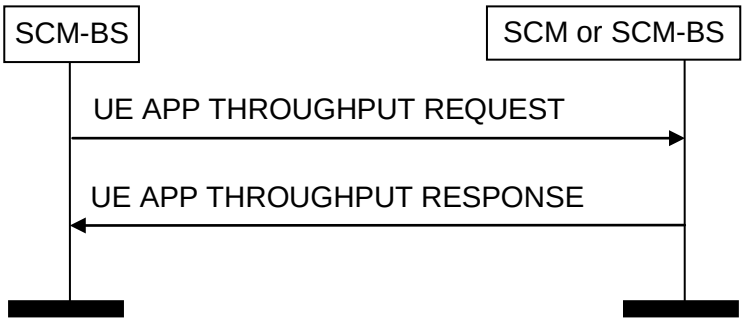


Figure 82. UE throughput per application, successful operation.

4.5.2.1.4.2 *Successful Operation*

An SCM or SCM-BS initiates the procedure by sending the APP THROUGHPUT REQUEST message to the pair SCM-BS. The pair SCM-BS replies with the APP THROUGHPUT RESPONSE message including the throughput in each direction. The procedure is application specific.

The following IE is included in the APP THROUGHPUT REQUEST message:

- ***Application-ID*** IE

The APP THROUGHPUT RESPONSE may include:

- ***Application-ID*** IE
- ***DL_Throughput*** IE;
  - (Throughput in eNB to UE direction)
- ***UL_Throughput*** IE;
  - (Throughput in UE to eNB direction)

4.5.2.1.5 Backhaul Throughput per Application

4.5.2.1.5.1 *General*

The purpose of the Backhaul throughput procedure is for eNB1 to provide the SCM or the pair SCM-BS, at the moment of request, with an IE containing the assessed or reserved backhaul capacity per a specific application, in bits/sec. This information will be based on the priority of the bearer or of the IP packets carrying the application data.

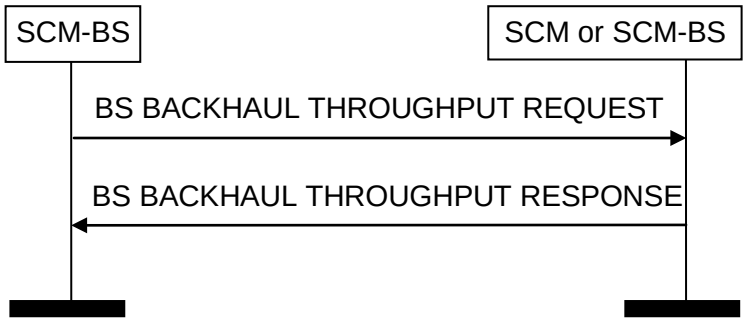


Figure 83. Backhaul throughput per application, successful operation.

4.5.2.1.5.2 *Successful Operation*

An SCM or SCM-BS initiates the procedure by sending the BACKHAUL THROUGHPUT REQUEST message to the pair SCM-BS. The pair SCM-BS replies with the BACKHAUL THROUGHPUT RESPONSE message including the answer. The procedure is application specific.

The following IEs are included in the BACKHAUL THROUGHPUT REQUEST message:

- ***Application-ID*** IE

The BACKHAUL THROUGHPUT RESPONSE may include:

- ***Application-ID*** IE
- ***Throughput*** IE

4.5.2.1.6 Transparent Container BS

4.5.2.1.6.1 *General*

This procedure allows to send data and eventual commands from the radio serving eNB to a Virtual Machine not located on the radio serving eNB and in the reverse direction, from a Virtual Machine not located on the radio serving eNB to the radio serving eNB.

4.5.2.1.6.2 *Successful Operation*

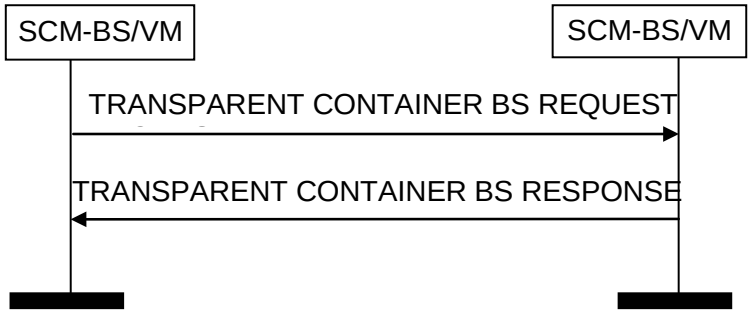


Figure 84. Transparent container, successful operation.

An UE or the serving eNB or a VM located on external platforms (the Container Source) initiates the procedure by sending the TRANSPARENT CONTAINER BS REQUEST message, including the block of transparent commands and data, to the to the Container Recipient. The Container Recipient

replies with the TRANSPARENT CONTAINER BS RESPONSE message to acknowledge the correct data reception. The Transparent Container is defined as byte string and its size is in bytes.

- ***Container_ID IE*** (an identification number)

- ***Container_Source IE*** (to use only the relevant IEs)**,** providing:
  - IP Address, including the port information, of the Source
  - Application_identifier_on_UE
  - UE_Identifier (if UE is the Source)

- ***Container_Recepient IE*** (to use only the relevant IEs)**,** providing:
  - IP Address, including the port information of the Recipient
  - Application_identifier_on_UE
  - UE_Identifier (if UE is the Recipient)

- ***Container_size*** (in bytes).

TRANSPARENT CONTAINER RESPONSE includes the ACK/NAK of the correct reception of a container identified by Container_ID.

- ***Container_ID IE*** (the identification number used in TRANSPARENT CONTAINER REQUEST message)
  - UE_Identifier
  - Application_identifier_on_UE

- ***ACK_NACK.***

4.5.2.1.6.3 *Unsuccessful Operation*

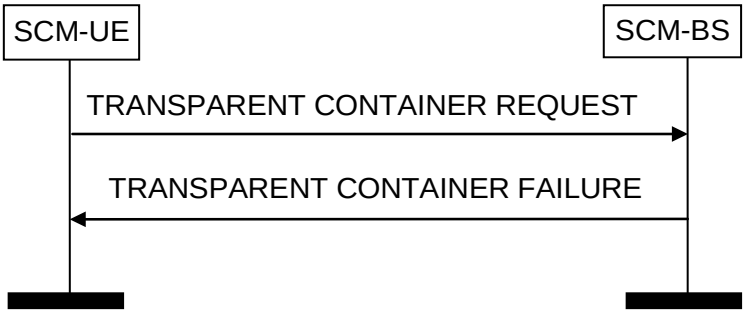


Figure 85. Transparent container BS, unsuccessful operation.

If the Recipient did not receive the Container content, it shall respond with an TRANSPARENT CONTAINER BS FAILURE message with appropriate cause value.
If the TRANSPARENT CONTAINER BS FAILURE message includes the *Time To Wait* IE the initiating UE shall wait at least for the indicated time before reinitiating the TRANSPARENT CONTAINER BS procedure towards the same Receipient.

4.5.2.1.7 Served UE Status

4.5.2.1.7.1 *General*

The purpose of the Served UE Status procedure is to provide the SCM or pair SCM-BS with information of the served UE parameters, such as UE battery status transferred over Z-UE interface.

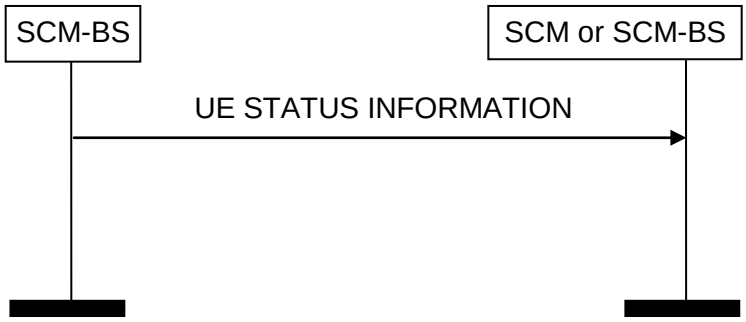


Figure 86. UE STATUS INFORMATION.

The SCM-BS may send the following IEs:

- ***UE_ID*** IE
- ***UE_battery_status, indicating*** in the percentage of available battery relative to the full status
- ***UE_mobiliy_indication,***indicating if the UE's GPS indicates a change of position
- ***UE_energy_consumption_per_processor_cycle***
- ***UE_energy_consumption_per_transmitted_bit_per_application***
    - ***Aplication_ID***
    - ***Energy_per_bit***
- ***UE_energy_consumption_per_received_bit_per_application***
    - ***Aplication_ID***
    - ***Energy_per_bit.***

4.5.3 **High-level Small cell Cloud Management Procedure (SCMP) for advanced architectures**

Based on the options for SCM placement introduced in [TROPIC-D22] (Design of network architecture for femto-cloud computing), including the advanced options outlined in section 9.2, "D42 document" (Adaptation of virtual infrastructure manager and implemented interfaces), and the offloading study provided within the previous sections of this document, the principal details of a respective communication, i.e. the Z protocol, were specified in [TROPIC-D42]. Even though, it covers the communication between the SCeNBce and the SCM, a general procedure compatible with the Z protocol needs to be further examined to include not only communication among SCeNBces, SCM and EPC, but also suitable to integrate mechanism necessary for advanced architectures designs, that were specified within the [TROPIC-D22].

Such management procedure requires to be designed, as it is designated for mutual communication among all cloud enabled elements of the future mobile networks while it covers as many as possible currently known security threats and vulnerabilities. For the reference purposes, it is denoted as the Small-cell Cloud Management Procedure (SCMP). The security analysis of the proposed procedure will be further analyzed, evaluated and summarized in the delivery document [TROPIC-D53].

4.5.3.1 ***Small cell Cloud Management Procedure***

Within this section, the high-level communication principles of the newly proposed SCMP are specified and the impact of the SCM location within the network on the communication exchange is analyzed consequently. In order to propose a new communication scheme among the small cell cloud entities, the current network signalling cannot be affected by the new small cell cloud control signaling. As a result, the entire process of the small cell cloud management can be divided into four phases:

- system bootstrap
- authorization phase

- offloading phase
- system cleanup phase

The bootstrap phase is understood as deployment of virtual machines and hypervisor starts up.

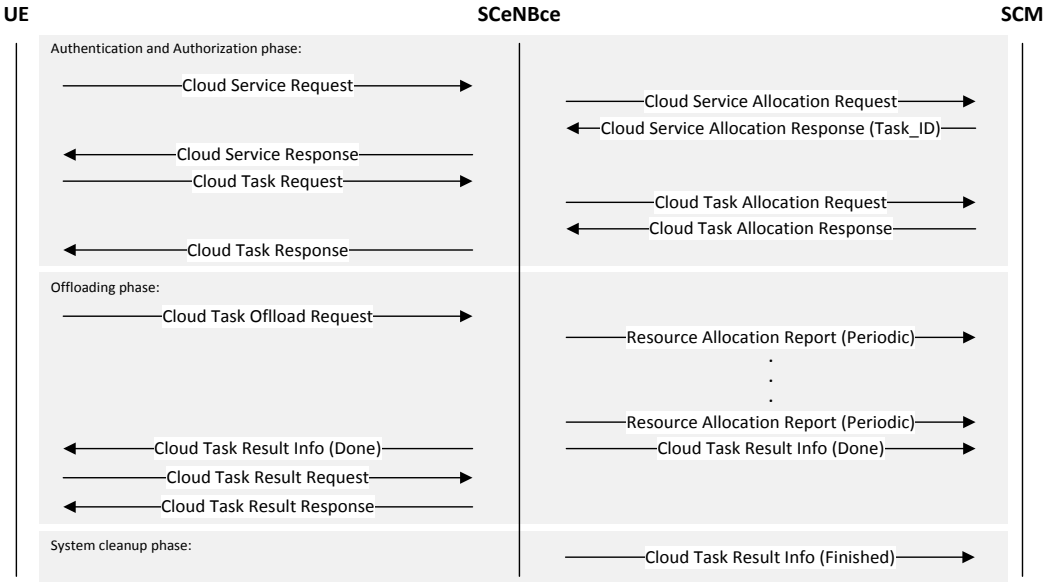


Figure 87. Small-cell Cloud Management Procedure (SCMP).

Once the UE is authorized to the network for common communication purposes, and assuming that none of the following was previously finished earlier, the UE demanding cloud services is queried using a cloud service request message, which is forwarded by the SCeNBce to the SCM as a cloud service allocation request. Based on the result, the UE is either allowed or denied to utilize the available cloud services. Considering the approval was successful, which means that the SCeNBce was assigned a dedicated identifier (Task_ID) to recognize the specific task, the UE sends a cloud task request to the SCM in order to allocate required resources at the SCeNBce(s) (see Figure 87). Apparently, the UE communicates only with the same serving SCeNBce which forwards requests to the SCM only after the UE is approved to use the cloud services (we do not consider the mobility option here). Such arrangement provides a DDoS preventive measurement to protect the SCM, which could have been targeted by malicious UEs.

During the offloading phase, the UE offloads the task to the SCeNBce using the cloud task offload request. As a result, the SCeNBce assigns specific resources (as per details shared by the SCM within the cloud service allocation response) which process the task consequently. To track the current processing state, the SCeNBce periodically reports the status of the task resolution to the SCM. Once the computation is completed, the UE as well as the SCM are informed about the successful task resolution using the cloud task result info message. Once suitable for the UE, it polls the SCeNBce for the offloading results using the cloud task result request, which are immediately delivered using the cloud task result response message.

Afterwards, the successful results delivery is followed by the system cleanup phase, which ensures that the system resources are released and the SCM is updated about the same.

Apparently, this section does not describe any details of the protocol messages such as header or payload fields. Generally it is assumed, that the SCMP is an application layer protocol which utilizes reliable Transmission Control Protocol (TCP) for its proper function. The Z protocol structure is partially specified within the [TROPIC-D42]. However, it covers only necessary processes related to a specific Java-based client-server application example.

4.5.3.1.1 SCMP-Hierarchical (SCMP-H)

The Hierarchical advanced architectural concept distributes the SCM function among multiple SCMs for end-to-end delay minimization as well as for backup purposes. The cloud functions are considered to be still available, even though the SCM located in EPC (R-SCM) is unavailable, by having installed a local SCM (L-SCM) nearer to the SCeNBces, from the distance/delay perspective on the same data path. Therefore, it is necessary to update the communication protocol scheme with messages, which synchronize the system states and status of L-SCMs with the R-SCM.

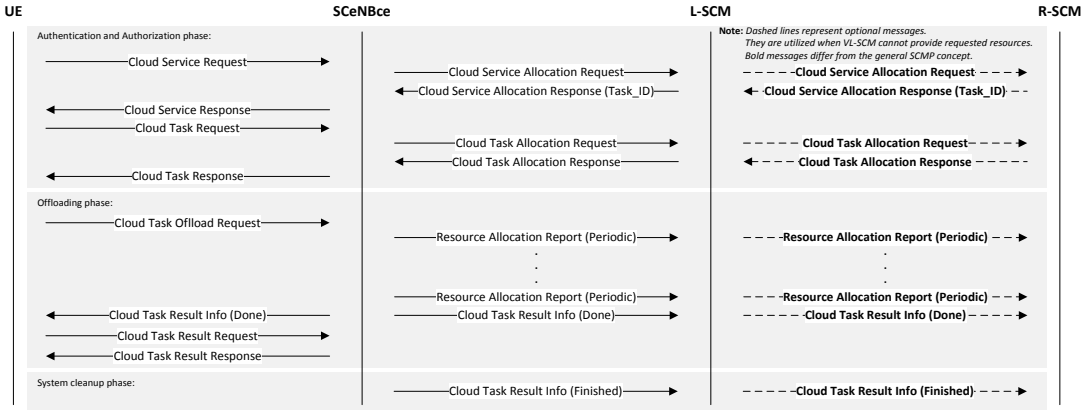


Figure 88. SCMP-Hierarchical (SCMP-H).

Since the architectural scheme counts with unreliable connection among L-SCMs and R-SCM, messages, which are considered to be delivered with delay as a result of physical or logical link failure (R-SCM unavailability), are depicted as dashed lines (see Figure 88). Messages, which are not delivered within a predefined time window, are dropped.

4.5.3.1.2 SCMP-Virtual Hierarchical (SCMP-VH)

The Virtual-Hierarchical concept is a generalized Hierarchical approach, which integrates the L-SCM function directly into SCeNBce. Note, that the messages between SCeNBce and VL-SCM are formally described as between two devices. Even though, they both represent the same physical appliance, they are considered as two logically separated devices from the protocol perspective (see Figure 89).

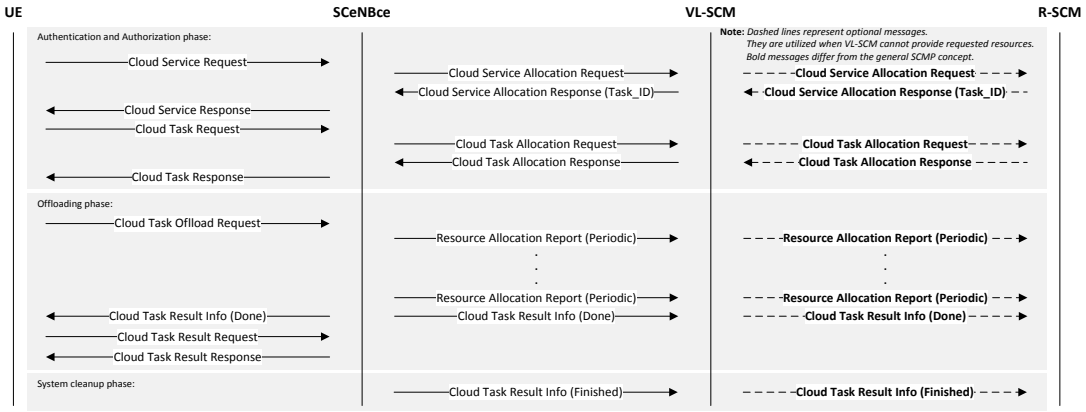


Figure 89. SCMP-Virtual Hierarchical (SCMP-VH).

In this architecture the SCM-BS and eventually the virtual machines are located in the Serving eNB. The communication between the SCM-UE and the SCM-BS is done over the Uu radio interface.

The relevant messages of the Z protocol may be encapsulated within the user data or can be part of the control information transmitted over the Uu interface. In the latter case it is assumed that the Z protocol is part of the cellular radio interface.

4.5.3.1.3 Comparison of advanced architectures

The benefits of advanced designs of the SCC architecture, i.e., hierarchical and virtual-hierarchical, are discussed in [TROPIC-D22]. On the other hand, quantification of advantages has not been addressed yet. Therefore, we focus on formulation of efficiency of both advanced schemes in the terms of signaling overhead and signaling delay.

The performance assessment is based on several assumptions. First, we evaluate Z-protocol messages (defined in [TROPIC-D42]), which are necessary for successful management communication among SCeNBces and the SCM. Second, we analyze information that is needed for local- to remote-SCM synchronization. Since the Z-protocol is not primarily designed for such purpose, we denote all SCM-to-SCM traffic as small-cell cloud management procedure (SCMP). See previous chapter for more details.

Parameter	Value
number of UEs	30
number of SCeNBces	2
average number of requests per single UE within simulation	1
simulation time	6000s
Z protocol periodic updates due in every	15s
SCMP periodic updates due in every	60s
physical link	optic fiber

Table 29. Simulation parameters (unless stated otherwise in text).

Initial conditions, which are set up for the analysis, can be found in Table 29, unless stated otherwise. During the simulation, ideal parameters of SCM and VM are considered as well as parameters of transmission lines. In other words, no memory and disk resources depletion is assumed during the simulation as well as no traffic line congestion is assumed, thus maximum bandwidth is always available.

4.5.3.2 *Z-protocol message analysis*

Basically, the Z-protocol messages are differentiated based on the communication flow direction, i.e. SCeNBce → SCM, namely: `CONNECT` (12B), `RECONNECT` (6B), `CONNECT_USER` (3B), `DISCONNECT_USER` (2B), `MONITOR` (3B); `REQ_VM_IP` (4B) and `REQ_PARALLELIZATION` (2B), and in the reverse direction, namely: `PING` (5B) and `MONITOR` (3B). As these messages are required for successful task offloading process within the small cell cloud architecture, we estimate an average value of size of a typical Z-protocol message exchange including the headers as 40 B.

4.5.3.3 *SCMP message analysis*

The role of the remote SCM (R-SCM) is to act as a backup device in case the local (L-SCM) fails. For that purpose, the SCM requires to store context information of the SCC system. Therefore, the Context vector (CV) is designed to keep track of allowed and assigned memory, CPU and other computation resources. The exact details of the content can be found in [TROPIC-D42]; but basically, the SC table contains ID, IPv4, port number and some other fields requiring 10B, VMtable contains SC_ID, IPv4 and some other fields (10B), Usertable contains ID, name, VM_ID and some other fields (36B) and Parallelization table contains required time and number of unattended VMs (4B). As these messages are required for successful synchronization of the SCM context information, we estimated the average amount of information to be exchanged as 60B. In order to address each SCM for the message exchange, two 1B ID fields are required within the SCMP header making the average size of complete SCMP signaling message equal to 62B.

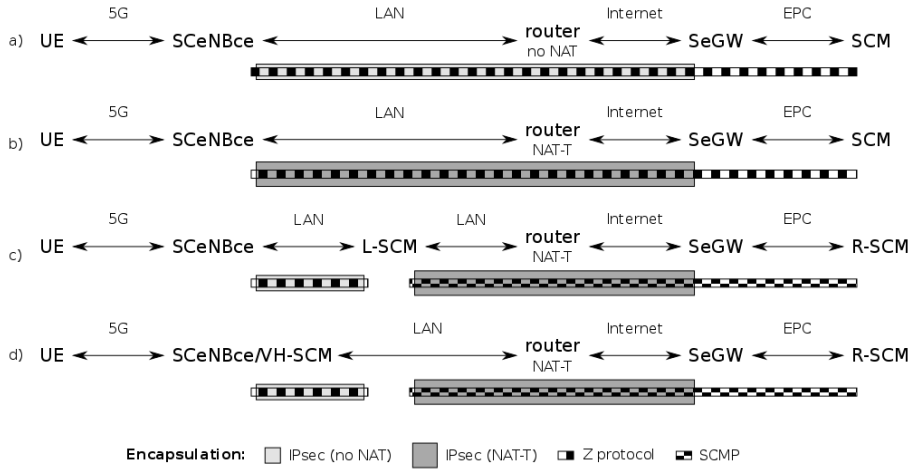


Figure 90. Encapsulation of signalling protocol for different architectural options: a) centralized (No NAT), b) centralized (NAT-T), c) hierarchical (NAT-T), d) virtual-hierarchical (NAT-T).

The discussed advanced architectures utilize various communication schemes and therefore require different encapsulation based on the network part via which the signalization data flow (see Figure 90). Further analyses focus on lower layer protocol encapsulation overhead and usage of either Z-protocol and SCMP, which are designed for cloud signaling purposes.

Since the communication is based on the well-known TCP/IP protocol and related protocols, the header sizes (and related trailer sizes, where applicable) are assumed to be of a size presented in Table 30.

Protocol	**Header size [B] (incl. trailer size where applicable)**
TCP	20
IPv4	20
ESP (IP protocol 50)	24
UDP (port 4500 for NAT-T)	8
Eth (Ethernet)	26

Table 30. Protocol header size.

We denote the appropriate sum of headers in the particular network part by the following equations:

$$\mathrm{HdrSum_{LAN}} = \mathrm{Hdr_{TCP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{Eth}}\ [\mathrm{B}] \quad (51)$$

$$\mathrm{HdrSum_{LAN\_IPsec}} = \mathrm{Hdr_{TCP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{ESP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{Eth}}\ [\mathrm{B}] \quad (52)$$

$$\mathrm{HdrSum_{INT\_IPsec}} = \mathrm{Hdr_{TCP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{ESP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{Eth}}\ [\mathrm{B}] \quad (53)$$

$$\mathrm{HdrSum_{LAN\_IPsec\_NAT-T}} = \mathrm{Hdr_{TCP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{ESP}} + \mathrm{Hdr_{UDP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{Eth}}\ [\mathrm{B}] \quad (54)$$

$$\mathrm{HdrSum_{INT\_IPsec\_NAT-T}} = \mathrm{Hdr_{TCP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{ESP}} + \mathrm{Hdr_{UDP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{Eth}}\ [\mathrm{B}] \quad (55)$$

$$\mathrm{HdrSum_{EPC}} = \mathrm{Hdr_{TCP}} + \mathrm{Hdr_{IPv4}} + \mathrm{Hdr_{Eth}}\ [\mathrm{B}] \quad (56)$$

4.5.3.4 *Signalling Data Overhead*

The newly proposed TROPIC architectures introduce additional signaling overhead. This section deals with analysis of each approach and its contribution to the total overhead required for the management of the offloaded cloud computing services.

For comparison purposes among the proposed designs, network topology presented in Figure 90 is assumed. Cumulative end-to-end signaling overhead for the management purposes of an average service request is studied also for architecture with centralized SCM without network address translation (C-SCM no NAT) scheme for benchmarking. As the real-world IPv4-based scenarios assume NAT applied at the LAN perimeter, we consider also centralized and advanced architectures with IPsec traffic flow NAT traversal (NAT-T) application only, i.e. centralized (C-SCM with NAT), hierarchical (H-SCM with NAT) and virtual-hierarchical (VH-SCM with NAT). As shown in Figure 91 and Figure 92, the H-SCM approach reduces the signaling overhead by 28% and the VH-SCM approach reduces the signalling overhead by 35% compared to C-SCM.

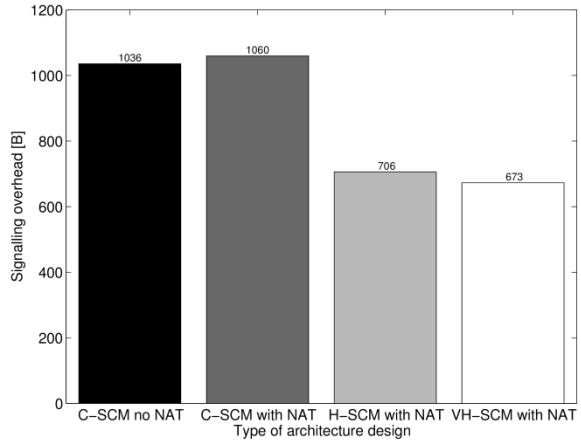


Figure 91. Cumulative signaling overhead of an average cloud service request.

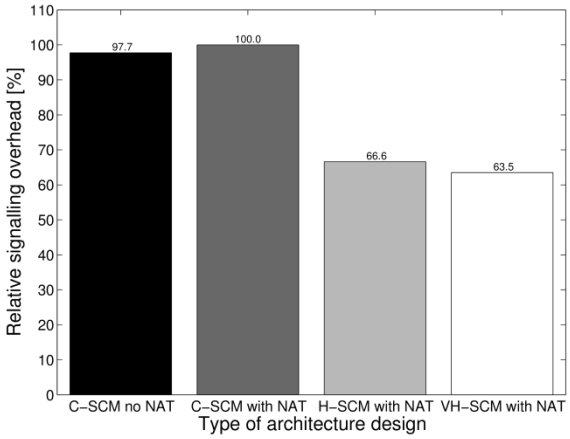


Figure 92. Signaling overhead efficiency of an average cloud service request.

Compared to advanced designs, the cumulative overhead increases rapidly with the number of UEs for the centralized approach (see Figure 93). The same trend can be observed with increasing simulation time (see Figure 94). As a result can be seen, that the advanced designs can save significant amount of data when compared to the centralized scheme. This is caused by the fact that the offloading requests are processed by the appropriate SCM within the LAN rather than transmitted via the Internet.

The difference between H-SCM and VH-SCM approaches is not significant, as it depends on number of SCeNBces. Assuming that each SCeNBce serves the same average number of offloading requests within the simulation time, and average of 1/NumberOfSCeNBces of requests is not forwarded to the VH-SCM, since the VH-SCM is the SCeNBce itself (see Figure 95 and Figure 96).

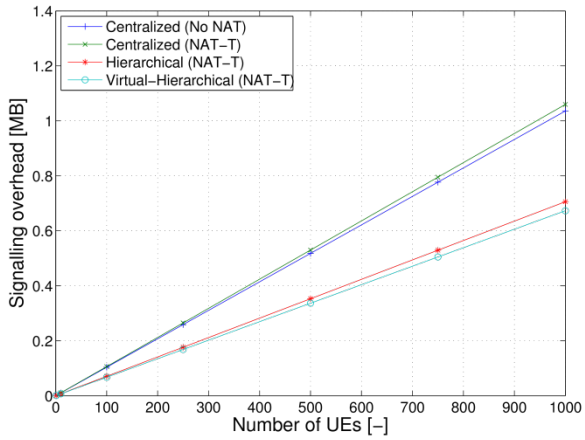


Figure 93. Signaling overhead over number of UEs (2 SCeNBces, 60 seconds).

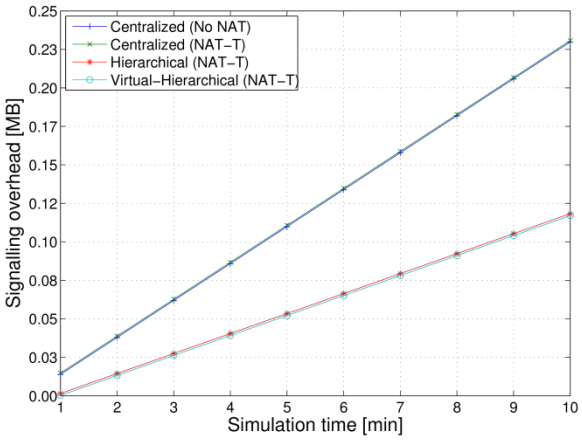


Figure 94. Signaling overhead over time (2 SCeNBces, 30 UEs).

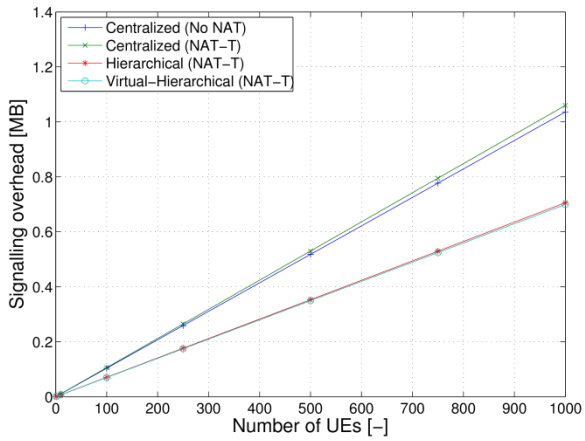


Figure 95. Signaling overhead over number of UEs (10 SCeNBces, 60 seconds).

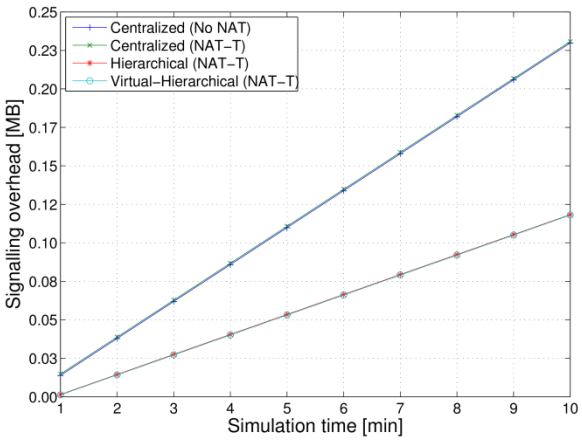


Figure 96. Signaling overhead over time (10 SCeNBces, 30 UEs).

4.5.3.5 *Signalling Delay*

Besides signaling overhead, it is also important to minimize delay of the signalization procedures required to successfully offload computation tasks to the cloud environment. Therefore, we examine also delay of signalling messages, which is introduced by the network environment.

The simulation results show that the decentralized designs are far more efficient when comparing signaling delay in the terms of absolute values (see Figure 97 and Figure 98) as well as relative efficiency (see Figure 97 and Figure 98).

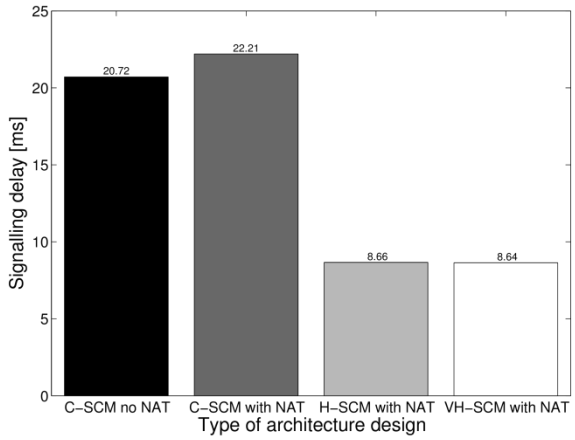


Figure 97. Cumulative signaling delay of an average cloud service request (ADSL).

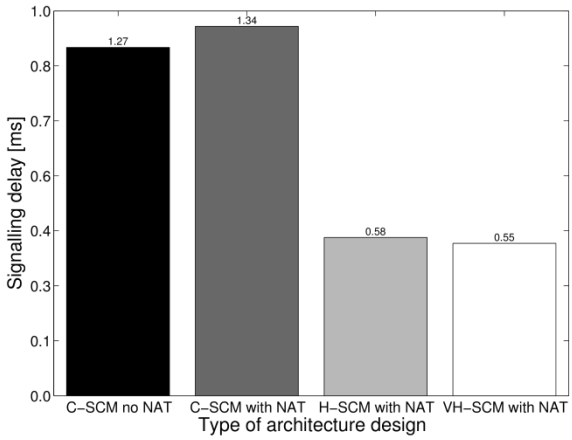


Figure 98. Cumulative signaling delay of an average cloud service request (optical fiber).

The impact of the used WAN technology on the delay is apparent. The slower the transmission rates, the higher is the positive effect of the decentralized approaches The H-SCM reduces the end-to-end signaling delay by 61% and 57% when considering ADSL and optical fiber, respectively. The VH-SCM reduces the delay by approximately 1% and 2% more when considering ADSL and optical fiber, respectively (see Figure 99 and Figure 100).

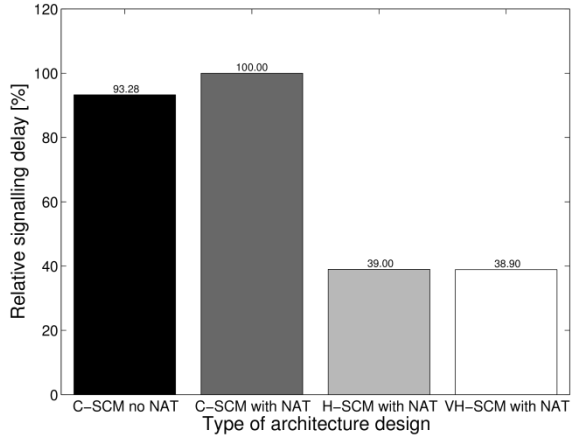


Figure 99. Signaling delay efficiency of an average cloud service request (ADSL).

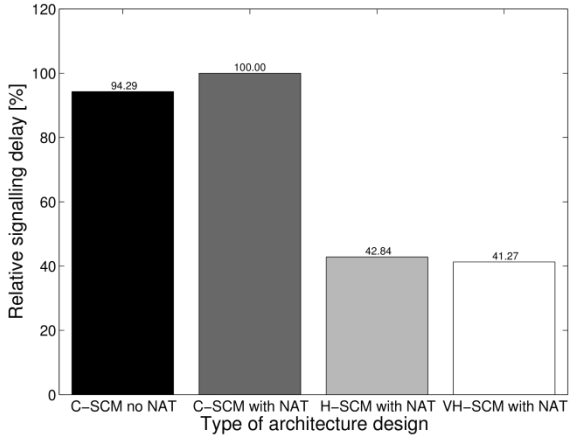


Figure 100. Signaling delay efficiency of an average cloud service request (optical fiber).

Along with the signaling overhead as well as the delay efficiency of advanced designs, the virtual-hierarchical scheme enables to provide immediate backup when the primary VH-SCM fails or becomes otherwise inaccessible.

4.6 *Summary of SCC architecture and protocols*

In this section, we describe functionalities of the SCC system and of all involved functional blocks, modules and interfaces. Also, two possible architectures of small cell cloud and related control protocols are thoroughly presented and analyzed.

The first architecture option does not impact on LTE architecture and it is fully compatible with existing approach. It requires enhancement of the legacy SCeNB with computing capabilities (CPU, memory, storage) and with gateway splitting SCC traffic from the conventional traffic. This way, it is not needed to route all traffic through P-GW and latency due to data transmission is minimized. The second architecture modifies addressing of network entities. This requires that the S-GW will forward the user traffic to the SCeNBce where the application is offloaded. Even if the second option requires

no hardware modification of the small cell base station for protocol translation, it still requires extension of currently available base stations with computing capabilities.

Based on the fact that the first architecture can be used with the existing S-GW, we recommend the first option with protocol translation by SCC-GW as the final architecture adopted for WP6 and for future networks.

As shown by simulations, potential enhancements of network by means of distribution of SCM functionalities closer to the cell edge can significantly reduce signaling overhead as well as signaling delay. Therefore, both options of architecture are designed for easy extension towards advanced architecture solutions by simple distribution of computing capabilities to SCM-BS located in SCeNBces.

5 OFFLOADING PROCESS

Computation offloading has attracted a lot of research efforts as a way to augment the capabilities of resource-constrained and energy-hungry mobile handsets by migrating computation to more resourceful servers. Offloading is useful either to enable smartphones to run more and more sophisticated applications, while meeting strict delay constraints, or to prolong the battery lifetime by limiting the energy spent at the mobile handset to run a given application, still under application-dependent delay constraints. In the current mobile handset market, user demand is increasing for several categories of energy-hungry applications. Video games are becoming more and more popular and they have large energy requests. Mobile users are also increasingly watching and uploading streaming video from YouTube [Kincaid09]-[Lewin08]. Furthermore, mobile devices are increasingly equipped with new sensors that produce continuous streams of data about the environment. New applications relying on continuous processing of the collected data are emerging, such as car navigation systems, pedometers, and location-based social networking, just to name a few. All these applications are severely energy demanding for today's mobile handsets. On the other hand, the trend in battery technology makes it unlikely that the energy problem will disappear in the near future. Indeed, recent researches on the adoption of new battery technologies, e.g., new chemicals, are unlikely to significantly improve battery lifetime [Lewin95]-[Palacin09]. This explains why mobile computation offloading is actually a promising idea to overcome the limitations of the mobile handsets.

Offloading typically requires code partitioning, so that part of the code runs locally, whereas the remaining part may be offloaded to the service provider based on contextual parameters, such as battery level, delay constraints, channel state, and so on. Program partitioning may be static or dynamic. Static partition has the advantage of requiring a low overhead during execution, but it works well only if the parameters related to the offloading decisions are accurately known in advance or predicted. In contrast, enforcing a dynamic decision mechanism makes possible to adapt to different operating conditions. Given the variability of the wireless channel, dynamic partitioning seems more appropriate, but it has an associated higher signaling overhead, which must be taken under control. A dynamic approach to computation offloading consists in establishing a rule to decide which part of the computations may be advantageously offloaded, depending on channel conditions, server state, delay constraints, and so on. Hence in the following sections we propose alternative cross-layer optimization strategies that encompass application, MAC and physical layers into a joint framework. The idea is that the full exploitation of the cloud computing capabilities can be achieved only if the allocation of radio and computational capabilities is performed jointly.

Some of the issues to be tackled by a small cell cloud network are listed below:

- Once a mobile user has instantiated a request for computation offloading, the small cell cloud functionality has to decide whether this request can be satisfied, given the current status of computational requirements and radio channel attenuation (interference).
- Once a request is satisfied, the radio resources (channels, power, modulation and coding scheme, etc.) and computational resources (CPU cycles, etc.) are assigned jointly.
- In a context comprising small cells coexisting within the same macrocell or across nearby macrocells, the computation requests involve the decision about how to assign mobile users to small cloud capabilities, accessing through which small cell base station. The assignment task should take into account, jointly, the congestion concerning both the radio and the computational aspects.

An example of decision process is reported in Figure 101, where the MUE instantiates an offloading requirement and the cloud manager decides whether and how to serve such a requirement. As the first step, offloading criteria (Section 5.1) such as users' requirements and overall network capabilities, including radio, backhaul as well as cloud part are evaluated to make decision on whether to offload or not (Section 5.2). Optimization strategies for joint allocation of transmission and computational resources are also investigated in Section 5.2. Once the offloading decision is positive (even in a way

of partial offloading), a cluster of SCeNBces for computation is selected. The cluster selection must consider performance of the individual cells, their load, and status of the network between the UE and target cells for computation (see [TROPIC_D52]). If a cluster is selected the most efficient path (including various radio and backhaul options) for delivery of data for processing must be found (Section 6.1). Meantime, VMs are deployed at considered SCeNBces. For the selected part, transmission resources are scheduled (WP3) and data are delivered to the computing SCeNBces. After computation, analogical steps are followed in order the following order: select the path, allocate resources, and deliver data.

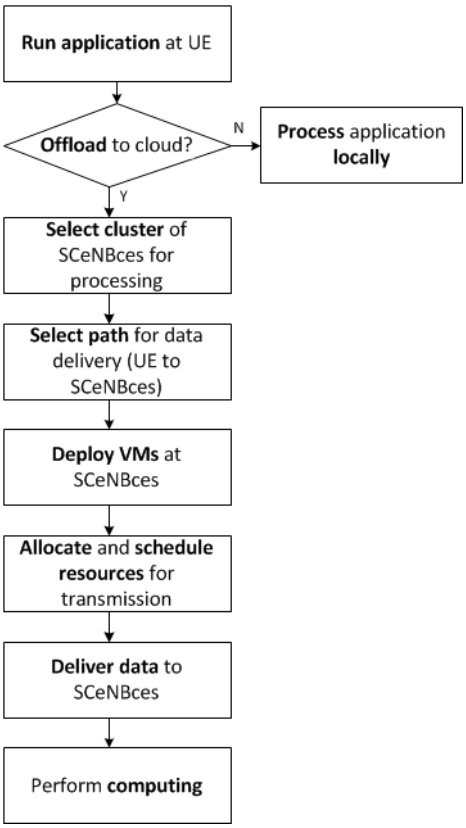


Figure 101. Tasks related to the offloading of the application to SCC.

5.1 *Offloading Criteria*

The full potential of cloud computing can be exploited only when the computation and storage are offloaded to the SCeNBce in order to save energy consumption at the mobile terminal under application-dependent delay constraint. The UE has the possibility to offload the computation or not, depending on several factors such as the type of application, the number of instructions to be executed, the condition of the wireless channel between the mobile terminal and the intended SCeNBce and the availability of the server. The decision on whether to offload or not should take into account the following issues:

- To minimize energy consumption at the UE under latency constraint (or vice versa)
- To guarantee the stability of the queue of instructions to be executed at the mobile side

However these goals can be achieved only if the optimization of the radio and computational resources is performed jointly. According to this, in this deliverable we propose a cross-layer optimization which encompasses the application, MAC and physical layer within a joint framework.

Making a decision in relation with the offloading process has to be carried out under clearly defined criteria. This is of paramount importance since, otherwise, the resulting decision could lead to solutions much worse than the classical full execution of the application at the UE.

One of the main limitations of the current terminals is the limited lifetime of their batteries, specially taking into account the new highly computationally demanding applications. Thus, offloading strategies that minimize the **energy spending at the UE** are required. This energy should encompass not only the energy spend for the local execution of some of the processes of the application at hand, but also the energy that has to be spent for the wireless transmission of the data associated to the execution of the processes of the application that are run remotely at the SCC.

In addition to the previous aspect related to the energy spending, the decision should also take into account the **latency** associated to the execution of the application. It may happen that fully offloading of the application could be very beneficial in terms of energy spending at the UE, but that the delay associated to the wireless transmission of the data to the SCC is higher than the maximum allowable latency. In other words, another important criterion to be considered in the making of the decision is the latency, where this latency has to take into account the time needed for the execution of the processes of the application that are run locally at the UE, the time needed for the execution of the processes of the application that are run remotely at the SCC, and also the time needed for the wireless transmission of the data from/to the UE to/from the SCC.

5.2 *Offloading Strategies*

In general, the offloading decision may involve an entire set of applications running concurrently over a mobile handset or only part of them. Let us start with a single application running on the mobile handset. There are different alternative ways to split the computation related to a single application in parts to be run locally, at the mobile site, or remotely, as listed below.

- Data-oriented partitioning: The application is represented as a string of data to be processed and the offloading decision consists in identifying what percentage of this string should be processed locally and what percentage remotely;
- Program-oriented partitioning: The program to be run is represented as a call graph, i.e. as a set of procedures that compute part of the program and exchange data among them; in such a case, the offloading decision consists in identifying which parts of the program must be run locally or remotely;
- Instruction-oriented partitioning: In a multitask environment, the mobile handset has a queue of instructions to be run, possibly coming from multiple applications running concurrently; the offloading decision has to identify, dynamically, which set of instructions should be run locally or remotely.
- Full offloading – The application or program cannot be split into parts and it is offloaded as a whole.

In all the cases, the offloading decision is made taking into account energy consumption and latency constraints and it is the result of a joint optimization over the communication and computation resources. In this section, we will consider all these alternatives, each addressed in specific following section.

5.2.1 **Data-oriented offloading**

5.2.1.1 ***Optimization of the tradeoff between the energy consumption at the UEs and the latency***

In this section we address the problem formulation corresponding to the offloading process in the case of data-partitioned applications. In this kind of applications, we have a predefined quantity of data to be processed. Such data to be processed can be divided into subgroups so that these subgroups can be processed in parallel (maybe one subgroup will be processed locally at the UE, whereas the other

subgroups will be processed remotely in the SC-cloud). In the following subsections we show how to decide whether offloading part (or all) the processing to the SC-cloud is beneficial and how to split the data.

5.2.1.1.1 Single-user case

In this subsection we propose a methodology for the joint allocation of radio and computational resources, under the goal of optimizing the trade-off between the energy consumption at the mobile terminal and the total latency required for the computation-communication process. As a result of the resource allocation, the optimal transmission strategy is obtained (including the transmission power and the transmission rate for transferring the data among the local handset and the serving small cell), as well as the optimal distribution of the computation at the mobile terminal and the serving small cell.

To formulate the master problem that includes communication and computation, we consider as primary variables the number of bits to be processed at both the mobile terminal, s_{P0}, and the small-cell cloud, s_{P1}, and the time spent for the wireless transmission in both uplink, t_{UL}, and downlink, t_{DL}. We treat the communication stage as a black box within the general master allocation after characterizing the relationship between the energy, time and number of communicated bits for both uplink and downlink.

As a first step, we formulate an optimization problem whose ultimate goal is to minimize the energy consumed by the mobile terminal subject to a latency constraint imposed by the application. We focus on the energy spent by the mobile terminal only as it is the handset the one that is usually battery limited. The energy consumption includes the energy spent for the processing and the energy for sending the input/output bits from the user equipment to SC-cloud and vice versa. Note that if our objective was to minimize the total energy including mobile terminal and the small cell cloud, we should add also the energy terms corresponding to the remote processor and the energy spent in communication by the serving small cell.

The following notation is used in the following: S_{app} denotes the total number of bits to be processed, s_{P1} denotes the bits that will be processed remotely at the SCC, whereas s_{P0} corresponds to the number of bits that will be processed locally at the UE, which means that $S_{app} = s_{P0} + s_{P1}$. We assume that the number of bits to be exchanged is proportional to the amount of processing to be done remotely, i.e., to s_{P1}. Note that by adjusting properly the proportionality factors in the uplink and the downlink, β_{UL} and β_{DL}, we may include a wide range of applications. On the other hand, the energy spent for the processing at the UE, $\varepsilon_{P0} s_{P0}$, is considered to be proportional to the number of bits to processed locally. Of course, the processor has to execute instructions, but this is included in the proportionality factor ε_{P0} that accounts jointly for the Joules/cycle of the processor and cycles/bit of the application.

The optimization problem corresponding to the offloading decision process is formulated as follows:

$$
\begin{aligned}
\underset{s_{P0}, s_{P1}, t_{UL}, t_{DL}}{\text{minimize}} \quad & e_{UL}\left(t_{UL}, \beta_{UL} s_{P1}\right) + \varepsilon_{P0} s_{P0} + e_{DL}\left(t_{DL}, \beta_{DL} s_{P1}\right) \\
\text{s.t.} \quad & \text{C1: } \max\left\{\tau_{P0} s_{P0}, t_{UL} + \tau_{P1} s_{P1} + t_{DL}\right\} \le L_{\max}, \\
& \text{C2: } s_{P0} + s_{P1} = S_{app}, \\
& \text{C3: } e_{UL}\left(t_{UL}, \beta_{UL} s_{P1}\right) - k_{tx,1} t_{UL} \le k_{tx,2} t_{UL} P_{tx,UE}, \\
& \text{C4: } \beta_{DL} s_{P1} \le t_{DL} R_{DL}^{\max}.
\end{aligned}
\tag{57}
$$

The latency is constrained to a maximum value $L_{\max}$ (constraint C1). Note that the latency on the other hand is the maximum between the time required for the local processing, and the sum of the time for

sending the input bits to the SC-cloud, the remote processing itself, and sending the output bits back to the mobile terminal.

Actually, having characterized the energy and latency as a function of the basic variables, we could also focus on minimizing the latency, subject to an energy constraint as well, by exchanging the cost function and constraint C1. Both problems are actually equivalent and provide the same inherent tradeoff between latency and energy consumption.

To generalize the problem it is considered that the application at hand can be divided with full granularity. For the sake of simplicity, again, we assume that such division does not imply any overhead. Although in a practical case full granularity may not be possible, this approach allows understanding the fundamental tradeoffs of the radio-computational resource allocation problem.

Additional constraints to account for the maximum radiation power (constraint C3) and maximum transmission rate which is possible in the downlink channel (constraint C4) may be included as well (note that the maximum DL rate is derived as (20) and is denoted by $R_{DL}^{\max}$).

As the objective function is convex and the constraints are either convex or linear in the optimization variables, problem (57) is convex. Therefore, the solution can be computed efficiently resorting to standard software packages. In general terms, the main drawback of this kind of numerical algorithms is that, although they are quite generic and versatile, they have associated a computational cost that is higher than the cost required by algorithms designed specifically for the problem at hand. In this concrete case, it is possible to find a simplified algorithm that allows solving the optimization problem (57) easily and with a low complexity. The complete steps for obtaining such algorithm can be found at [Munoz14c] and are not reproduced here for the sake of brevity and clarity. Only some partial steps will be detailed in the following.

One of the keys associated to this algorithm is the result that states that the UL energy divided by the number of bits transmitted in the UL depends only on the data rate. According to this, we define the following function (normalized energy):

$$\bar{e}_{UL}\left(r_{UL}\right)=\frac{e_{UL}\left(t_{UL},s_{UL}\right)}{s_{UL}} \tag{58}$$

Thanks to the fact that the UL energy function $e_{UL}\left(t_{UL},s_{UL}\right)$ is convex, it can be proved that the normalized energy $\bar{e}_{UL}\left(r_{UL}\right)$ has a unique minimum that is achieved at the rate $\breve{R}_{UL}$.

From the constraints detailed in problem (57), a lower and an upper bound can be found for the number of bits to be processed remotely s_{P1} and also for the UL rate r_{UL}. These bounds are formulated as

$$s_{P1,\min}\le s_{P1}\le s_{P1,\max},\qquad s_{P1,\min}=\max\left\{0,S_{app}-\frac{L_{\max}}{\tau_{P0}}\right\},\qquad s_{P1,\max}=\max\left\{S_{app},\frac{L_{\max}}{\frac{\beta_{UL}}{R_{UL}^{\max}}+\tau_{P1}+\frac{\beta_{DL}}{R_{DL}^{\max}}}\right\}$$

$$r_{UL,\min}\le r_{UL}\le r_{UL,\max},\qquad r_{UL,\min}\left(s_{P1}\right)=\frac{\beta_{UL}s_{P1}}{L_{\max}-\tau_{P1}s_{P1}-\frac{\beta_{DL}}{R_{DL}^{\max}}s_{P1}},\qquad r_{UL,\max}=R_{UL,\max} \tag{59}$$

It can be proved that the optimum UL rate to be used depends on the number of bits to be processed remotely, and can be written as

$$r_{UL}^{*}(s_{P1}) = \begin{cases} r_{UL,\min}(s_{P1}), & \bar{R}_{UL} < r_{UL,\min}(s_{P1}) \\ \bar{R}_{UL}, & r_{UL,\min}(s_{P1}) \le \bar{R}_{UL} \le R_{UL}^{\max} \\ R_{UL}^{\max}, & \bar{R}_{UL} > R_{UL}^{\max} \end{cases} \quad (60)$$

Now, using the previous results, the objective function in problem (57) to be minimized can be expressed equivalently as the following function f_0 that depends only on the variable s_{P1} and that is convex on such variable:

$$f_0(s_{P1}) = \beta_{UL} s_{P1} \bar{e}_{UL}\left(r_{UL}^{*}(s_{P1})\right) + \left(k_{rx,1}\frac{\beta_{DL}}{R_{DL}^{\max}} + k_{rx,2}\beta_{DL} - \varepsilon_{P0}\right)s_{P1} + \varepsilon_{P0} S_{app} \quad (61)$$

The problem now consists in finding the value of s_{P1} that minimizes the previous expression within the margin $s_{P1,\min} \le s_{P1} \le s_{P1,\max}$. This means that the original multi-variable optimization problem described in (57) has been converted into a one-dimensional convex optimization problem that can be solved using simple gradient-algorithms or techniques based on nested intervals that present excellence convergence properties in terms of accuracy and speed.

For the simulations, we have considered as exemplary application the compression of a set of files with a latency constraint. Some of the files are to be compressed locally; some are to be compressed remotely. Those files to be compressed remotely will be sent through the UL and then the result of the compression will be sent back through the DL. Based on some papers in the literature for commercial handsets and practical applications, meaningful parameters of the model for the local computation are: $\varepsilon_{P0} = 8.6 \cdot 10^{-8}$ J/bit, for the energy required per compressed bit, and $\tau_{P0} = 10^{-7}$ s/bit for the time required per compressed bit [Miettinen10].

The other parameters describing the simulations in terms of energy model, remote computation, bandwidth, remote computation, etc., are $k_{tx,1} = 0.4$ W, $k_{tx,2} = 18$, $k_{rx,1} = 0.4$ W, $k_{rx,2} = 2.86 \cdot 10^{-3}$ W/Mbps, $\tau_{P_1} = \tau_{P_0}/2$, $S_{app} = 5$ Mbytes, $\beta_{UL} = \beta_{DL} = 1$, $W_{UL} = W_{DL} = 10$ MHz, $P_{tx,UE} = P_{tx,SC}$ =100mW. In addition to that, we have imposed an additional lower and upper bound the uplink and downlink transmission rate according to a LTE, where the minimum transmission rate is limited by the minimum modulation and coding scheme $MCS_{\min} = 0.15$ bits/symbol, and the maximum transmission rate is limited by the maximum modulation and coding scheme $MCS_{\max} = 5.55$ bits/symbol. The values for ε_{P0} and τ_{P0} have been aligned with the measurements given in [Miettinen10] for frequency and energy characteristics of local computing in commercial handsets, as well as computation to data ratios in practical applications such as gzip ASCII compress.

We take in the following simulations the case of a SISO channel. The channel gain is assumed to be the same in both UL and DL, i.e., $\gamma_{DL} = \gamma_{UL} = \gamma$, and the serving small cell is assumed to use the maximum transmission power.

An important result is that the offloading decision depends, as expected, on the channel gain. Figure 102 shows that percentage of the file to be processed remotely as function on the channel gain for a total latency constraint of 4 seconds. The blue line corresponds to total offloading, for which all the computation tasks are forced to be done in either the mobile terminal or the serving small cell. The red line corresponds to partial offloading, for which the tasks can be distributed.

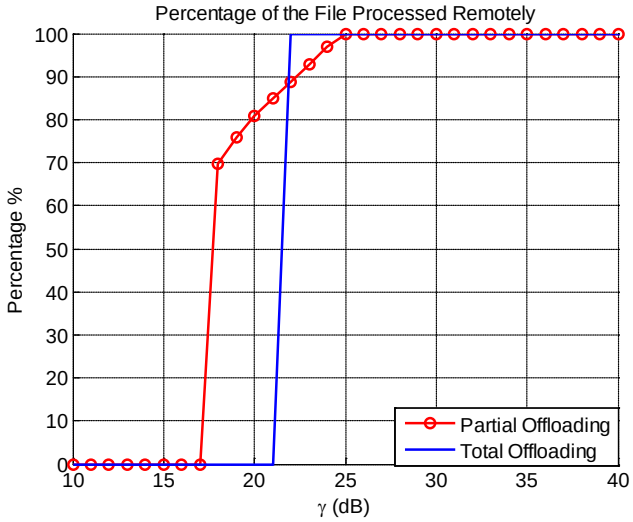


Figure 102. Percentage of the file to be processed remotely versus the channel gain for a total latency constraint of 4 seconds.

Notice that when the link quality is very bad, it is better to do all the processing locally. The reason is that in this case the wireless transmission would require a very high energy, so it is better to process everything in the handset. As the link conditions improve, some of the processing is transferred to the SC-cloud, and for very good link conditions, all the processing is done remotely.

Figure 103 shows the energy consumption when partial or total offloading is decided according the proposed method. Notice that to compress 5 Mbytes locally, around 3.5 Joules are needed. However, using offloading with a channel gain of 25 dB, only 1.5 Joules are needed. The energy saving is therefore greater than 50%. On the hand, notice that partial offloading may reduce the energy consumption at the mobile terminal, as compared to the case of total offloading, for which all the computation tasks are forced to be done either at the mobile terminal or the SC-cloud.

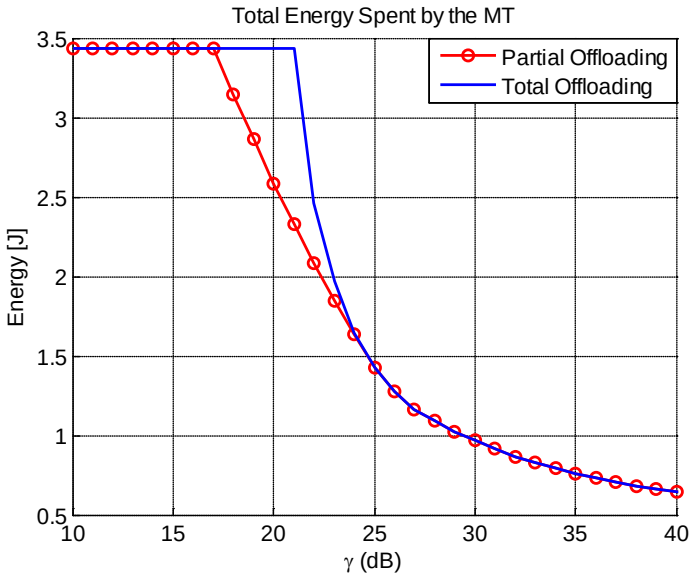


Figure 103. Energy consumption versus channel gain for a total latency constraint of 4 seconds.

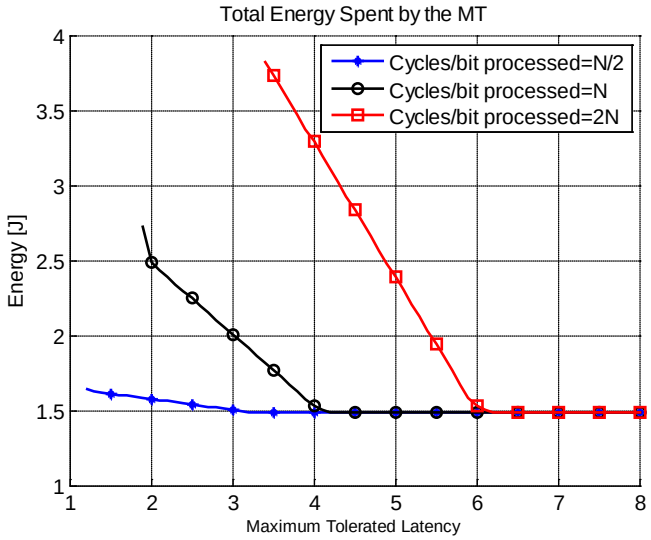


Figure 104. Energy consumption versus the total latency constraint for a channel gain of 25 dB.

The black curve in Figure 104 corresponds to the very same application, but considering different values for the latency constraint. Making the latency constraint tighter, for instance 2 seconds, there is less energy saving compared to the 3.5 Joules corresponding to local execution, either because we need to do some processing locally or because we need to increase the transmission rate to fulfill the latency constraint. Still there is a high energy saving.

The red and blue curves show what happens for more or less computationally demanding applications, respectively. The red curve corresponds to the case of application which requires twice the number of instruction per bit than the gzip compression, and the blue line the half. As the computation demand is greater, the saving in energy is of course greater.

In any case, the total energy spent by the mobile terminal reduces when increasing the maximum tolerated latency, but eventually converges to a minimum value. After this point, the energy saving is not improved even if we relaxed the latency constraint.

The previous figures have shown the evaluation for the case of a SISO channel. The impact of having multiple antennas on the system performance is shown in the following figure through the evaluation of the energy saving in percentage with respect to the case of executing the application completely at the UE for different mean channel gains. In the simulations used for the generation of this figure we have assumed the parameters mentioned before changing the value of β_{DL} so that $\beta_{DL} = 0.2$. As it can be seen, increasing the number of antennas implies that a higher reduction in energy consumption is achieved, as expected.

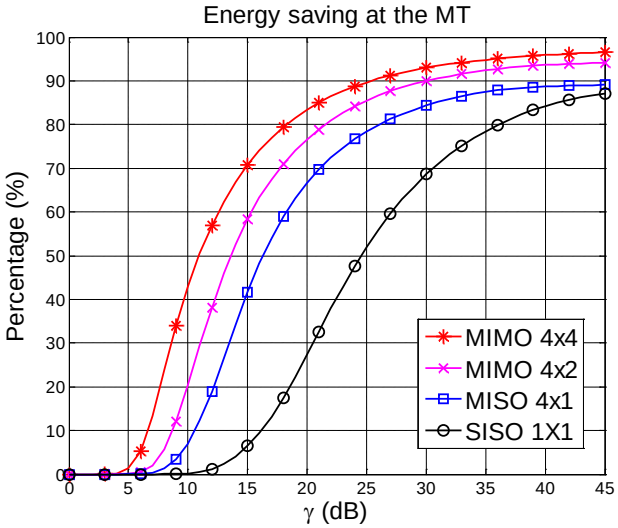


Figure 105. Energy saving with respect to the case of no performing any offloading for different numbers of antennas.

In conclusion, with the proposed approach we are able to adapt the strategy as far as offloading is concerned to the link quality. The optimal strategy and the energy saving will depend as well on the maximum tolerated latency, the application and processors parameters, and the communication constraints.

Previous results have been presented in [Munoz13a], and further analytical analysis with the complete mathematical developments can be found in [Munoz14c].

5.2.1.1.1.1 *Offloading decision protocol*

As already commented previously, the decision concerning whether offloading the data to be processed by an application (or a part of it) from the UE to the SC-cloud is beneficial or not depends on several aspects. These aspects encompass the wireless channel state between the UE and the serving small cell, the computational load capacity and status of the UE and the SC-cloud, the maximum delay/latency constraints imposed by the user, and the characteristics of the application under consideration, among others.

In addition to the optimization itself leading to the decision, an important issue to be taken into account is that the implementation of the decision-taking process requires interchanging a number of messages and information among the entities participating in the process, namely, the UE, the serving small cell, and the SCM. In the following, we describe a preliminary protocol for the interchange of such messages. This description is based on the following assumptions:

- **Application:** the application considered in this protocol has to process a predefined number of bits S_{app}. It is also assumed that such application can be divided into a number of sub processes (each of them processing a part of the total number of bits S_{app}) that can run in parallel where, maybe, some of these sub processes are offloaded to the SC-cloud and some of them run locally at the UE.
- **Decision criterion:** the protocol described in the following is based on the interchange of messages where the objective is to minimize the energy spent by the UE (both in the local computation and the wireless transmission) subject to a maximum delay (also called latency) constraint in the execution of the application.

This preliminary protocol should be generalized to accommodate, for example, other kinds of quality constraints or applications (e.g., applications where the quantity of information to be processed is not

known a-priori since it is generated in real-time as a result from the interaction with the user). Also, this protocol should be adapted to encompass other practical constraints related, for example, with the granularity of the rates to be used in the wireless communication according to the standard at hand, the quantization of the information to be exchanged, etc. All these issues will be considered in the description of the protocol to be included in the implementation of the simulator and the demonstrator in WP6, producing a new practical version of the protocol that will be described in the corresponding milestones and deliverables.

The complete graph of the preliminary protocol is shown in the following figure:

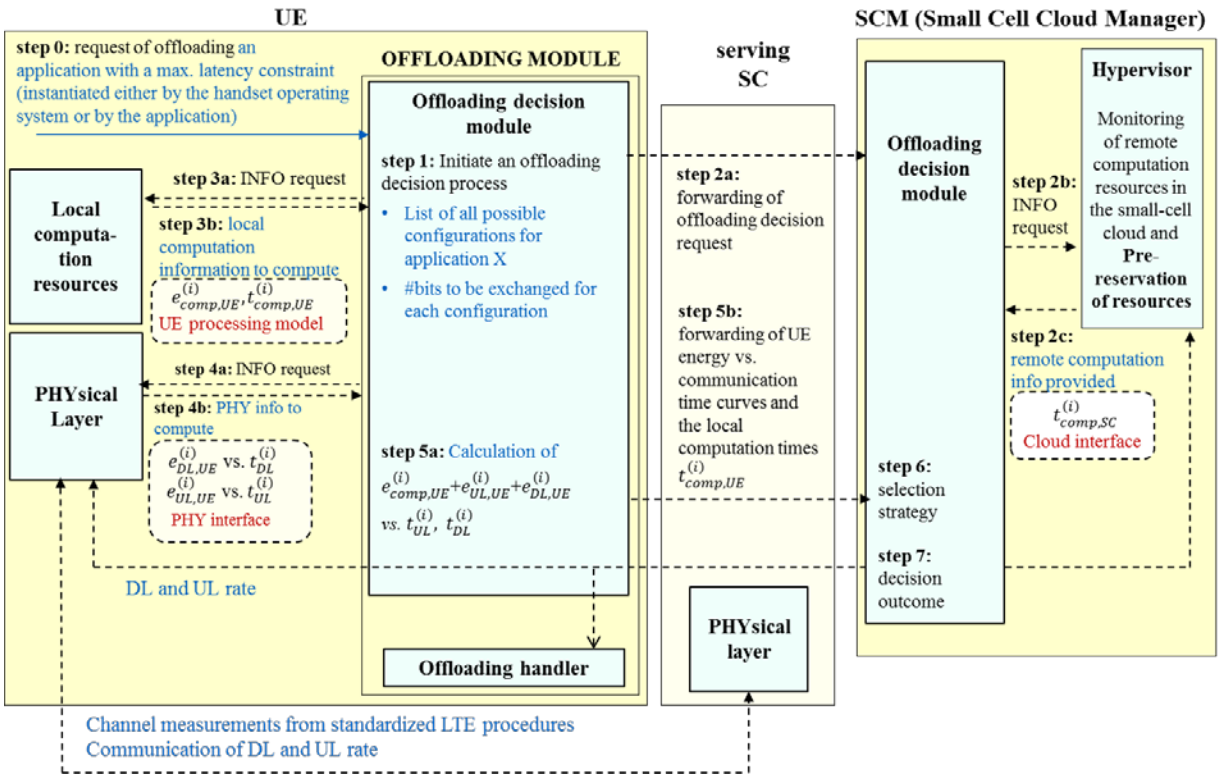


Figure 106. Block diagram for the implementation of a preliminary protocol for the interchange of messages in the offloading decision process.

In what follows, a description of the main generic steps comprising the offloading protocol shown in the previous figure is provided:

Step 0: Generation of a request from the UE for executing an application

Description: The UE launches a request for executing an application with certain QoS parameters, such as a maximum allowed latency. This request implies that a decision has to be taken concerning what is the most beneficial offloading strategy or if it is better to execute the complete application locally at the UE.

Step 1: Reception of such request at the offloading decision module at the UE

Description: The offloading decision module within the UE receives the request and initiates the process to take a decision concerning the offloading of such application. Such decision will be finally taken at the SCM.

Comments: Notice that each application will have a certain number of bits to be processed, S_{app}. This information must be available at the offloading decision module. If the processing can be divided

between the UE and the SCC, the amount of bits to be processed locally (s_{P_0}) and remotely (s_{P_1}) will be related to this quantity. Intuitively, both s_{P_0} and s_{P_1} will increase with S_{app} and decrease with each other. A simple model for an application which is divided into 2 modules to be executed in parallel (in the UE and the SCC) is the following (this is the model taken in the problem formulation (57)):

$$s_{P_0} + s_{P_1} = S_{app}, \tag{62}$$

The number of bits to be exchanged between the UE and the SCC in both directions (s_{UL} for the communication UE→serving SC in the UL and s_{DL} for the communication serving SC→UE in the DL) is a parameter that depends on each possible configuration for the execution of the application. Both numbers will depend on the amount of processing to be done remotely. Intuitively, both s_{UL} and s_{DL} will increase with s_{P_1}. A simple illustrative example of a possible model could be the following:

$$s_{UL} = \beta_{UL} s_{P_1}; \;\; s_{DL} = \beta_{DL} s_{P_1}, \tag{63}$$

where β_{UL} and β_{DL} are constants. Again, this is the formulation adopted in problem (57).

Actions to be taken by the offloading decision module at the UE after the initiation of the procedure to take an offloading decision:

A1.1 For the considered application, the offloading decision module consults the catalogue of applications to be possibly offloaded. Such catalogue should contain, for the application at hand, the list of all possible configurations in terms of distributed sub processes to be divided between the UE and the SCC. The catalogue will be based on application model (Section 3.3). For each possible configuration information about s_{P_0}, s_{P_1}, s_{UL}, s_{DL}, and number of cycles/byte_processed locally, C_{loc}, and remotely, C_{rem}, must be retrieved from the catalogue to complete the following table, where it has been considered as an illustrative example of application the gzip ASCII compression [Miettinen10].

Possible configuration of offloading modules		
i=0	$s_{P0}= S_{app}$, s_{P1}=0, s_{UL}=0, s_{DL}=0, C_{loc}=330 cycles/byte, C_{rem}=0	Total local processing (1 sub process at UE)
i=1	s_{P0}=0, $s_{P1}= S_{app}$, $s_{UL}= S_{app}$, s_{DL}=0.2$S_{app,}$ C_{loc}=0, C_{rem}=330 cycles/byte	Total offloading (1 sub process at SC-cloud)
i=2	s_{P0}=0.6·S_{app}, s_{P1}=0.6·S_{app}, s_{UL}=0.6·S_{app}, s_{DL}=0.12·$S_{app,}$ C_{loc}=330 cycles/byte, C_{rem}=330 cycles/byte	Partial offloading (1 sub process at UE, 1 process at SC-cloud)
i=3	s_{P0}=(0.3+0.3)·S_{app}, s_{P1}=0.6·S_{app}, s_{UL}=0.6·S_{app}, s_{DL}=0.12·$S_{app,}$ C_{loc}=330 cycles/byte, C_{rem}=330 cycles/byte	Partial offloading (2 parallel sub processes at UE, 1 process at SC-cloud)
i=4	s_{P0}=0.6·S_{app}, s_{P1}=(0.3+0.3)·S_{app}, s_{UL}=0.6·S_{app}, s_{DL}=0.12·$S_{app,}$ C_{loc}=330 cycles/byte, C_{rem}=330 cycles/byte	Partial offloading (1 sub process at UE, 2 parallel sub processes at SC-cloud)
…	…	…
i=N		

Table 31. Example of a list of different possible configurations consisting in possible divisions of the application into sub processes to be executed at the UE and the SC-cloud in parallel. The considered application is gzip ASCII compression with S_{app} = 5 Mbytes.

The notation $s_{Pi}=(X+Y)\cdot S_{app}$ in the previous table means that at side *i* (either the UE or the SC-cloud) 2 sub processes run in parallel, each of them processing $X\cdot S_{app}$ and $Y\cdot S_{app}$ bits. This notation could be extended to more than 2 parallel sub processes.

A1.2 Do steps 2a, 3a, and 4a.

Step 2a: Offloading decision module at the UE → Offloading decision module at the SCM

Description: The offloading decision module at the UE will forward the offloading request to the offloading decision module at the SCM. It is assumed that the same catalogue of applications (jointly with the corresponding possible execution configurations) is available also at the offloading decision module at the SCM. This means that the offloading decision module has also the information in Table 31, at least regarding s_{P1}, the configuration of how the sub processes are executed in parallel, and the required cycles/byte for the modules to be executed remotely at the SC-cloud for each configuration.

Step 2b: Offloading decision module at the SCM → Remote computation monitor (at the Hypervisor)

Description: After receiving a request from the offloading decision module at the UE, the offloading decision module at the SCM requests information to the monitoring module at the hypervisor in the SCM (the module at the SCM in charge of monitoring the computational resources of the SCC).

Actions to be taken by the monitoring module after receiving the request from the offloading decision module: The monitoring module will consult the available computational resources in the SCC, and eventually will make a pre-reservation of resources.

Step 2c: Monitoring module → Offloading decision module at the SCM

Description: Based on the information retrieved by the remote computation module, two options could be possible:

- The monitoring module will compute an estimation of latency associated to the remote execution of the potentially offloaded sub processes associated to each configuration: $t_{comp,SC}^{(i)}$ and forward this information to the offloading decision module at the SCM.
- The remote computation module will forward the information retrieved to the offloading decision module at the SCM, that will compute the estimate of $t_{comp,SC}^{(i)}$ for each possible configuration.

Step 3a: Offloading decision module at the UE → Local computation module at the UE

Description: The offloading decision module requests information to the local computation module for the (i) J/cycle (local energetic cost) and (ii) cycles/sec (local computational capacity), as these values may depend on the current CPU usage and the energy efficiency/status of the terminal.

Step 3b: Local computation module at the UE → Offloading decision module at the UE

Description: For example, for a Nokia N810 we may have the following combinations of clock rate (cycles/sec) and cycles/energy: (400 MHz, 480 MC/J), (330 MHz, 480 MC/J), (266 MHz, 540 MC/J), (165 MHz, 510 MC/J) [Miettinen10]. We assume that the working combination (frequency, cycles/energy) is chosen by the local computation module.

Action for the offloading decision module at the UE after receiving a response from the local computation module: From the information provided by the local computation module about the

J/cycle and cycles/sec values, the offloading decision module calculates firstly the value for ε_{P_0} (energy/bit_processed) and τ_{P_0} (seconds/bit_processed). From these values and $s_{P_0}^{(i)}$ (i.e., number of bits to be processes locally for each possible execution configuration), the offloading decision module will compute:

- $e_{comp,UE}^{(i)}$: the energy spent by the UE associated to the execution of the local sub processes corresponding to configuration *i*. It can be computed as $e_{comp,UE}^{(i)} = s_{P_0}^{(i)} \varepsilon_{P_0}$.
- $t_{comp,UE}^{(i)}$: the latency associated to the execution of the local sub processes at the UE corresponding to configuration *i*. It can be computed as $t_{comp,UE}^{(i)} = s_{P_0}^{(i)} \tau_{P_0}$.

Possible configuration of offloading modules	
i=0 (No offloading)	ε_{P_0} =1e-7; τ_{P_0} = 8.6e-8, $e_{comp,UE}^{(0)}$ =3.43 J, $t_{comp,UE}^{(0)}$ = 4.12 s
i=1 (Total offloading)	ε_{P_0} =1e-7; τ_{P_0} = 8.6e-8, $e_{comp,UE}^{(0)}$ =0, $t_{comp,UE}^{(0)}$ =0
…	
i=*N*	

Table 32. Example of calculation of the local computation time and local computational energy spending at the UE for different execution configurations assuming that the UE is characterized by the following computational parameters: 400 MHz (clock rate), 480 MC/J (cycles/energy). The considered application is gzip ASCII compression with S_{app} = 5 Mbytes.

Step 4a: Offloading decision module at the UE → PHY layer at the UE

Description: The offloading decision module requests information to the PHY layer about the optimum trade-off curves ($e_{DL,UE}^{(i)}$ vs. $t_{DL}^{(i)}$ and $e_{UL,UE}^{(i)}$ vs. $t_{UL}^{(i)}$, both for the DL and UL wireless transmission for each possible execution configuration) relating the energy (that will be spent by the UE) and the time necessary to transmit information through the wireless channel or the necessary information to compute and reproduce such curves. Actually, only a set of points in such curves will be feasible, according to the MCSs allowed by the radio interface defined by the standard and the channel conditions.

Step 4b: PHY layer at the UE → Offloading decision module at the UE

Description: The PHY layer will provide the required information to compute the optimum trade-off curves ($e_{DL,UE}^{(i)}$ vs. $t_{DL}^{(i)}$ and $e_{UL,UE}^{(i)}$ vs. $t_{UL}^{(i)}$).

Step 5a: Computation of energy and latency at offloading decision module at the UE

Description: From the information received from the PHY layer, the optimum trade-off curves ($e_{DL,UE}^{(i)}$ vs. $t_{DL}^{(i)}$ and $e_{UL,UE}^{(i)}$ vs. $t_{UL}^{(i)}$) will be computed (in a practical implementation case, actually only the feasible points of these curves will be computed, according to the possible set of discrete MCSs allowed by the radio interface). The total energy spending at the UE vs. the UL $t_{UL}^{(i)}$ and DL $t_{DL}^{(i)}$ times will be computed as $e_{comp,UE}^{(i)} + e_{UL,UE}^{(i)} + e_{DL,UE}^{(i)}$.

Step 5b: Offloading decision module at the UE → Offloading decision module at the SCM

Description: The offloading decision module at the UE will forward the energy $e_{comp,UE}^{(i)} + e_{UL,UE}^{(i)} + e_{DL,UE}^{(i)}$ vs. UL/DL times ($t_{UL}^{(i)}$ and $t_{DL}^{(i)}$) to the offloading decision module at the SCM., jointly with the local computation time for each possible configuration $t_{comp,UE}^{(i)}$.

Step 6: Offloading decision taken at the offloading decision module at the SCM

Description: The offloading decision module at the SCM will discard all the points in the trade-off curves $e_{comp,UE}^{(i)} + e_{UL,UE}^{(i)} + e_{DL,UE}^{(i)}$ vs. total time ($\max\left\{t_{UL}^{(i)} + t_{DL}^{(i)} + t_{comp,SC}^{(i)}, t_{comp.UE}^{(i)}\right\}$) that do not fulfill the maximum latency constraint. Then it will search the configuration that minimizes the energy among the non-discarded combinations (those that fulfill the maximum latency constraint). This will result in the decided offloading strategy.

Step 7: Offloading decision communication taken at the SCM → Offloading handler at the UE

Description: The offloading decision module at the SCM communicates the decision to the offloading executive module (offloading handler) at the UE and to the hypervisor at the SCM. The offloading handler takes over and takes the necessary steps to proceed with the execution of the application under the configuration decided by the offloading decision module and the hypervisor finalizes the corresponding reservation of computation resources at the SC cloud.

During the process some remote resources could have been pre-reserved and now can be released if the decision indicates so.

5.2.1.1.2 Example of extension to a multi-user scenario

In section 5.2.1.1.1 we have presented a framework for the optimization of the offloading suitable for data-partitioned applications and single-user scenarios. Such framework is directly applicable to a multi-user scenario where the frequency resources have been pre-allocated among the different users. Furthermore, although the initial formulation assumes that the objective is to minimize the energy consumption under a maximum latency constraint, such formulation can be adapted easily to encompass other objectives, constraints and/or scenarios.

In this subsection we consider an example of a possible extension of the previous commented offloading strategy for the case of a multi-user scenario where it is considered that the time resources have been pre-allocated uniformly among all the users. Specifically, we consider a system with *K* active users where each user receives 1/ *K* of the time resources. In such a case, the formulation needs to be slightly modified. Besides, we assume that all the users have already decided to offload all the data to the SC under the objective of minimizing the experienced latency. This problem is addressed in the following and it has been already presented in [Munoz14a].

For the *k*-th user, the number of bits to be transmitted in the uplink and downlink are given by $S_{UL,k} = \beta_{UL,k} \cdot S_{app,k}$ and $S_{DL,k} = \beta_{DL,k} \cdot S_{app,k}$ respectively. In a single user setup, $t_{UL,k}$ only depends on the UL data rate, $r_{UL,k}$, and the number of bits to be transmitted, i.e., $t_{UL,k} = S_{UL,k} / r_{UL,k}$. However, in a multiuser setup with time round robin scheduling, the time required is given by $t_{UL,k} = K \cdot S_{UL,k} / r_{UL,k}$ as the user receives only 1 over K of the time resources. Similarly, for the remote processing and the DL transmission, the time required in a multiuser setup with *K* users are $t_{P,k} = K \cdot I_{P,k} / R_P$ and $t_{DL,k} = K \cdot S_{DL,k} / r_{DL,k}$, respectively, where R_P represents the number of instructions per second that the SCeNB is able to execute and $r_{DL,k}$ is the DL data rate corresponding to the *k*-th user. The problem to be solved is the following:

$$\begin{aligned} \min_{r_{UL,k}, r_{DL,k}} \quad & L = \frac{K \cdot \beta_{UL,k} \cdot S_{app,k}}{r_{UL,k}} + \frac{K \cdot I_{P,k}}{R_P} + \frac{K \cdot \beta_{DL,k} \cdot S_{app,k}}{r_{DL,k}} \\ \text{s.t.} \quad & e\left(r_{UL,k}, r_{DL,k}\right) \le E_{\lim,k}, \\ & r_{UL,k} \le R_{UL,k}, \\ & r_{DL,k} \le R_{DL,k}. \end{aligned} \tag{64}$$

where the objective function L is the latency of the overall offloading process and the function $e(r_{UL,k}, r_{DL,k})$ models the energy consumed by the k-th user terminal for the offloading. As it has been already explained, $e(r_{UL,k}, r_{DL,k})$ depends on the UL and DL data rates of the k-th user, $r_{UL,k}$ and $r_{DL,k}$, whose maximum values are given by $R_{UL,k}$ and $R_{DL,k}$ respectively. Finally, $E_{\lim,k}$ is a limit imposed over the energy spent by the UE. A meaningful criterion for setting $E_{\lim,k}$ in the energy constraint of problem (64) is to compare the energy consumption for the offloading with the energy required to do all the computation locally at the terminal side, that is $E_{\lim,k} = \alpha \varepsilon_{P,k} S_{app,k}$, being $1-\alpha$ the desired target percentage of saved energy at the user terminal when doing the offloading, with respect to the energy consumption associated to local processing.

An important aspect to be considered in a multiuser set up is if the user may turn off the transmitter and receiver chain while waiting a new transmission/reception opportunity. Taking this into account, we have considered three cases:

Case (i): We consider the additional energy that the terminal has to spend for the offloading is the energy term that depends on the transmitted power and the decoding rate, i.e. the transmitter and receiver are always on even if the processing is done locally. In such a case, we may write the energy constraint in problem (64) as follows:

$$k_{tx,2} \frac{S_{UL,k}}{r_{UL,k}} \frac{2^{\frac{r_{UL,k}}{W\Gamma}} - 1}{\gamma_{UL,k}} + k_{rx,2} S_{DL,k} \le \alpha \varepsilon_{P,k} S_{app,k} \tag{65}$$

We can rearrange previous expression as follows:

$$\underbrace{\frac{k_{tx,2}}{r_{UL,k}} \frac{2^{\frac{r_{UL,k}}{W\Gamma}} - 1}{\gamma_{UL,k}}}_{f_1\left(r_{UL,k}\right)} \le \underbrace{\alpha \varepsilon_{P,k} \frac{S_{app,k}}{S_{UL,k}} - k_{rx,2} \frac{S_{DL,k}}{S_{UL,k}}}_{c_1} \tag{66}$$

It is easy to check that function $f_1(r_{UL,k})$ increases monotonically with the rate. As a result, to minimize the cost function in problem (64), we need to increase the UL data rate until either the constraint in (66) is fulfilled with equality or $r_{UL,k}$ reaches $R_{UL,k}$.

Case (ii): An important strategy to reduce the energy consumption is to induce the mobile terminal to a light sleep whenever it has not scheduled traffic [Wigard09]. If the user carries out the computation locally, the transmitter and receiver chains can be OFF with the consequent energy saving. If the user performs offloading, the energy required to have the transmitter and receiver chains ON must be considered in the energy consumption associated to the offloading. Let us assume that the resource allocation granularity is given by the minimum time the user equipment has to be ON before turning to a light sleep mode. In such a case, the time the transmitter and receiver chains need to be ON is given by $t_{tx,ON} = S_{UL,k} / r_{UL,k}$ and $t_{rx,ON} = S_{DL,k} / R_{DL,k}$, respectively. As a result, instead of (66), we need to consider the following constraint:

$$\underbrace{\frac{k_{tx,1}}{r_{UL,k}}+\frac{k_{tx,2}}{r_{UL,k}}\frac{2^{\frac{r_{UL,k}}{W\Gamma}}-1}{\gamma_{UL,k}}}_{f_2(r_{UL,k})} \leq \underbrace{\alpha\varepsilon_{P,k}\frac{S_{app,k}}{S_{UL,k}}-\left(\frac{k_{rx,1}}{R_{DL,k}}+k_{rx,2}\right)\frac{S_{DL,k}}{S_{UL,k}}}_{c_2}. \quad (67)$$

Case (iii): Finally, we are interested in considering the case in which each user receives $1/K$ of every UL and DL frame, instead of transmitting or receiving in bursts during several frames and remaining inactive while other users are served. If the user receives resources each frame, even if it is small portion of them, it seems unpractical to switch off the transmitter and receiver chains on a frame time basis. If the transmitter and receiver chains are ON during all the frames with scheduled traffic, that is for $t_{tx,ON} = K \cdot S_{UL,k} / r_{UL,k}$ and $t_{rx,ON} = K \cdot S_{DL,k} / R_{DL,k}$, the constraint to be considered in this case is:

$$\underbrace{\frac{k_{tx,1}K}{r_{UL,k}}+\frac{k_{tx,2}}{r_{UL,k}}\frac{2^{\frac{r_{UL,k}}{W\Gamma}}-1}{\gamma_{UL,k}}}_{f_3(r_{UL,k})} \leq \underbrace{\alpha\varepsilon_{P,k}\frac{S_{app,k}}{S_{UL,k}}-\left(\frac{k_{rx,1}K}{R_{DL,k}}+k_{rx,2}\right)\frac{S_{DL,k}}{S_{UL,k}}}_{c_3} \quad (68)$$

Unlike previous cases, in this case the number of users in the system affects the energy saving, and not only the latency experienced by each user.
Both functions $f_2(r_{UL,k})$ and $f_3(r_{UL,k})$ are quasi-convex, i.e., each one of them has a single minimum or none. As a result, the UL data rate should be the maximum data rate that fulfills the energy constraint (provided that this value is less than or equal to $R_{UL,k}$). This means that the optimum UL date rate is given by:

- $r^*_{UL,k} = R_{UL,k}$, if the function f_i decreases monotonically until $R_{UL,k}$,
- $r^*_{UL,k} = f_i^{-1}(c_i) \leq R_{UL,k}$, if the function increases after a certain point that can be zero if the function is monotonically increasing. f_i^{-1} denotes the inverse function of f_i.

Having selected $r^*_{UL,k}$, if the latency experienced by the user is greater than an acceptable value, then the user may still decide to offload but reducing the target energy saving 1-α. Eventually, the user may take the decision of not doing offloading if 1-α is not high enough.

For illustrative purposes, we consider two different applications from [Miettinen10]: the compression algorithm, gzip, and the x264 video encoder that require to 330 and 1900 instructions/byte, respectively. For both of them, we consider $S_{DL,k} / S_{UL,k} = 0.8$. The maximum transmission power for both the terminal and the SCeNB is 200 mW. The terminal processor corresponds to a commercial model that performs 480×10^6 instructions per Joule [Miettinen10] when working at a clock speed of 250 MHz. We assume that the remote processor is 20 times faster than the local one. We have considered a flat fading channel with γ_{UL} and γ_{DL} equal to 25 dB.

The results obtained correspond to the three cases described above: (i) receiver and transmitter are always ON and the extra energy required for the offloading comes from the actual transmission and reception; (ii) receiver and transmitter are only ON for those frames with traffic scheduled but each user transmits or receive in bursts of several frames followed by an inactivity period while the other users are transmitting or receiving, (iii) receiver and transmitter are ON for those frames with traffic scheduled but each user receives one part out of *K* of each frame.

Figure 107 shows the latency versus the target energy saving for different system loads, i.e. active users, considering that the receiver and transmitter are always ON (case (i)). Solid lines correspond to full rate granularity while solid dotted lines correspond to the case where only a subset of rates is

possible (obtained from the 15 LTE MCS values ranging from 0.1523 to 5.5547). As expected, the greater the targeted saving is, the greater the total latency is. For a concrete target energy saving, the latency increases as the number of users increases. As a result, we may need to increase the UL rate at the cost of reducing the energy saving to keep a certain quality of experience as the number of users in the system increases. For instance, for 10 active users, we may reduce the latency to a half by reducing the energy saving from 90% to 85%. Notice that there is a limit on the energy saving. This is because decreasing the UL data rate beyond a certain value does not bring a further energy saving.

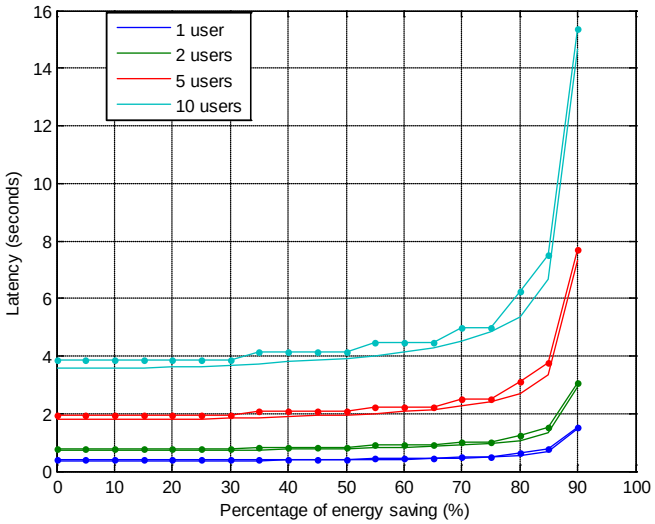


Figure 107. Latency versus target energy saving for *K*=1, 2, 5, and 10 users when receiver and transmitter are always ON (case (i)). Application complexity: 330 instructions/byte.

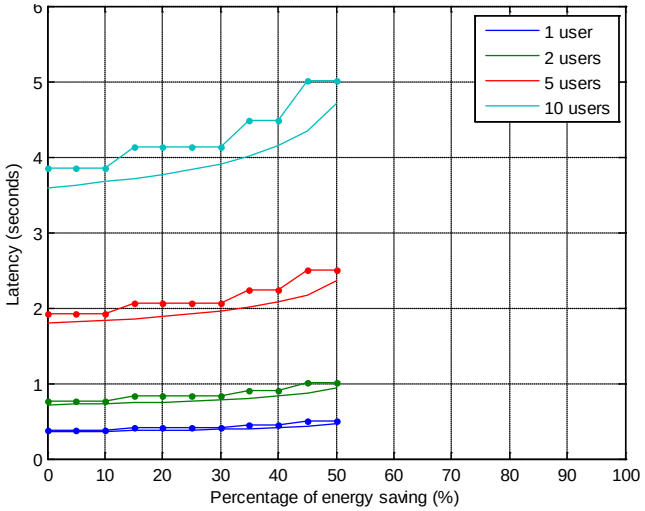


Figure 108. Latency versus target energy saving for *K*=1, 2, 5, and 10 users. Communication chains ON if traffic is scheduled (case (ii)). Application complexity: 330 instructions/byte.

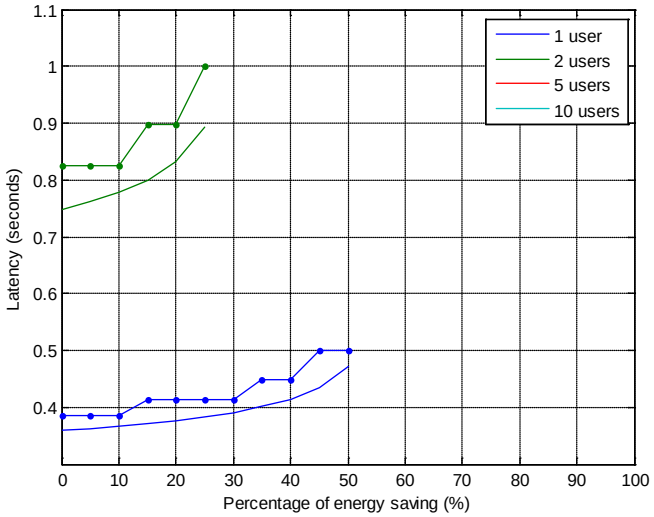


Figure 109. Latency versus target energy saving for *K*=1, 2, 5, and 10 users. Communication chains ON if traffic is scheduled. Each user receives 1/*K* of each frame (case (iii)). Application complexity: 330 inst./byte.

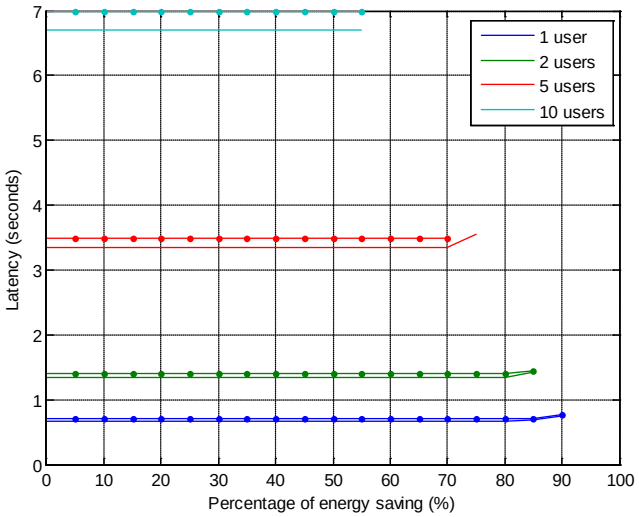


Figure 110. Latency versus target energy saving for *K*=1, 2, 5, and 10 users. Communication chains ON if traffic is scheduled. Each user receives a portion of each frame (case (iii)). Application complexity: 1900 inst./byte.

When considering micro sleep, there is an additional energy consumption compared to the case of no offloading, as in this case the transmitter and receiver chains may be switched off when doing local computation. This is the case shown in Figure 108 (case (ii)) and Figure 109 (case (iii)). Now, as the number of users increases, the actual throughput decreases and the user must be ON a longer time to send or receive the data. As a result, as the system load increases, it may happen that offloading is not beneficial any more. This is particularly important when the user receives a portion of each frame (Figure 109), so the transmitter and receiver cannot be switched off until the end of the whole transmission and reception, respectively.

For more computationally intensive applications as in Figure 110, the energy is less sensitive to the UL rates because of the fact that even for high UL rates (and, therefore, transmission power) we may still save a lot with respect to the case of no offloading. Now, even if the user has scheduled traffic at each frame and cannot switch off the transmitter and receiver until the end of the whole transmission and reception respectively (case (iii)), the energy saving is so important that offloading is worthy even for high system loads.

In summary, when micro sleep is allowed, as the system load increases, the offloading may not be worthy as the extra energy required for having the transmitter and receiver chains ON increases (cases (ii) and (iii)). In order to reduce this effect, it is much better to allow transmission in bursts (case (ii)), switching off transmitter and receiver between bursts, rather than transmitting during a part of each frame (case (iii)), as the transmitter and receiver cannot be switched on and off at each frame.

5.2.1.1.3 Extension to multiple VMs

Problem (57) defines a scenario where the serving SC is the only element able to process the offloaded bits. In a general setup, multiple VM's within the cloud may be available to process the offloaded bits that are transmitted through the serving SC. In this sense, a proper distribution of such bits among the VM's should be carried out. A graphical representation of the considered scenario is shown in the following figure:

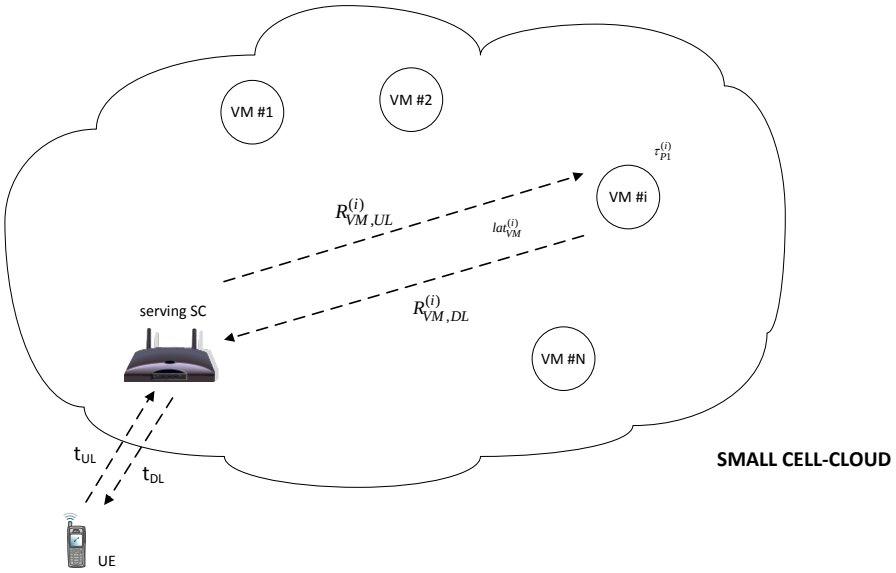


Figure 111. General offloading setup with multiple VM's available at the SC-cloud. In this setup, in addition to the communication link between the UE and the serving SC, the backhaul links between the serving SC and the VM's should be considered explicitly.

The generalization of the optimization problem (57) to encompass the presence of multiple VM's can be formulated as follows:

$$
\begin{aligned}
\underset{s_{P0},s_{P1},t_{UL},t_{DL},\{s_{P1}^{(i)}\}}{\text{minimize}} \quad & e_{UL}\left(t_{UL},\beta_{UL}s_{P1}\right)+\varepsilon_{P0}s_{P0}+e_{DL}\left(t_{DL},\beta_{DL}s_{P1}\right) \\
\text{s.t.} \quad & \text{C1: } \tau_{P0}s_{P0} \le L_{\max}, \\
& \text{C2: } s_{P0}+s_{P1}=S_{app}, \\
& \text{C3: } e_{UL}\left(t_{UL},\beta_{UL}s_{P1}\right)-k_{tx,1}t_{UL} \le k_{tx,2}t_{UL}P_{tx,UE}, \\
& \text{C4: } \beta_{DL}s_{P1} \le t_{DL}R_{DL}^{\max}, \\
& \text{C5: } \sum_i s_{P1}^{(i)} = s_{P1}, \\
& \text{C6: } s_{P1}^{(i)}\left(\frac{\beta_{UL}}{R_{VM,UL}^{(i)}}+\frac{\beta_{DL}}{R_{VM,DL}^{(i)}}+\tau_{P1}^{(i)}\right)+lat_{VM}^{(i)}+t_{UL}+t_{DL} \le L_{\max} \qquad \forall i.
\end{aligned}
\tag{69}
$$

In the previous problem, $s_{P1}^{(i)}$ represents the number of bits to be processed by the i-th VM, whereas s_{P1} is the total number of bits offloaded by the UE to the SC-cloud. According to this, constraint C5 indicates that the offloaded bits have to be distributed completely among all the VM's available in the

SC-cloud. Constraint C6 captures the latency constraint associated to the processing of $s_{P1}^{(i)}$ offloaded bits in the i-th VM. Such latency includes the data rate of the backhaul link between the serving SC and the i-th VM both in UL and DL (denoted by $R_{VM,UL}^{(i)}$ and $R_{VM,DL}^{(i)}$, respectively), the fixed total latency $lat_{VM}^{(i)}$ of such link, and the time $\tau_{P1}^{(i)}$ needed to process one bit in the i-th VM. The parameters β_{UL} and β_{DL} are associated to the overhead corresponding to the transmission of the input and output bits through the backhaul links. Note that, according to C6, the transmission of all the processed bits through the DL from the serving SC to the UE will begin once the bits processed by all the VM's have been recollected at the serving SC. The previous formulation of the problem includes as a particular case the situation in which the serving SC also hosts a VM to process offloaded bits. In such a case, the parameters associated to such VM would be: $R_{VM,UL}^{(i)}=\infty$, $R_{VM,DL}^{(i)}=\infty$, $lat_{VM}^{(i)}=0$, $\tau_{P1}^{(i)}=\tau_{P1}$. This would allow reformulating the corresponding constraint in C6 as

$$s_{P1}^{(i)}\tau_{P1}+t_{UL}+t_{DL}\leq L_{\max} \tag{70}$$

which is included implicitly in constraint C1 in problem (57).

Problem (69) is convex and, therefore, can be solved numerically using standard optimization tools with an affordable computational complexity while guaranteeing that the optimum solution is found. At this point it is important to remark that, when checking the possible distributions among VM's, a set of hypothesis have to be enumerated in terms of which is the subset of VM's considered for possible offloading. For example, if take a SC-cloud with 5 VM's, we may consider that only 3 of them will be used for processing the offloaded data. In that case, the constraints C6 associated to the two inactive VM's should not be included in the optimization problem. According to this, in general terms, this means that, if the number of VM's is N, the total number of hypothesis to be tested is 2^N-1 (at least one VM should be active). For each of this possible hypothesis, problem (69) has to be solved. Finally, within all the possible hypothesis that lead to a feasible optimization problem, the one that provides the lowest value of the objective function should be taken as the solution to the problem.

The following figures show the impact of having several virtual machines available for offloading purposes. To account for the impact of a non-ideal backhaul, the parameter $lat_{VM}^{(i)}$ has been set to 1 second for all VM except for the one allocated in the serving SC. For the sake of simplicity we have considered however $R_{VM,UL}^{(i)}=\infty$, $R_{VM,DL}^{(i)}=\infty$, $\tau_{P1}^{(i)}=\tau_{P1}$. The rest of the parameters are $\varepsilon_{P0}=8.6\cdot10^{-8}$ J/bit, $\tau_{P0}=10^{-7}$ s/bit, $k_{tx,1}=0.4$ W, $k_{tx,2}=18$, $k_{rx,1}=0.4$ W, $k_{rx,2}=2.86\cdot10^{-3}$ W/Mbps, $\tau_{P_1}=\tau_{P_0}/2$, $S_{app}=5$ Mbytes, $\beta_{UL}=1, \beta_{DL}=0.2$, $W_{UL}=W_{DL}=10$ MHz, $P_{tx,UE}=P_{tx,SC}$=100mW, $MCS_{\min}=0.15$ bits/symbol, and $MCS_{\max}=5.55$ bits/symbol For a channel gain of 25 dB, the following figures show the energy consumption at the UE, the percentage of file to be processed remotely, the total time spent, and the selected UL transmission rate. In all cases the horizontal axis corresponds to the maximum latency allowed for the application to be completed.

Let us consider first the case where the offloading decision is taken to minimize the total energy consumption of the UE due to the offloading considering that the transmitter and receiver can be switched off without scheduled traffic.

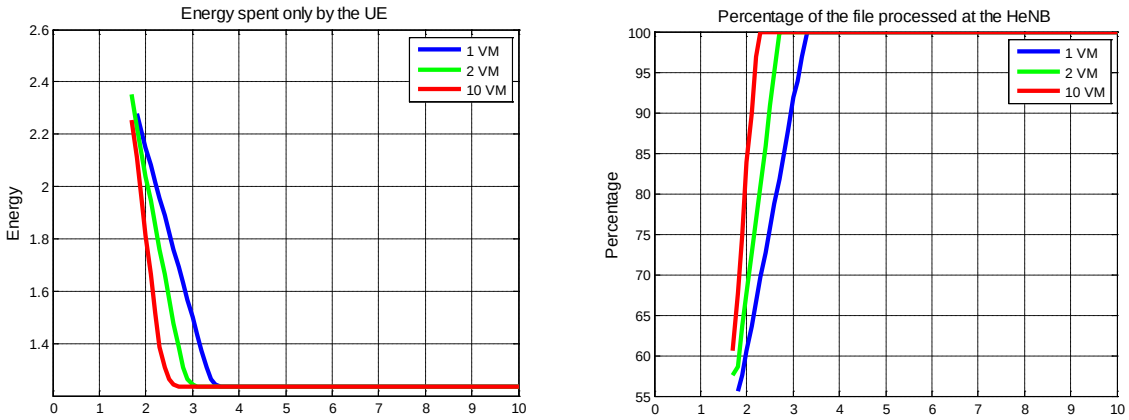


Figure 112. a) Energy consumed by the UE. b) Percentage of the file to be processed remotely. In both cases, the energy to be minimized includes the energy consumption for having the transmitter and receiver switched on.

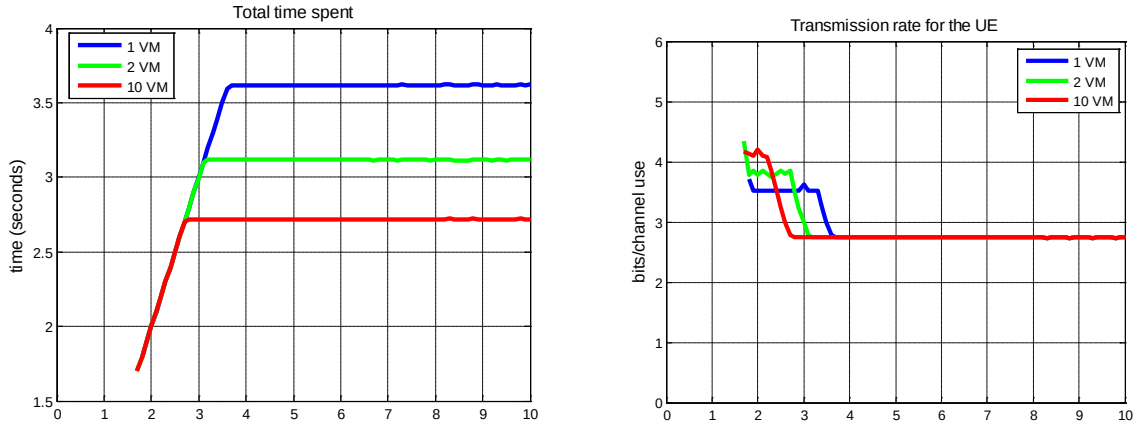


Figure 113. a) Total time spent for the completion of the application. b) Optimum UL transmission rate. In both cases, the energy to be minimized includes the energy consumption for having the transmitter and receiver switched on.

We observe that when the system is latency constrained, i.e. when the total time is equal to the maximum latency constraint, having several virtual machines impact on the total energy consumption. This due to the fact that having several virtual machines allows for reducing the time required for the remote processing, and therefore the uplink transmission time can be increased (or equivalently, the UL transmission rate can be decreased). If operating at the UL transmission rate was not possible with one VM because of the latency constraint, increasing the number of virtual machines will allow for reducing the energy consumption. When the UL transmission rate for minimum energy fulfils the latency constraint even if we have only one VM, then increasing the number of virtual machines does not have an impact on the energy consumption of the UE, but it has an impact on the experienced latency, as it can be observed in Figure 113a.

Let us consider now the case where the transmitter and receiver cannot be switched off even if there is not scheduled traffic. This baseline power consumption brings, of course, energy consumption. However, as it this energy will be consumed anyway, we do not consider this energy contribution in the optimization problem, i.e. $k_{tx,1} = 0$ and $k_{rx,1} = 0$.

In this case, reducing as much as possible the UL data rate (i.e. increasing as much as possible the uplink transmission time) will reduce the energy consumed by the UE. The uplink transmission time will be therefore increased until the latency constraint is fulfilled with equality, as it can be on

observed in Figure 115a. As a result, the number of virtual machines does not have an impact now on the total time spent, but it does have an impact on the energy consumed by the UE. We observe, that the energy consumption is smaller for a higher number of virtual machines, but the energy saturates anyway to the same value when the maximum latency constraint increases. This saturation value is given by the energy consumption required for decoding in the DL that depends on the amount of data received but not on the DL rate.

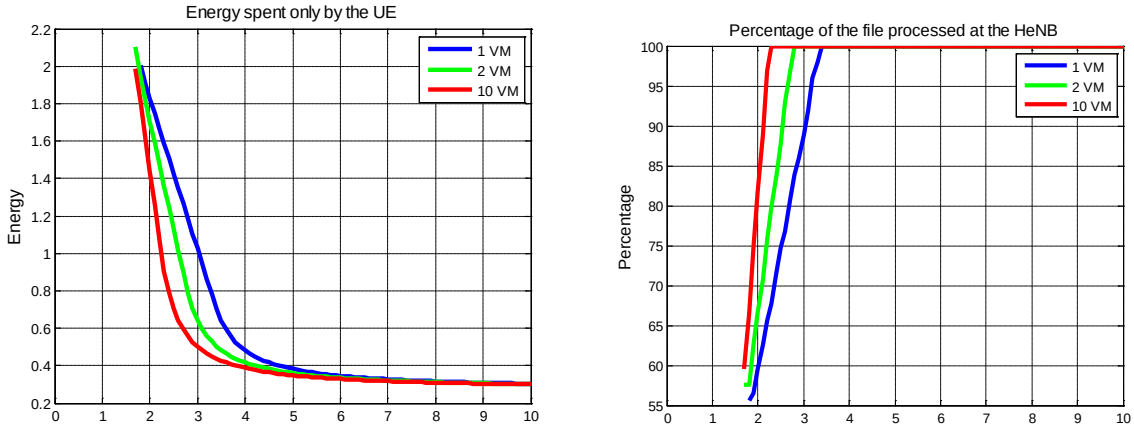


Figure 114. a) Energy consumed by the UE. b) Percentage of the file processed remotely. In both cases, the energy to be minimized does not include the energy consumption for having the transmitter and receiver on, i.e. $k_{tx,1} = 0$ **and** $k_{rx,1} = 0$**.**

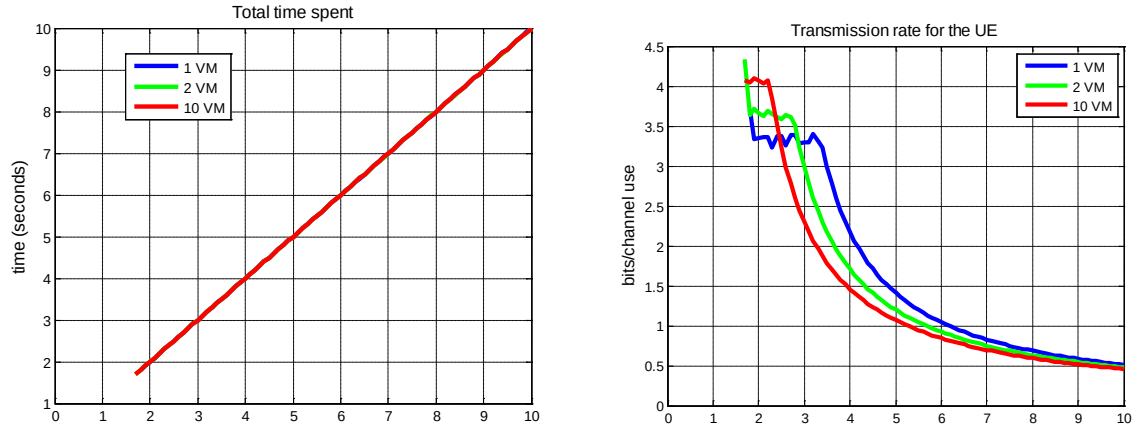


Figure 115. a) Total time spent. b) Optimum UL transmission rate. In both cases, the energy to be minimized does not includes the energy consumption for having the transmitter and receiver on, i.e. $k_{tx,1} = 0$ **and** $k_{rx,1} = 0$**.**

5.2.1.2 *Application aware scheduling: Resource allocation and Offloading*

5.2.1.2.1 Offloading Protocol

In order to leverage cloud computation resources with a mobile cellular network, appropriate signalling and architectural components have to be implemented in both realms: cloud computation infrastructure and radio access network. The signalling will depend on the remote execution model that will be adopted. Two main remote execution models can be defined:

- Transparent execution: this model is transparent to the mobile users, who are not aware about the cloud resources, only the radio access network has access to computation resources. With this model, not all applications can take benefits from the cloud computation capabilities that are available in the cloud.

- Application aware mobile applications: with this model, the applications that are run on the mobile user equipments are aware about the availability of remote computation capabilities. This model allows more degrees of freedom to jointly optimize the radio communication parameters and the processing portioning between the local user processor and the cloud computation infrastructure. This model can be implemented with a remote procedure call.

5.2.1.2.1.1 *Remote procedure call model*

With this model, a program is divided into a finite or infinite series of procedure calls. These procedures, also denominated methods or functions depending on the programming environments, are run by the terminal. The user equipment makes the decision to offload or not a certain function or procedure after a specific request to the small cell cloud manager who has a complete view of radio parameters and cloud computation resource availability. Upon a positive feedback, the procedure execution request is sent to the concerned computation resources, which can be one or many small-cell equipments, or even external computation devices.

Depending on the input and the output of the offloadable procedures, we can distinguish three main categories:

1. Remote data, local result: The input of the procedure is a chunk of data available in the Internet, the output is the result of processing this data by the computation resource: video decompression, re-encoding to fit the display device, file compression/decompression are typical example of this type.
2. Local data, local result: The input data is located in the mobile user equipment, this data is sent to the computation resource, and the output is the result of the processing of this chunk of data. Image processing, local document processing and cryptographic operations are typical examples of this type of remote processing.
3. Remote data, remote result + local result: The input data and the output data are located in an external storage resource; the user equipment receives only a partial view of the processed data: this is the case of remote document processing where the user equipment receives only the display of a certain part of the processed file.

5.2.1.2.1.2 *Architectural components and signalling strategies*

The main architectural component to support remote procedure call in a mobile network is the Small cell Cloud Manager. This device must have a complete knowledge of the radio parameters and the backhaul conditions of the set of SCeNBce it controls. In an LTE framework, a dedicated messaging system needs to be implemented between the MME gateways, the SCeNBs-ce and the SCM.

The SCM receives offloading requests and upon positive feedback it allocates computing resource in the concerned SCeNBce and asks the UE to create dedicated radio bearers that accommodates the type of traffic that is needed by the remote procedure call.

Since, most applications will be IP enabled, this requires that SCeNBs-ce be reachable via IP protocol. In order to get this connectivity, two approaches can be adopted:

- The SCeNBsce are seen as an external IP peers, and the S-GW makes the routing optimization in order to avoid unnecessary paths.
- Each SCeNBce includes its own serving gateway which will make the handover procedure more complex.

A typical message exchange between a UE, the small cell manager and the cloud enabled small cell eNBs is shown in Figure 116.

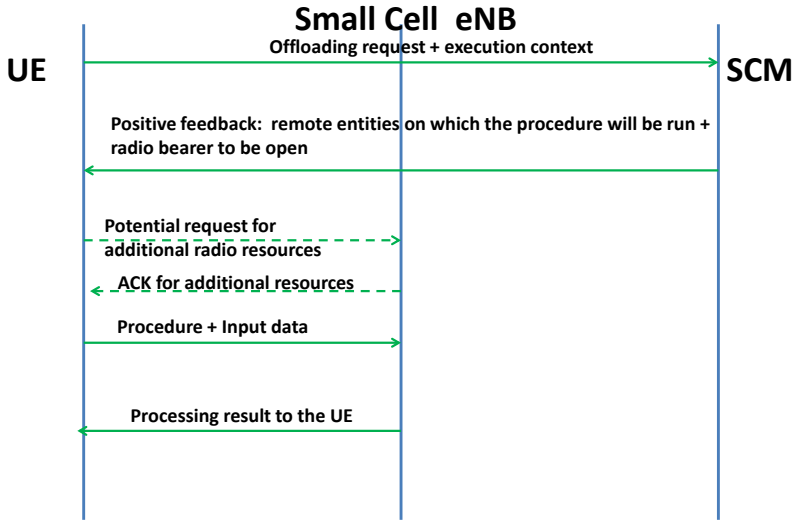


Figure 116. Typical signalling for remote procedure call initiated by a UE.

5.2.1.2.1.3 *Application aware scheduling*

With cloud enabled radio access network, the scheduling task is distributed between the small cell cloud manager and the cloud enabled SCeNBs. Following the spirit of LTE standard, radio bearers are created between the UE and the network gateway (P-GW) in order to meet predefined quality of service requirements, and the radio resource allocation and scheduling are left to the eNB in an autonomous, yet distributed, way. That's why the scheduling strategies are executed in two levels: application level and radio level.

The application level scheduling consists in deciding offloading a certain procedure to one or more SCeNBces. This decision takes into account a global view of the radio environment and the computation resource availabilities. It ensures that the stream of data generated by the offloading process can be supported by the involved devices.

The radio level scheduling is performed by the impacted eNBs which allocate appropriate resources so that to fulfill the quality of service promised upon the offloading decision. A specific care is dedicated to the quality of service expressed in delay constraints and battery level. These are among the most important parameters which impact the quality of experience seen by the end user.

We start first by describing the adopted radio scheduling strategies, and then integrate the offloading processing in the scheduling approach.

5.2.1.2.1.3.1 Radio scheduling strategies

In the Small Cell-Cloud context, energy-efficiency is a key issue for mobile users that are more and more application-hungry, sharing delay-sensitive contents (for example, real-time multimedia streaming, videoconferencing, gaming ...). Such applications and services impose requirements in terms of data rates and latency, and on the other hand consume battery for transmission and processing. In addition, these delay-sensitive communications usually operate in dynamic environments experiencing time-varying fading channels and dynamic traffic loads (variable rates). Therefore, to tradeoff between energy-efficiency and low delay, the scheduling decisions should be adapted to the time-varying environment.

In this section, we address the problem of finding the optimal scheduling policy that minimizes the energy consumption at the mobile UE (required for reliable data transmission) under delay constraints. While one approach consists in resolving the optimization problem for instantaneous channel

realization and under instantaneous delay constraint as performed in Section 5.2.3.1 we adopt in the sequel an online learning approach to compute the optimal scheduling policy for a single user fading channel. Such techniques learn online, at a given transmission time, the best action to be decided that minimizes the average power while satisfying an average delay constraint for the whole communication. Therefore, learning from the past allows to quickly adapt the scheduling decisions to the a priori unknown channel and traffic conditions.

a- Single User System model

We consider a point-to-point communication between one UE and one SCeNB over a fading wireless channel. The transmitter buffer of the user has a maximal size of B packets. The time is divided into time slots of duration ΔT ($\Delta T = 1$msec in LTE context for example). At time slot *n*, let $Q_n \left(0 \le Q_n \le B\right)$ denote the buffer state, U_n the number of packets transmitted at slot $n \left(U_n \le Q_n\right)$ and A_{n+1} the number of arrivals at the beginning of the slot *n+1*. Then, the buffer state at the beginning of the slot *n+1* can be expressed by:

$$Q_{n+1} = \max\left\{Q_n - U_n\right\} + A_{n+1} \tag{71}$$

We assume that there is no buffer overflow by imposing the following condition on the arrival packets $A_{n+1} \le B - Q_n$.

We consider a Rayleigh block fading channel having a complex gain H_n that remains constant within one time slot and changes independently across slots. The communication is disturbed by an additive white Gaussian noise N_n with zero mean and variance N_0. The user transmits a signal X_n in the slot *n* so the received signal by the base station is given by $Y_n = H_n X_n + N_n$. Let $Z_n = \left|H_n\right|^2$ be called the channel state at the slot *n*. H_n is an exponentially distributed random variable with a mean equal to θ. We suppose that the transmission is error-free; then from the Shannon capacity formula, we can obtain that the required transmission power in the slot *n* can be expressed as:

$$P\left(Z_n, U_n\right) = \frac{WN_0}{X_n}\left(2^{\frac{lU_n}{W}} - 1\right) \tag{72}$$

where l is the number of bits in a packet and W the transmission channel bandwidth. We denote the state of the described system $S_n = \left(Q_n, Z_n\right)$ depending on the queue length and the channel state. Assuming that Z_n can take values from a discrete space, then the state space of the system is also discrete finite space χ. We define the scheduling policy μ_n as the control policy that determines the number of packets U_n in time slot *n* to be transmitted upon the state S_n. The induced random process $\left\{S_n\right\}$ is a Markov chain since the conditional law of the state S_{n+1} of the dynamical system, given the past history of the system states and past controls up to time slot *n*, depends only upon the state on S_n and $\mu_n\left(S_n\right)$. Let $p\left(S_{n+1} \middle| U_n, S_n\right)$ denotes the state transition conditional probability of this Markov decision process. Assuming that the queue length and the channel state are independent, the probability of state transition can be expressed as:

$$p\left(S_{n+1} \middle| U_n, S_n\right) = p\left(Q_{n+1} \middle| U_n, Q_n\right) p\left(Z_{n+1} \middle| Z_n\right) = p\left(A_{n+1}\right) p\left(Z_{n+1} \middle| Z_n\right) \tag{73}$$

b- Problem Formulation

The objective is to find the optimal policy that minimizes the average transmission power subject to an average delay constraint. The average delay is related to the average queue length by Little's Law. The problem can be formulated as a Constrained Markov Decision Problem (CMDP). Let us define

$L_P(S_n,U_n)=P(Z_n,U_n)$ as the cost in terms of power required to transmit U_n packets at state S_n and $L_D(S_n,U_n)=\frac{(Q_n-U_n+A_{n+1})l}{\bar{a}W\Delta T}$ as the delay cost related to buffering packets where $\bar{a}$ denotes the arrival rate. Under the policy μ, the average power cost and delay cost are expressed as:

$$\bar{P}^{\mu}=E^{\mu}\left[L_P\left(S_n,\mu(S_n)\right)\right]=\sum_{u,s}\rho^{\mu}(s)\mu(u|s)L_P\left(s,\mu(s)\right) \tag{74}$$

$$\bar{D}^{\mu}=E^{\mu}\left[L_D\left(S_n,\mu(S_n)\right)\right]=\sum_{u,s}\rho^{\mu}(s)\mu(u|s)L_D\left(s,\mu(s)\right) \tag{75}$$

where E^{μ} is the expectation w.r.t. the state distribution ρ^{μ} of the Markov chain $\{S_n\}$. Then, the scheduler objective is to find the optimal policy which resolves the constrained problem:

$$\min \bar{P}_{\mu} \quad \text{subject to} \quad \bar{D}_{\mu}\le\delta \tag{76}$$

c- Lagrangian Approach

This CMDP can be converted into an unconstrained one using the Lagrangian approach. Let define the function $L(\lambda,s,u)=L_P(s,u)+\lambda\left(L_D(s,u)-\delta\right)$. For a given Lagrangian Multiplier (LM) $\lambda>0$, the unconstrained MDP is to determine the optimal policy μ^* that minimizes the Lagrangian:

$$\Gamma(\mu,\lambda)=\bar{P}^{\mu}+\lambda\left(\bar{D}^{\mu}-\delta\right) \tag{77}$$

The optimizing policy for this MDP is obtained by resolving the Bellman equation [Bertsekas07] based on dynamic programming:

$$V(s)=\min_{u}\left[L(\lambda,s,u)-\beta+\sum_{s'}p(s,u,s')V(s')\right] \quad s'\in\chi \tag{78}$$

where β characterizes the corresponding optimal cost per stage (the cost after we apply the optimal policy μ^*). In addition, using standard optimization theory [Altman99], the constrained problem has a stationary optimal policy that is optimal for the unconstrained problem for a particular choice of the LM $\lambda=\lambda^*$ that satisfies the average delay constraint. Then, the optimal transmitted packets $u^*(s)$ verifies:

$$u^*(s)=\arg\min_{u}\left[L(\lambda,s,u)-\beta+\sum_{s'}p(s,u,s')V(s')\right] \quad \forall s\in\chi \tag{79}$$

For a fixed λ, the relative value iteration (RVI) algorithm resolves the Bellman equation of the unconstrained MDP in an iterative manner. However, this algorithm requires the knowledge of the transition probabilities $p(s,u,s')$ from a state to another. These probabilities depend on the arrival and the channel state distributions that are, in real system, often unavailable *a priori*. In order to cope with transmission in an unknown environment, a post-decision state-based learning approach was proposed in [Salodkar08] and will be followed in the sequel.

d- Post-Decision State based Learning Algorithm

The post-decision state (PDS) is defined as the state reached by the system after taking the decision to transmit U_n and before unknown dynamics take place, *i.e.* before new data packets A_{n+1} arrive and next channel state Z_{n+1} is realized. Thus, the PDS at time *n* is related to the state $S_n=(Q_n,Z_n)$ of the system at time *n* and the action U_n that is taken leading to the transition from state S_n to the state S_{n+1}. This PDS is denoted by $\tilde{S}_n=(Q_n-U_n,Z_n)$ and the next state at time *n+1* called also pre-decision state

is defined as: $S'_{n+1}=(Q',Z')=(Q_n-U_n+A_{n+1},Z_{n+1})$. Then, the PDS value function can be defined as the expectation over all the pre-decision states which could be reached from a post decision state $\tilde{S}_n$. For a PDS $\tilde{S}_n=(Q_n,Z_n)$ at time *n*, the PDS value function at time *n+1* satisfies the dynamic programming equation:

$$\tilde{V}_{n+1}(\tilde{S}_n)=\sum_{A_{n+1},Z_{n+1}} p(A_{n+1})p(Z_{n+1}|Z_n)\min_{U_{n+1}\le Q_n+A_{n+1}}\Big[L(\lambda,(Q_n+A_{n+1},Z_{n+1}),U_{n+1}) +\tilde{V}_n(Q_n+A_{n+1}-U_{n+1},Z_{n+1})\Big]-\tilde{V}_n(\tilde{S}_0) \quad (80)$$

where $p(A_{n+1})$ denotes the arrival distribution and $p(Z_{n+1}|Z_n)$ the channel state transition probability and a fixed $\tilde{S}_0$. This PDS value iteration equation is then resolved using online primal-dual RVI algorithm and following the theory of stochastic approximation [Kushner97]. The states of the post decision state function are updated as follows:

$$\tilde{V}_{n+1}(\tilde{S}_n)=(1-\alpha_n)\tilde{V}_n(\tilde{S}_n)+\alpha_n\Big[\min_{U_{n+1}}\Big[L(\lambda,(Q_n+A_{n+1},Z_{n+1}),U_{n+1})+\tilde{V}_n(Q_n+A_{n+1}-U_{n+1},Z_{n+1})\Big]-\tilde{V}_n(\tilde{S}_0)\Big] \quad (81)$$

where $\alpha_n\in[0,1]$ is a time-time varying learning rate factor [Sutton98] that satisfies $\sum_{n=1}^{\infty}\alpha_n=\infty$, $\sum_{n=1}^{\infty}(\alpha_n)^2<\infty$ e.g. $\alpha_n=1/n$. This method is "one state per time" update as it updates only PDS value function in the PDS $\tilde{S}_n$ that is visited at the current time slot, while the other states of the value function matrix remain unchanged. Based on this online algorithm, the optimal scheduling policy computes the number of packets to be transmitted according to equation (79).

e- Simulation Results

The online learning algorithm is evaluated for a Rayleigh fading channel where the channel state *Z* is an exponentially distributed random variable. The channel state is then quantized to get a finite channel space, thus a finite dimension value matrix. The arrival packets are modeled as i.i.d. random Poisson variables with a fixed arrival rate, and no overflow in the queue is assumed. We consider the channel bandwidth $W=5$ MHz and WN_0 is normalized to 1. The slot duration is 1 ms, the packet size $l=5000$ and the transmitter can transmit 1 to 8 packets in a slot.

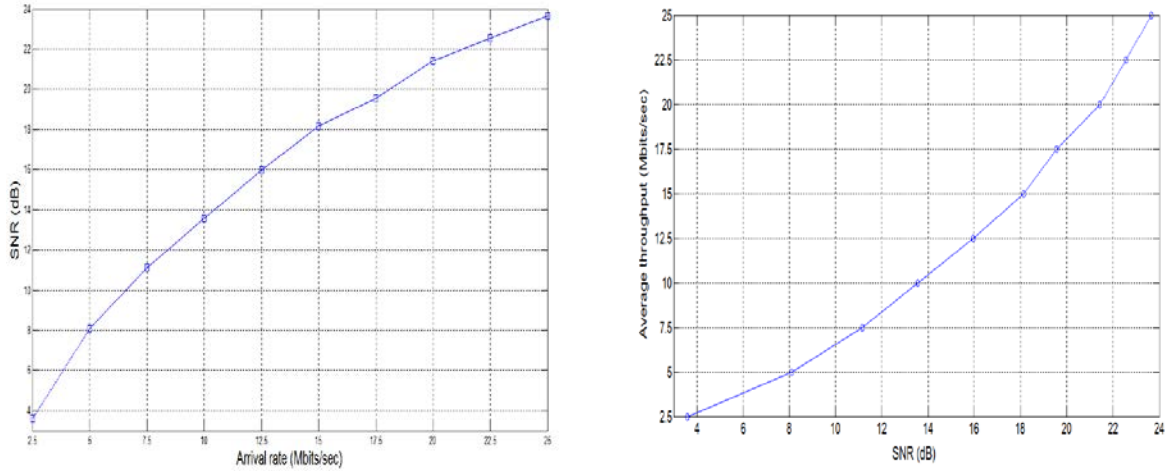


Figure 117. Required SNR vs Arrival Rate and Average Throughput.

The left side figure shows the required SNR at the receiver as function of the arrival rates under a fixed queue length constraint. It can be observed that when increasing the arrival rates, in order to satisfy this queue constraint, the scheduler transmits more packets with higher SNR, since more power is required to transmit these packets. From both figures, we can notice the delay/queue constraint is met since the transmission rate is equal to the arrival rate.

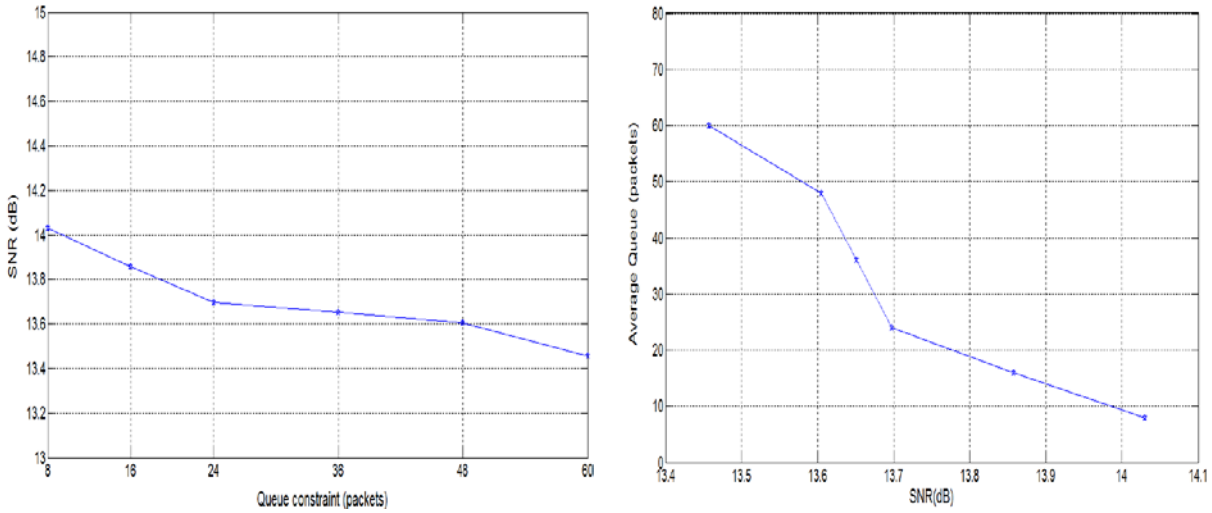


Figure 118. Required SNR vs Queue Constraint and Average Queue Length.

From the left figure, we can see that when increasing the queue constraint, the SNR decreases since the average transmitted power decreases. In fact, if this constraint is low, the queue length can exceed it quickly when arrival rate is high. Therefore, the scheduling policy will try to send the maximum of packets, without considering the channel conditions, such that the queue stability is reached. If the arrival rate is low, the control policy will be dependent also on the channel conditions to determine the amount of data to be transmitted.

5.2.1.2.2 Dynamic Programming for Joint resource allocation and offloading strategies in single user case

Now, we consider that the UE can benefit from cloud services offered by a nearby SCeNBce to offload his application (or a part of it) and hence reduce his energy consumption. In this case, the scheduling strategy takes into account the offloading decision at UE upon "request to" and "positive feedback of" the Small Cell Manager, aware of all the available radio and computation resources at the SCeNBces.

Therefore, the objective is to find the optimal offloading-scheduling policy based on offline strategies and online learning strategies in order to optimize the global energy consumption (locally at UE and remotely at SCeNBce) under average delay constraints imposed by the application.

5.2.1.2.2.1 *System Model*

We consider that the processing decisions are performed at successive time slots of duration ΔT. These periods can be aligned with the frame duration in LTE networks and with the instants ΔT where radio scheduling messages are executed. The communication channel between the UE and the SCeNBce is considered as Rayleigh fading channel with complex fading gain H_n. The channel remains constant during ΔT and changes in an independent and identically distributed (*i.i.d.*) manner across time slots. We define Z_n as the quantized channel state and W_n the additive complex white Gaussian noise with a zero mean and variance ${N_0}^2$. If offloading decisions are performed from the UE to the SCeNBce, then the UE transmits a signal y_n and the SCeNBce receives: $R_n = H_n y_n + W_n$.

The user's application data is defined in terms of data packets of equal size l bits. The packets' arrival follows a Poisson distribution with a fixed arrival rate $\bar{a}$ and these arrivals are *i.i.d.* These data packets are queued in the UE buffer until they are scheduled, either for local execution or for remote processing on the SCeNBce. The user buffer has a finite size of B packets. Let Q_n denote the queue length in the buffer *i.e.* the buffer state at a given time slot t_n, $Q_n \le B$. During a slot duration ΔT, A_{n+1} packets can arrive and this instantaneous arrival A_{n+1} is observed at t_{n+1}. Thus at t_{n+1}, the buffer state will be equal to $Q_{n+1} = \max\left(Q_n - U_n, 0\right) + A_{n+1}$. In addition, we consider that buffer overflow can occur when the queued packets exceed the buffer size, thus leading to packets loss. This event is defined with a probability of buffer overflow that should be kept under a predefined threshold δ_O.

The described buffer state information QSI and channel state information CSI define our system state as $S_n = \left(Q_n, Z_n\right)$ at time slot t_n. We assume that the state information is sent by the UE to the SCeNBce through channel quality indicators and buffer status reports as in LTE standards. Based on these status reports and on the available computation resources, the small cell manager SCM performs decisions upon user offloading requests to minimize the energy consumption at the UE terminal. Thus, at each time slot duration ΔT, three scheduling decisions are possible: Processing of U_n queued packets $\left(U_n \le Q_n\right)$ either by (1) local processing at the UE or by (2) computation offloading at the SCeNBce, or (3) staying idle *i.e.*, waiting until the next decision time to process.

According to the possible decisions, different costs can be generated at each slot t_n in terms of processing delay:

$$D_n = \begin{cases} \dfrac{U_n L_{\mathrm{UL}}}{W_{\mathrm{UL}} \log_2\left(1 + \dfrac{P_t Z_n}{W_{\mathrm{UL}} N_0}\right)} + \dfrac{U_n L_{\mathrm{DL}}}{W_{\mathrm{DL}} \log_2\left(1 + \dfrac{P_{SCeNB} Z_n}{W_{\mathrm{DL}} N_0^i}\right)} + U_n T_w & \text{if offloading} \\ U_n T_{UE} & \text{if local processing} \\ T_{idle} & \text{if no processing} \end{cases} \tag{82}$$

and energy consumed by the mobile terminal $E_n = L_E\left(Z_n, U_n\right)$:

$$E_n = \begin{cases} \dfrac{U_n L_{\mathrm{UL}}}{W_{\mathrm{UL}} \log_2\left(1 + \dfrac{P_t Z_n}{W_{\mathrm{UL}} N_0}\right)} P_t + \dfrac{U_n L_{\mathrm{DL}}}{W_{\mathrm{DL}} \log_2\left(1 + \dfrac{P_{SCeNB} Z_n}{W_{\mathrm{DL}} N_0}\right)} P_r + U_n T_w P_w & \text{If offloading} \\ U_n T_{UE} P_{UE} & \text{If local processing} \\ T_{idle} P_{idle} & \text{If no precessing} \end{cases} \tag{83}$$

In case of offloading of U_n packets, the UE would incur an extra overhead in terms of delay and energy for transmitting the data packets to be processed at the SCeNBce. Once the remote processing is completed, the SCeNBce will send back the resulting data to the UE. Accordingly, the energy consumed at the mobile terminal will include the energy spent for: data transmission, data reception and waiting the remote processing completion. Therefore, P_t is the power used at the UE to transmit the UL data, with $P_t \le P_{\max}$, $P_{\max}$ being the maximum transmission power of the UE. P_r is the power consumed by the UE to receive DL data. P_{SCeNB} is the transmission power used at the cloud enabled SCeNB to send the processed result to the UE. The transmitted data packets have equal size of $L_{\mathrm{UL}} = l$ bits and the received processed packets have equal size of L_{DL} bits. W_{UL} and W_{DL} are respectively the bandwidths allocated for these UL and DL transmissions. P_w is the power consumed by the UE while

waiting for the processing at the SCeNBce and T_w is the corresponding time spent by the SCeNBce to execute 1 offloaded packet. Note that for simplicity, only radiated power is considered in the power consumption model.

In case of local processing of U_n packets, the UE terminal will consume a power per packet equal to P_{UE} and accumulate a processing time per packet equal to T_{UE}. For idle mode, P_{idle} is the power consumed by the UE in this mode.

For all cases, since the decision time instants are scheduled at fixed duration ΔT then $D_n \leq \Delta T$. In case the decision is to wait, we have $T_{idle} = \Delta T$. For local processing, we can fix also $T_{UE} = \Delta T$ for simplicity such that the maximal number of packets to be processed locally is equal to 1. For the offloading case, the offloading decision can be possible only if there exists $P_t \in]0, P_{\max}]$ to transmit the offloaded packets $U_n \leq Q_n$ such that the processing at the SCeNBce is achieved within a time ΔT. Satisfying the equality $D_n = \Delta T$ allows to compute the transmit power P_t at the UE and define the maximal number of offloaded with respect to the channel conditions as $U_{\max} = f(Z_n)$:

$$U_{\max} = \max \left\{ U \; ; \; \Delta T - \left(\frac{U L_{\mathrm{DL}}}{W_{\mathrm{DL}} \log_2 \left(1 + \frac{P_{SCeNB} Z_n}{W_{\mathrm{DL}} N_0}\right)} + U T_{cloud} \right) > 0 \right\} \quad (84)$$

5.2.1.2.2.2 *Problem Formulation*

The scheduler determines the optimal scheduling-offloading policy μ at processing time instants that specifies the processing decisions, local processing or offloading or idle, and the corresponding number of packets to be processed locally or offloaded. The optimal policy aims at minimizing the energy consumption at the mobile terminal to process its applications while satisfying QoS requirements in terms of delay constraints imposed by these applications.

Let $U_n = \mu(S_n)$ be the optimal policy at t_n at the state S_n. The conditional law of the state S_{n+1} of this dynamic system depends only on S_n and the control μ_n, thus making it a discrete time Markov chain. Therefore, the scheduling-offloading optimization problem can be formulated as a constrained Markov decision process CMDP with finite states and actions spaces as following:

The state space: The state space is the set of the system states $s = (q, z)$. It varies in time and takes into account the buffer level of the UE and the channel condition. It takes values from finite channel and buffer spaces Z and Q, respectively. In our case, $\mathrm{Q} = \{0, ..., B\}$ and Z is discretized into 8 values. Thus, the state space is finite and has cardinality: $|\mathrm{S}| = |\mathrm{Q}| \times |\mathrm{Z}|$.

The action space: The action space is also a finite discrete space defined by $\Omega = \{-1, 0, \ldots, U_{\max}\}$. At decision time, the control action $u \in \Omega$ determines the number of packets that can be processed locally or remotely within time ΔT, $|u| \leq min\ (q, U_{\max})$. -1 denotes local processing of 1 packet.

The transition probabilities: The state transition probability of the Markov decision process is defined by $p(s' | s, u)$, the probability to go from a state $s = (q, z)$ to another state $s' = (q', z')$ when action $u \in \Omega$ is performed. It depends on the channel transitions and random arrivals that can occur . Assuming that the buffer state (packets' arrivals a) and the channel state are independent of each other and that the channel states are not correlated, the probability of state transition is defined by

$$p\left(s'|s,u\right)=p\left(q'|q,u\right)p\left(z'\right)=p\left(a\right)p\left(z'\right) \tag{85}$$

From [Altman99], the state space is unchained and there exists a class of stationary policies (randomized and deterministic) which are complete for the expected average cost criterion. μ is a stationary policy that does not depend on the time at which the decision is made. It is a mapping from the state space to the action space. In an infinite horizon, the control policy μ aims at minimizing the average consumed energy at the UE side while satisfying requirements in terms of average delay experienced by the UE packets before being processed and average buffer overflow constraint. The average delay constraint can be considered as an average queue length constraint (by Little's Law). Under strategy μ, the time averaged energy cost, the average delay constraint and the average buffer overflow constraint are defined on an infinite horizon by

$$\begin{aligned}\overline{E}&=\limsup_{N\to\infty}\ \frac{1}{N}\ E^{\mu}\left[\sum_{n=1}^{N}L_E(S_n,U_n)\right]\\ \overline{Q}&=\limsup_{N\to\infty}\ \frac{1}{N}\ E^{\mu}\left[\sum_{n=1}^{T}L_Q(S_n,U_n)\right]\\ \overline{O}&=\limsup_{N\to\infty}\ \frac{1}{N}\ E^{\mu}\left[\sum_{n=1}^{N}L_O(S_n,U_n)\right]\end{aligned} \tag{86}$$

The state space S is composed of a single communicating class. Thus, we can move from any state to another with a positive distribution ρ^{μ} under any stationary policy. In addition, all the states are aperiodic with finite and non-fixed hitting time, so they can be visited an infinite number of times during an infinite process. Therefore, $\{S_n\}$ is an ergodic Markov chain and has a unique stationary distribution ρ^{μ} [Bertsekas07]. Let E^{μ} denote the expectation w.r.t. ρ^{μ}. Under policy μ, the average energy cost and the average queue length cost can be expressed as:

$$\begin{aligned}\overline{E}^{\mu}&=E^{\mu}\left[L_E\left(S_n,\mu\left(S_n\right)\right)\right]=\sum_{S_n,U_n}\rho^{\mu}\left(S_n\right)\,w^{\mu}\left(U_n|S_n\right)\,L_E\left(S_n,U_n\right)\\ \overline{Q}^{\mu}&=E^{\mu}\left[L_Q\left(S_n,\mu\left(S_n\right)\right)\right]=\sum_{S_n,U_n}\rho^{\mu}\left(S_n\right)\,w^{\mu}\left(U_n|S_n\right)\,L_Q\left(S_n,U_n\right)\\ \overline{O}^{\mu}&=E^{\mu}\left[L_O\left(S_n,\mu\left(S_n\right)\right)\right]=\sum_{S_n,U_n}\rho^{\mu}\left(S_n\right)\,w^{\mu}\left(U_n|S_n\right)\,L_O\left(S_n,U_n\right)\end{aligned} \tag{87}$$

where $L_E\left(S_n,\mu\left(S_n\right)\right)$ is the energy cost at the state S_n with $L_E\left(S_n,U_n\right)=L_E\left(Z_n,U_n\right)$ as defined previously according to the possible processing decisions with the control policy $U_n=\mu\left(S_n\right)$. $L_Q\left(S_n,U_n\right)$ is the delay/queuing cost with $L_Q\left(S_n,U_n\right)=L_Q\left(Q_n\right)=Q_n$. $L_O\left(S_n,U_n\right)$ is the buffer overflow cost with $L_O\left(S_n,U_n\right)=L_O\left(Q_n,U_n\right)=1_{(Q_n>B)}$. $w^{\mu}\left(U_n|S_n\right)$ is the probability of performing the control action U_n at the state S_n.

Then, the scheduler objective is to find the optimal strategy μ^* which resolves the constrained problem by minimizing the average energy consumed at the UE while satisfying an average delay on the processed packets, *i.e.*, a queue length below a certain value δ_Q, and a buffer overflow probability under a fixed threshold δ_O. The optimization problem can be stated as

$$\begin{aligned}\mu^*&=\min_{\mu}\quad\overline{E}^{\mu}\\ \text{subject to}\ \overline{Q}^{\mu}&\le\delta_Q\quad\text{and}\quad\overline{O}^{\mu}\le\delta_O\end{aligned} \tag{88}$$

Two main approaches have been proposed to resolve an infinite horizon average cost problem [Bertsekas07, Altaman99, Salodkar08]: offline and online dynamic programming strategies. In the following, we investigate both solutions to solve our problem.

5.2.1.2.3 Offline Dynamic Programming Approach

The offline approach relies on *a priori* knowledge of the channel statistics and the application properties (rate of generated data). It offers the advantage of optimality when these conditions are met. In fact, in many cases, the arrival rate of packets can be known or estimated in advance with a certain accuracy. Besides, in low mobility scenarios the channel variations can be acquired over a sufficiently long period before running the application. When such information is available, pre-calculated offline strategies offer a promising option for optimization problem. For instance, the SCeNB cloud manager can pre-calculate an application dependent strategy for various values of arrival rates and for common channel distributions. A dedicated signalling can be devised inside the application so that to fetch the strategy from the cloud manager before starting to process the data. This option allows to minimize the signalling overhead caused by useless offloading requests which receive negative answer. In this way, the offloading decision can be always made by the application after an initial approval by the small cell cloud manager SCM. Two offline strategies are studied in this section, deterministic offline strategy and randomized offline strategy.

5.2.1.2.3.1 *Deterministic offline strategy*

The deterministic offline policy consists in a deterministic mapping from the state space S to the action space Ω, associating for each state s a unique action u that is performed when this state is visited. The scheduling-offloading policy μ determines the number U_n of packets to be processed as $U_n = \mu(S_n) \;\; \forall n \in \mathbb{N}$ with $w^{\mu}(U_n | S_n) = 1$.

In order to find the optimal deterministic policy μ^*, the CMDP problem can be converted into an unconstrained problem using Lagrangian approach as

$$\begin{aligned} L_{av}(\mu, \lambda_1, \lambda_2) &= \bar{E}^{\mu} + \lambda_1 \left(\bar{Q}^{\mu} - \delta_Q \right) + \lambda_2 \left(\bar{O}^{\mu} - \delta_O \right) \\ &= \limsup_{N \to \infty} \frac{1}{N} E^{\mu} \left[\sum_{n=1}^{N} L_E(S_n, U_n) + \lambda_1 \left(L_Q(S_n, U_n) - \delta_Q \right) + \lambda_2 \left(L_O(S_n, U_n) - \delta_O \right) \right] \\ &= \limsup_{N \to \infty} \frac{1}{N} E^{\mu} \left[\sum_{n=1}^{N} L_E(S_n, U_n) + \lambda_1 \left(Q_n - \delta_Q \right) + \lambda_2 \left(1_{Q_n > B} - \delta_O \right) \right] \end{aligned} \tag{89}$$

where λ_1 and λ_2 are the Lagrange multipliers corresponding to the delay and overflow constraints, and an instantaneous cost per stage is defined as

$$L(S_n, U_n, \lambda_1, \lambda_2) = L_E(Z_n, U_n) + \lambda_1 \left(Q_n - \delta_Q \right) + \lambda_2 \left(1_{Q_n > B} - \delta_O \right) \tag{90}$$

As the energy is a convex function of the number of packet sent, the optimal policy μ^* for the constrained problem is also optimal for the unconstrained problem. μ^* minimizes the average Lagrangian $L_{av}(\mu, \lambda_1, \lambda_2)$ for a particular choice of the Lagrange multiplier (λ_1, λ_2). The Lagrange multipliers are parameters which characterize the flexibility of the scheduler respectively to the buffering constraints and the overflow constraints.

However, the objective is to determine the optimal policy μ^* and optimal Lagrange multipliers $(\lambda_1^*, \lambda_2^*)$ such that the saddle point optimality condition can be satisfied [Salodkar08]:

$L\left(\mu^*,\lambda_1,\lambda_2\right)\le L\left(\mu^*,\lambda_1^*,\lambda_2^*\right)\le L\left(\mu,\lambda_1^*,\lambda_2^*\right)$ with $\left(\mu^*,\lambda_1^*,\lambda_2^*\right)$ is a saddle point for the Lagrange average cost $L_{av}\left(\mu,\lambda_1,\lambda_2\right)$.

The minimization problem of this MDP is handled by solving dynamic programming equations, known as Bellman optimality equations [Bertsekas07] using relative value iteration algorithm RVIA. This transforms the problem over the class of all policies into a set of coupled minimization problems over smaller sets of actions. For a fixed value of the Lagrange multipliers $\left(\lambda_1,\lambda_2\right)$, the optimal value function $V^*_{\lambda_1,\lambda_2}\left(s\right)$ is then expressed by this Bellman equation as:

$$V_{\lambda_1,\lambda_2}\left(s\right)=\min_{u\in\Omega}\left[L\left(s,u,\lambda_1,\lambda_2\right)+\sum_{s'}p\left(s'\middle|s,u\right)\,V_{\lambda_1,\lambda_2}\left(s'\right)-\beta\right] \tag{91}$$

where $s,s'\in\chi$ and β denotes the optimal average cost per stage. It represents the average cost per stage of the system according to an optimal policy μ^* along all the possible trajectories that can occur.
To converge to the optimal policy for a particular choice of $\left(\lambda_1,\lambda_2\right)$, we use the policy iteration technique based on the primal-dual optimization approach to find $\left(\mu^*,\lambda_1^*,\lambda_2^*\right)$. Policy iteration consists in obtaining an improved policy by means of a minimization process until no further improvement is possible. It is composed of two main steps executed simultaneously:

1- **Policy evaluation**: Given the policy $\mu_m^{\lambda_1,\lambda_2}$, compute the corresponding value function (cost to go) from all the states s on the infinite horizon and starting from an initial state s_0. It can be achieved iteratively using the RVIA algorithm to find the average cost to go $V_m^{\lambda_1,\lambda_2}\left(s\right)$ for $s\neq s_0$ and the average cost per stage $\beta_{\mu_m}^{\lambda_1,\lambda_2}$ by solving the linear program:

$$\begin{aligned}&-\ V_m^{\lambda_1,\lambda_2}\left(s\right)=L\left(s,\mu_m^{\lambda_1,\lambda_2}\left(s\right),\lambda_1,\lambda_2\right)+\sum_{s'}p\left(s'\middle|s,\mu_m^{\lambda_1,\lambda_2}\left(s\right)\right)V_m^{\lambda_1,\lambda_2}\left(s'\right)-\beta_{\mu_m}^{\lambda_1,\lambda_2}\quad\text{for}\quad s\neq s_0\\&-\ V_m^{\lambda_1,\lambda_2}\left(s_0\right)=0\end{aligned} \tag{92}$$

2- **Policy improvement**: Obtain a new policy $\mu_{m+1}^{\lambda_1,\lambda_2}$ allowing a better average cost:

$$\mu_{m+1}^{\lambda_1,\lambda_2}\left(s\right)=\operatorname*{argmin}_{u\in\Omega}\left[L\left(s,u,\lambda_1,\lambda_2\right)+\sum_{s'}p\left(s'\middle|s,u\right)V_m^{\lambda_1,\lambda_2}\left(s'\right)\right] \tag{93}$$

The iterations continue until $\mu_m^{\lambda_1,\lambda_2}=\mu_{m+1}^{\lambda_1,\lambda_2}=\mu^*_{\lambda_1,\lambda_2}$ thus: $\mu^*_{\lambda_1,\lambda_2}=\operatorname*{argmin}_{\mu}L_{av}\left(\mu,\lambda_1,\lambda_2\right)$.

As the constrained average cost problem is convex. The optimal $\left(\mu^*_{\lambda_1^*,\lambda_2^*},\lambda_1^*,\lambda_2^*\right)$ is a saddle point for $L_{av}\left(\mu,\lambda_1,\lambda_2\right)$ and verifies that $\left(\mu^*_{\lambda_1^*,\lambda_2^*},\lambda_1^*,\lambda_2^*\right)=\arg\max_{\lambda_1,\lambda_2}\left(\operatorname*{argmin}_{\mu}L_{av}\left(\mu,\lambda_1,\lambda_2\right)\right)$.

Finding the optimal Lagrange multipliers is the dual optimization, that is the iteration on the dual variables, the Lagrange multipliers. It can be obtained using an iterative gradient descent in an outer loop to update these parameters as

$$\begin{aligned}\lambda_1^{n+1}&=\lambda_1^n+\xi_1\left(\overline{Q}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}-\delta_Q\right)\\\lambda_2^{n+1}&=\lambda_2^n+\xi_2\left(\overline{O}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}-\delta_O\right)\end{aligned} \tag{94}$$

ξ_1 and ξ_2 are decreasing increment coefficients. The average delay cost $\overline{Q}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}$ and the average overflow cost $\overline{O}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}$ under policy $\mu^*_{\lambda_1^n,\lambda_2^n}$ can be computed following equation (*) with respect to a state distribution $\rho^{\mu^*_{\lambda_1^n,\lambda_2^n}}$ defined as function of the state transition probability under policy $\mu^*_{\lambda_1^n,\lambda_2^n}$. The dynamic offline algorithm can be summarized as following:

Initialization: $n=1,\quad \underline{\lambda_{1,0}}=\underline{0}\quad \underline{\lambda_{2,0}}=\underline{0}$

Choose an initial distribution ρ_0, choose an initial policy μ_0 and an initial state $s_0=(0,Z_1)$

for $n=0$ until $\left\|\lambda_1^{n+1}-\lambda_1^n\right\|<\theta_1$ and $\left\|\lambda_2^{n+1},\lambda_2^n\right\|<\theta_2$

for $m=1$ until $\mu_{m+1}^{\lambda_1^n,\lambda_2^n}=\mu_m^{\lambda_1^n,\lambda_2^n}$

* policy evaluation step: find $\beta_{\mu_m}^{\lambda_1^n,\lambda_2^n}$ and $V_m^{\lambda_1^n,\lambda_2^n}(s);s\neq s_0$

solve

$$\begin{cases} V_m^{\lambda_1^n,\lambda_2^n}(s)=L\left(s,\mu_m^{\lambda_1^n,\lambda_2^n}(s),\lambda_1^n,\lambda_2^n\right)+\sum_{s'}p\left(s'\middle|s,\mu_m^{\lambda_1^n,\lambda_2^n}(s)\right)V_m^{\lambda_1^n,\lambda_2^n}(s')-\beta_{\mu_m}^{\lambda_1^n,\lambda_2^n} & \text{for}\quad s\neq s_0 \\ V_m^{\lambda_1^n,\lambda_2^n}(s_0)=0 \end{cases}$$

*Policy improvement step to find $\mu_{m+1}^{\lambda_1^n,\lambda_2^n}$

$$\mu_{m+1}^{\lambda_1^n,\lambda_2^n}(s)=\arg\min_{u\in\Omega}\left[L\left(s,u,\lambda_1^n,\lambda_2^n\right)+\sum_{s'}p\left(s'|s,u\right)V_m^{\lambda_1^n,\lambda_2^n}(s')\right]\qquad\forall s\in S$$

$m=m+1$

End $\mu^*_{\lambda_1^n,\lambda_2^n}=\mu_m^{\lambda_1^n,\lambda_2^n}$

* occupation measure computation

Solve $\rho^{\mu^*_{\lambda_1^n,\lambda_2^n}}(s)=\sum_{s'}p\left(s\middle|s',\mu^*_{\lambda_1^n,\lambda_2^n}(s')\right)\rho^{\mu^*_{\lambda_1^n,\lambda_2^n}}(s')\quad\forall\ s\in S$

* Evaluation of the scheduling policy delay and overflow costs

$$\overline{Q}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}=\sum_s\rho^{\mu^*_{\lambda_1^n,\lambda_2^n}}(s)\ L_Q\left(s,\mu^*_{\lambda_1^n,\lambda_2^n}(s)\right)$$

$$\overline{O}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}=\sum_s\rho^{\mu^*_{\lambda_1^n,\lambda_2^n}}(s)\ L_O\left(s,\mu^*_{\lambda_1^n,\lambda_2^n}(s)\right)\qquad(*)$$

* Update the Lagrange multipliers

$$\lambda_1^{n+1}=\lambda_1^n+\xi_1\left(\overline{Q}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}-\delta_Q\right)$$

$$\lambda_2^{n+1}=\lambda_2^n+\xi_2\left(\overline{O}_n^{\mu^*_{\lambda_1^n,\lambda_2^n}}-\delta_O\right)$$

end

5.2.1.2.3.2 *Randomized offline strategy*

The randomized offline policy consists in a probabilistic mapping from the state space S to the action space Ω, associating for each state s an action $u=\mu(s)$ that is a random variable with a probability distribution uniquely associated to the visited state. Therefore, the offline randomized strategy μ relies on weighting each (state s, action u) pair with a given probability called the occupation measure. This measure represents the probability to visit each (s,u) pair according to the devised strategy [Altman99].

Let p_0 be the probability distribution of the initial state s_0. For an infinite length trajectory, the occupation measure $\rho^{\mu}(s,u)$ of a state-action pair (s,u) when the policy μ is employed corresponds to the fraction of time slots where the state s is visited and the action $u=\mu(s)$ is performed under the condition that the initial state s_0 of the trajectory is drawn according to p_0. It is defined by

$$\rho^{\mu}(s,u)=\lim_{N\to\infty}\frac{1}{N}E^{\mu}\left[\sum_{n=1}^{N}\mathbf{1}_{(s_n=s,\mu(s_n)=u_n)}\right]$$

The optimal occupation measure can be obtained by formulating the CMDP optimization problem as a linear programming problem [Altman99] as

$$\begin{aligned}&\rho^{*}=\underset{\mu}{\operatorname{argmin}}\sum_{s\in\mathrm{S},u\in\Omega}\rho(s,u)L_E(s,u)\\&\text{s.t.}\sum_{s\in\mathrm{S},u\in\Omega}\rho(s,u)q\le\delta_Q\ ,\ \sum_{s\in\mathrm{S},u\in\Omega|q>B}\rho(s,u)\le\delta_O\\&\sum_{s'\in\mathrm{S},u\in\Omega}\rho(s',u)\left(\mathbf{1}_{(s'=s)}-p(s\mid s',u)\right)=0\ \ \forall s\in\mathrm{S}\end{aligned}\qquad(95)$$

where the last line results from the Markov property of the process (S_n,U_n). An optimal randomized policy can be then obtained from ρ^{*} the solution of the above problem by choosing a random action u at a given state s according to the following probability

$$p_s(u)=\frac{\rho^{*}(s,u)}{\sum_{u'\in\Omega}\rho^{*}(s,u')}.\qquad(96)$$

5.2.1.2.4 Online Post-Decision State based Learning Approach

The offline strategies require the knowledge of the channel transition probabilities and the arrival distribution to resolve the Bellman equation of the constrained MDP. However, these parameters are unknown in an online context. Therefore, the online approach can be used by learning from past to find the policy that can cope with the unknown environment [Salodkar08]. The online strategy relies on a Lagrangian relaxation on top of a learning strategy. It offers the advantage of requiring low level of information on the application properties and the channel statistics. In the sequel, the post-decision state framework proposed in section 5.2.1.2.1.3 can be used to approximate the cost functions in the policy iteration method adopted for the deterministic offline strategy in section 5.2.1.2.3.1.

The post-decision state at slot t_n, $\tilde{S}_n=(\tilde{Q}_n,Z_n)=(Q_n-U_n,Z_n)$ is defined as previously as a virtual state that occurs after taking the decision $U_n\in\Omega$ at t_n (processing U_n during slot duration ΔT) before new packets A_{n+1} and channel state transitions Z_{n+1} happen at t_{n+1}. To solve the value iteration

problem, a value function on the space of post-decision states $\tilde{\mathrm{S}}=\left\{\tilde{s}(s,u)=(q,z), s\in \mathrm{S}, u\in\Omega\right\}$ can be defined as $\tilde{V}_{\lambda_1,\lambda_2}(\tilde{s})=\sum_{s'\in \mathrm{S}} p(s'|\tilde{s})V_{\lambda_1,\lambda_2}(s')$ and then the post-decision state value function at t_n is given by

$$\tilde{V}_{\lambda_1,\lambda_2}(\tilde{s})=\sum_{s'} p(s'|\tilde{s})\min_{u'\in\Omega}\left[L(s',u',\lambda_1,\lambda_2)+\tilde{V}_{\lambda_1,\lambda_2}(\tilde{s}')\right]-\tilde{V}_{\lambda_1,\lambda_2}(\tilde{s}_0) \tag{97}$$

with the instantaneous cost per stage is defined in section 5.2.1.2.3.1 at time t_n as

$$L(S_n,U_n,\lambda_1,\lambda_2)=L_E(Z_n,U_n)+\lambda_1(Q_n-\delta_Q)+\lambda_2(1_{Q_n>B}-\delta_O) \tag{98}$$

It can be noticed that the *expectation* operation has been moved outside the *min* operator. This expectation can be performed by averaging in time in order to determine the optimal value function. The online scheme needs to learn only about the current state being observed, and can update only the corresponding component in the value function. The post-decision state Markov chain preserves the same transition probabilities as the original Markov chain (pre-decision state).

For any post-decision state $\tilde{S}_n$ at time slot t_n, the online policy iteration algorithm can be expressed as

$$\begin{aligned}
&\tilde{V}_{n+1}(\tilde{S}_n)=(1-\alpha_n)\tilde{V}_n(\tilde{S}_n)+\alpha_n\left[\min_{U_{n+1}}\left[L\left((Q_n+A_{n+1},Z_{n+1}),U_{n+1},\lambda_1^n,\lambda_2^n\right)+\tilde{V}_n\left(\tilde{Q}_n+A_{n+1}-\left|U_{n+1}\right|,Z_{n+1}\right)\right]-\tilde{V}_n(\tilde{S}_0)\right]\\
&\tilde{V}_{n+1}(\tilde{S}_n')=\tilde{V}_n(\tilde{S}_n)\quad \forall \tilde{S}_n'\neq\tilde{S}_n=(\tilde{Q}_n,Z_n)\\
&\lambda_1^{n+1}=\lambda_1^n+\zeta_1^n(\tilde{Q}_n+A_{n+1}-\delta_Q)\\
&\lambda_2^{n+1}=\lambda_2^n+\zeta_2^n(\bar{O}-\delta_O)
\end{aligned} \tag{99}$$

where α_n, ζ_1^n and ζ_2^n are positive update sequences to guarantee convergence of the algorithm.

If we focus on the policy evaluation step, we note that the update of the post-decision state is performed by inserting the cost incurred by the next pre-decision state. It is updated after an observation of the consumed energy to transmit packets in current system state and the arrivals that can occur in the post-decision state.

In the policy improvement step, we consider the packets arrival A_{n+1} as a weighted sum of the real arrival during slot n observed at t_{n+1} and the expected arrival during slot $n+1$ for each decision that the system can take. α characterizes the accuracy of the current pre-decision queue state information. If $\alpha=1$, then the system can have an exact state information but has no control on the offloading power while if $\alpha=0$, we have no information on the current pre-decision state but an optimal control policy on the offloading power. In our case, we choose $\alpha=0.5$. It is a partial state information Markov decision process combining with the same ratio a real arrival with an estimated one based on the N_s previous observations. This consideration even though it modifies the arrival distribution, it respects well the average arrival rate.

Since the policy evaluation is based on the original arrival distribution, the characterization of the cost-to-go related to the policy obtained for each state satisfies our optimization problem. The packet scheduling policy that is found is deterministic for each pre-decision state but it is sub-optimal. In fact, it does not respond instantaneously to the stochastic shortest principle but will guarantee for the system the average delay constraints with the control on the power consumed for offloading. This model can be seen as an imperfect state information problem which the system has to estimate before taking a decision considered as optimal.

On the other hand, the transmission is scheduled on an integer number of packets, while the instantaneous arrival can be a real number. Thus, at the policy evaluation equation, we impose a projection operator to consider the nearest integer queue state of the current queue length. This will reduce the number of states allowing the system to converge in a reasonable number of iterations. Then, the approximated queue states are considered as integers to fill the value matrix.

The PDS online learning algorithm can be presented as following:

1. Initialization: $n=0$, $\tilde{V}_0(\tilde{s})=0$ for all $\tilde{s}\in \mathrm{S}$, $\lambda_0=0, Q_0=0, \tilde{Q}_0=0, t_0=0$ and ΔT

While $t_n < N$

2. Observe the arrival packets at the buffer A_{n+1} and the channel state Z_{n+1} at t_{n+1}.
3. Choose the optimal decision U_{n+1}(offloading, local processing, no processing) from policy improvement step of the PI algorithm.
4. Determine the number of packets to be scheduled and processed U_{n+1} and the energy consumed E_{n+1} to process these packets.
5. Update the value function of the current post-decision state $\tilde{S}_n=\left(\tilde{Q}_n, Z_n\right)$ and keep the other elements of the value matrix unchanged according to the policy evaluation step of the PI algorithm.
6. Update the Lagrange multipliers $\lambda_1^{n+1}, \lambda_2^{n+1}$
7. Set $n \leftarrow n+1$ $\quad t_{n+1}=t_n+\Delta T$ $\quad \tilde{Q}_{n+1} \leftarrow \tilde{Q}_n - U_{n+1} + A_{n+1}$ $\quad Z_{n+1} \leftarrow Z_n$

End

5.2.1.2.4.1 *Simulation results*

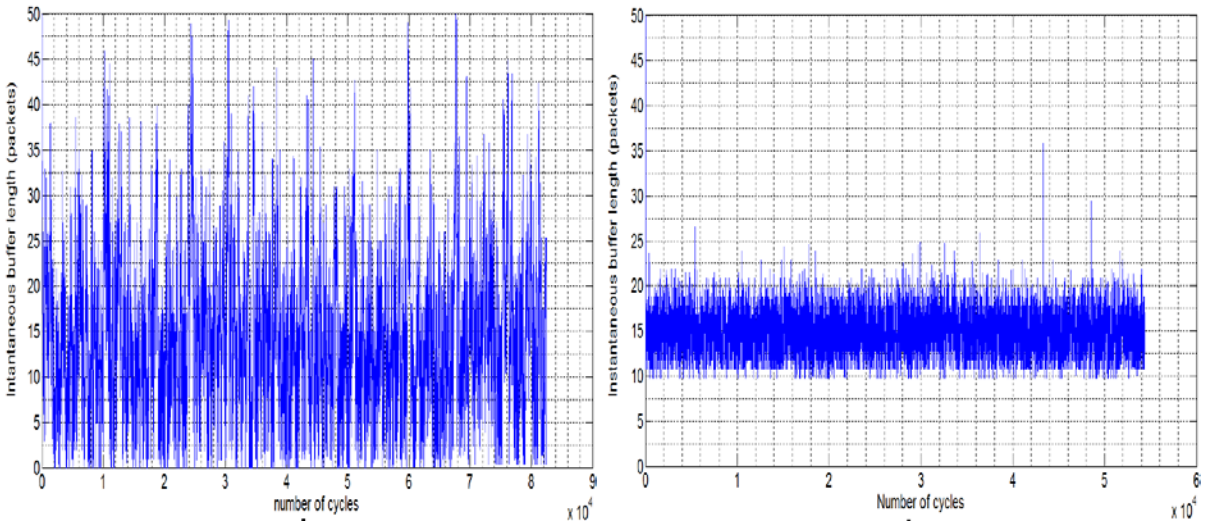
In this section, we compare both offline and online strategies devised for the scheduling-computation offloading problem. We consider a single small cell single user scenario in an LTE based network. The allocated bandwidths for UL and DL transmissions are equal to $W_{\mathrm{UL}}=500\,\mathrm{KHz}$ and $W_{\mathrm{DL}}=5\,\mathrm{MHz}$ and the time slot duration is $\Delta T=2\,\mathrm{ms}$. The data arrivals is modeled as a Poisson distribution with an average rate equal to 2.5 bits/sec. All packets that are processed have identical size $L_{\mathrm{UL}}=l=500\,\mathrm{bits}$ and the result of processing of each packet has size $L_{\mathrm{DL}}=5000\,\mathrm{bits}$. The maximum buffer size is $B=50$ packets. When more than 1 packets are offloaded to the SCeNBce, they are processed serially.

For the communication channel model, the channel state follows an exponential distribution with mean $\tau=0.47$ and the quantized channel state z can take 8 possible values from the finite space Expressed in dB Z=[−13,−8.5,−5.4,−3.3,−1.6,−0.08, 1.57, 3.18]. All powers are normalized by the signal to noise ratio experienced by the SCeNB when unitary power signal is sent at the UE side. This is equivalent to normalizing by WN_0/γ where γ is the path-loss of the channel between the UE and the SCeNB without fading. The numerical values of these powers at the user are given by: $P_{UE}=0.7$; $P_{\max}=6$; $P_r=0.5$; $P_{idle}=0$; $P_w=0.2$. At the SCeNBce, the transmission power is $P_{SCeNB}=20$ and the time spent to process one packet is $T_w=0.1\,\mathrm{ms}$. The maximum number of packets that can be offloaded during one time slot is set to 5 packets and the maximum number of packets that can be processed locally during one time slot is set to 1 packet.

In the sequel, we study the performance of the optimal policy computed by both the dynamic deterministic offline and the online algorithms under different conditions. For the offline model, we obtain the optimal policy according to the dynamic offline algorithm. Then, by applying this policy, we compute the average consumed energy and the average buffer length over a random trajectory of states following the same Markov chain transitions. On the other hand, for the online model, we obtain

an optimal value function and an optimal policy according to the online PDS algorithm. Then, we consider the same random trajectory to compare it to the offline algorithm in terms of average consumed energy and average buffer length.

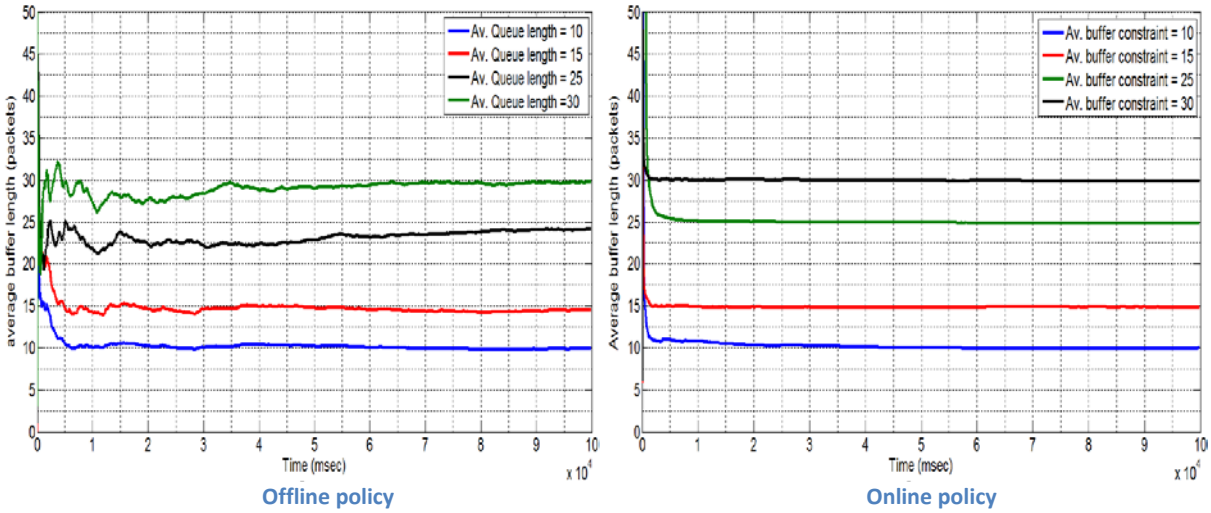
First, we impose different average delays that can be modeled as average buffer length. Then, we observe the system behavior under different queuing constraints $[10;15;25;30]$ packets.



Offline policy Online policy

Figure 119. The instantaneous buffer length variation for offline and online algorithms.

In Figure 119, we analyze the fluctuations of the instantaneous buffer for the offline and the online algorithms, with an arrival rate of 2.5 bits/sec and a buffer length constraint of 15 packets. We notice that the offline method exploits perfectly the buffer window since it has perfect knowledge on the exact arrival and channel transitions distribution. Thus, it sends the maximum number of packets when the channel conditions are good and limits its transmission when the channel is bad. However, the online policy has a restricted view of the Markov chain transitions, hence a limited action window. Therefore, it does not allow that the buffer exceeds a maximal queue length by imposing the system to send packets to return to a set of post-decision states considered as optimal even if the channel conditions are not beneficial for the energy consumption budget.

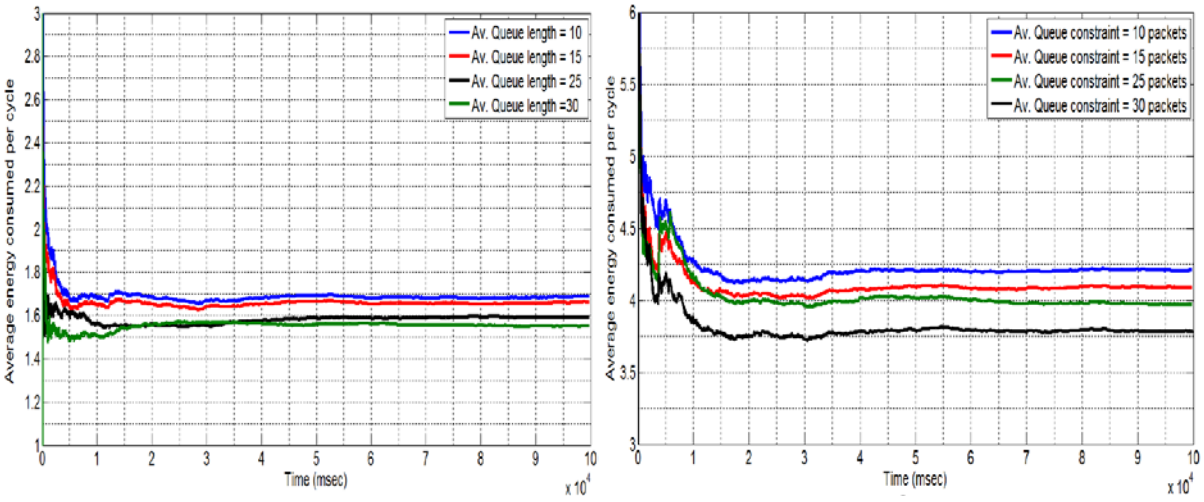


Offline policy Online policy

Figure 120. Convergence of the average buffer length for different buffering constraints.

Figure 120 confirms that both algorithms respect the average imposed delay. This means that the adopted policy allows the system to adapt to the dynamic environment by adopting an appropriate transmission rate and hence power consumption for all the state space of the system.

On one side, the optimal offline policy succeeds to establish an optimal opportunistic strategy that exploits the channel conditions and the buffer states window by managing well its instantaneous data rates and its offloading power profile. On the other side, the online obtained policy shows that it is able, despite the partial state information and partial Markov chain transitions knowledge, to find a set of recurrent post-decision buffer states for all the channel states which occur, allowing the system to be close to the average buffer length constraint with an acceptable energy consumption budget.



Offline policy Online policy

Figure 121. Convergence of the average energy consumed per time slot for different buffering constraints.

Figure 121 represents the average energy consumed per time slot for the offline and online methods. It shows that the UE terminal reaches a fixed energy level confirming its capability to adapt to the average data rate and the channel conditions. Moreover, it can be seen for both methods that more the queuing delay is rigorous; more the average energy consumed is high.

In fact, for the online method, the Lagrange multiplier varies inversely with the queue constraints. Thus for low delay constraints, the scheduler is less tolerant with buffer's excesses penalizing more the transient buffer states. It pushes the transmitter to spend higher power to establish lower processing delay and so return to the optimal post-decision recurrent states in a lower number of transitions.
For the offline method, having a low delay constraint means that the average buffer constraint has not a big margin from the buffer state 0. So it can be exceeded from 0 instantaneously in a low number of steps. This pushes the system to send also in states with bad channel conditions consuming thus more energy to remain in average close to the buffer constraint.

In the following results, the average queue constraint is $\delta_Q = 10$ packets and the tolerated overflow probability is $\delta_O = 10^{-5}$. We vary the arrival rate $[1.25; 2.5; 3.75; 5]$ Mbits/sec to study the strategy employed by the scheduler for defining the rate and the power for the state space.

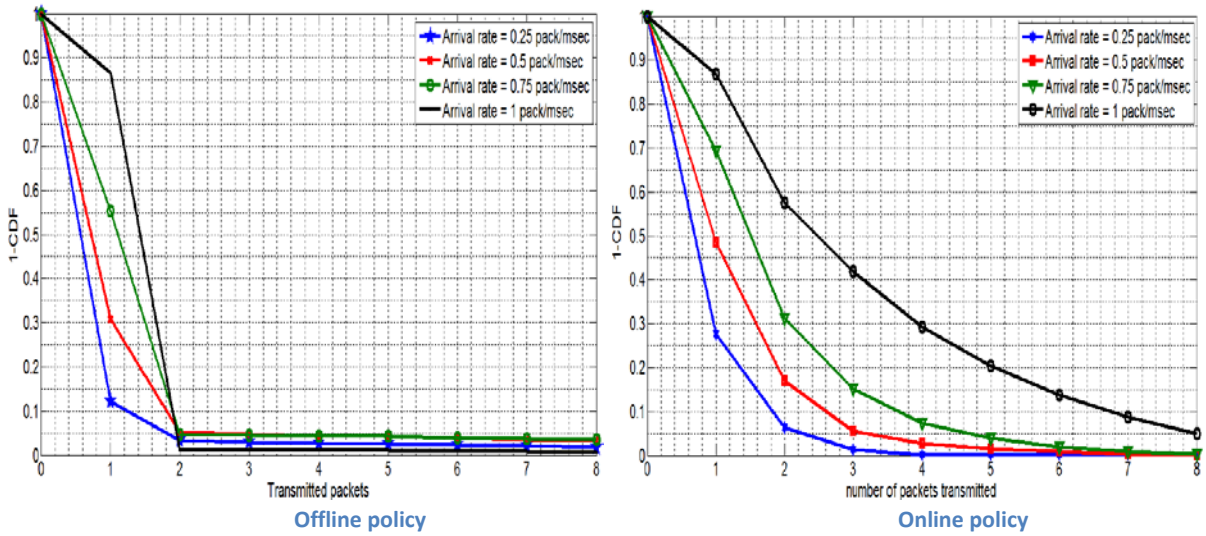


Figure 122. (1-CDF) of the packets transmission for different arrival rates.

From Figure 122, we can see that for both methods, the system adapts his data transmission strategy with the arrival rate. In fact, we have noticed that when the arrival rate is important, the system limits the number of states in which it does not process packets to be able to liquidate the next arrivals in an acceptable delay without consuming much energy.

For the offline method, we notice that the system sends most of the time only one packet at the beginning of each slot duration and can send more packets when the channel states are better (1.42 dB and 3.14 dB). This ensures to minimize the energy consumption. Regarding the number of states at which the scheduler sends, we observe that for low arrival rate, it processes only at states with good channel conditions. On the other hand, when the average arrival flow is high, it is forced to process even if the channel state is bad.

For the online method, we note that for higher arrival rate, instantaneous large excess over the average buffer constraint can occur in a low number of transitions with high probability. So the scheduler is forced to send more packets at each time slot to bring back faster his instantaneous post-decision buffer state to the set of states considered as optimal, in terms of respecting the delay constraint with acceptable energy consumption.

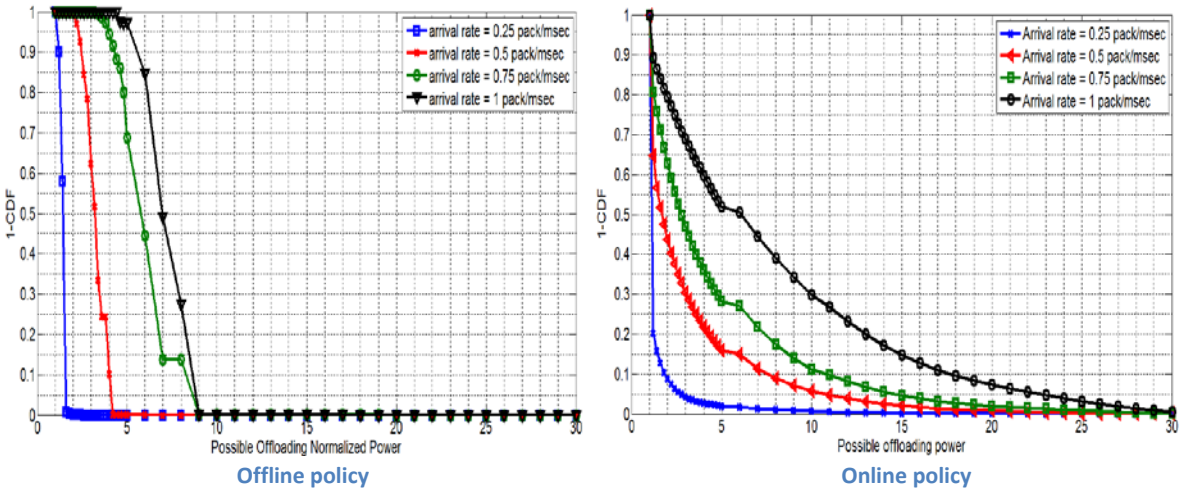


Figure 123. (1-CDF) of the power used for offloading at different arrival rates.

Figure 123 represents the distribution of the power consumed by the UE terminal in case of offloading decision following both offline and online policies. We note that the UE adapts its transmission

strategy, thus the power, to respect the queue constraint relatively to the arrival rate. We can see that for both cases, with higher arrival rate, the scheduler dispenses higher power for offloading to the SCeNBce.

For the offline method, we can see that the normalized power has never exceeded 8W, even if the arrival is about 1 packet/msec. This shows that the system adapts its strategy to the channel conditions. For instance, it is decided to locally process at the UE terminal if the channel is very bad. As the delay varies in $(1/\log P)$, spending higher power does not improve in this case the process duration.

For the online method, the system always tends towards returning to the set of the optimal post-decision states in a small number of transitions. Thus, if the arrival rate is high, it experiences a recurrent excess of the average constraint and is pushed sometimes to offload with a higher power to empty the buffer faster. Then, in the next time slots, it will be able to recover from the divergence of the instantaneous buffer caused by the partial state information without spending much energy in average.

In Figure 124, we illustrate the average consumed energy as function of the data arrival rates for different offline strategies. We compare the deterministic dynamic programming policy denoted "Dynamic programming policy" and the randomized dynamic programming policy denoted "Linear programming policy" to three other policies that maintain the same average queue length. The "Only offloading dynamic policy" is based on the deterministic dynamic programming policy but takes only offloading decisions to process the packets remotely at the SCeNBce. The "Only local processing" and "Immediate scheduling" policies are however static policies. The former one consists in processing locally at the UE all the packets that exceed the average queue constraint δ_Q. The latter one decides to process u packets when the queue length exceeds δ_Q according to: $u = \max\left(\min\left(q + a - \delta_Q, 5\right), 0\right)$. If $u = 1$, it can choose either to process locally or to offload depending on the action that minimizes the instantaneous consumed energy. On the other side, if $u > 1$, it performs only offloading.

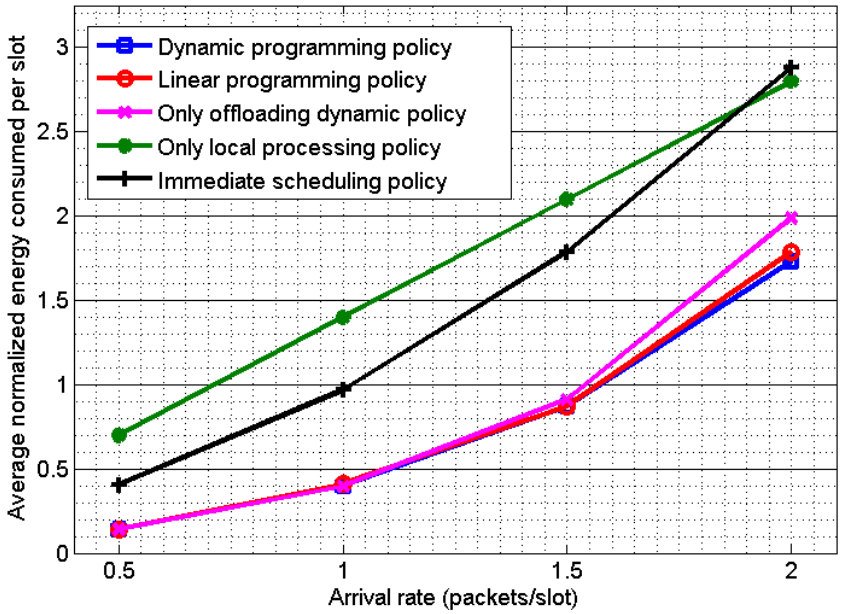


Figure 124. Average Energy Consumption for offline policies.

We can observe that the deterministic dynamic programming strategy achieves the optimal performance of the randomized linear programming strategy and gives the lowest energy consumption. Indeed, for these strategies, the system has perfect knowledge of the state transitions. It can adapt its processing decisions according to the arrival rate and the channel conditions while

satisfying the imposed delay constraints. Thus, more packets can be offloaded in good channel states whereas transmission is limited and local processing or idle mode are decided in case of bad channel states. The only offloading dynamic policy has close performance to the latter strategies for low and moderate arrival rates but consumes more energy for high arrival rates. This is due to the fact that the packets are offloaded for remote processing even when the channel is in bad conditions since local processing decisions are not possible. We can notice also that both static policies consume much more energy than dynamic ones since they do not exploit the radio link quality. Idle mode is not allowed in case of high arrival rates and bad channel states.

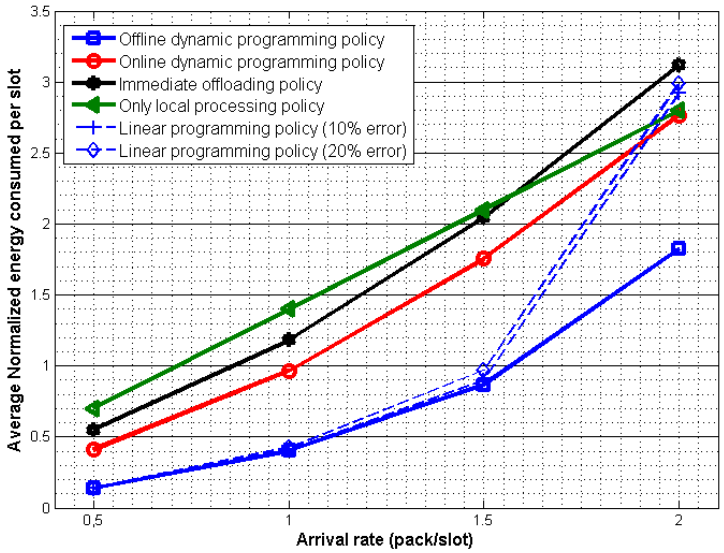


Figure 125. Average Energy Consumption for offline and online policies.

Figure 125 compares the offline policy "Offline policy perfect knowledge" with the "Online dynamic programming policy" (based on PDS learning) and the static strategies "Immediate offloading policy", where the terminal offloads all packets exceeding the average buffer size constraint δ_Q and the "Internal only processing policy" where it processes all packets exceeding the average buffer size constraint. The curves named "Linear programming policy (10% margin)" and "Linear programming policy (20% margin)" draw the average energy when the UE employs the pre-calculated randomized offline strategy that has been devised with an overestimated arrival rate (10% and 20% of excess). One can see that the offline strategy outperforms the online approach by more than 50% for low and moderate arrival rates and offers similar performance for high arrival rates. The online strategy offers between 20 and 30% of gain compared to static policies.

One can also see that the performance of pre-calculated offline strategies with 10% and 20% margins are close to those achieved by offline policy with perfect knowledge for low and moderate arrival rates. With the considered channel distribution the maximal rate of offloading is 2.5packets/slot. When the arrival rate approaches this value both offline and online strategies have the same performance as the immediate offloading policy. In fact, with the considered channel distribution, the instantaneous energy cost per packet is better for offloading (compared to internal processing) in more than 70% of cases. With high arrival rates the optimal strategy can no longer take benefits from the channel diversity by choosing the idle decision when the channel is in bad states.

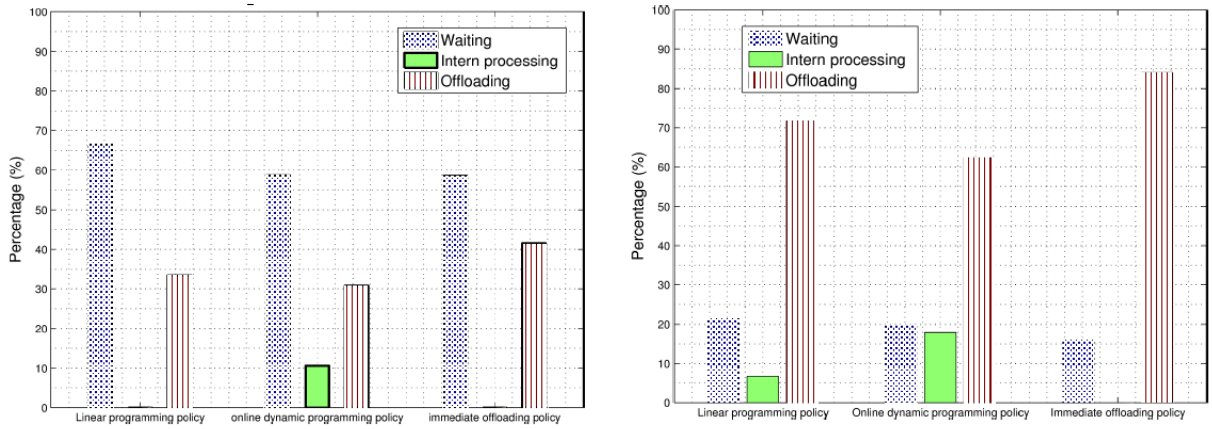


Figure 126. Percentage of processing decisions for different arrival rates : 1 packet/slot (left) and 2 packets/slot (right).

Figure 126 draws the processing decisions distribution for the offline, online and immediate offloading strategies for arrival rates of 1 packet/slot and 2 packets/slot, respectively. For high arrival rates (2 packets/slot), we can notice that the offline strategy favors offloading decisions for most of the time. Indeed, the arrival rate approaches the maximal rate of offloading (2.5 packets/slot) in this case, and hence, under delay and overflow constraints, the offline strategy makes decisions to process the queued packets either by offloading when channel states are good or by processing locally or waiting until next decision time under bad channel conditions. On the other hand, for low arrival rate (1 packet/slot), Idle mode is decided whenever the channel is in bad states and offloading decisions are made when the channel conditions allow for better energy saving, limiting thus the energy-consuming local processing. The offline strategy knows perfectly the channel distribution and takes better benefits from good channel conditions contrary to the online strategy which relies on incomplete knowledge of the channel distribution. The online strategy makes use of local processing for more than 10% of cases with 1 packet/slot and for about 20% of cases with 2 packets/slot.

5.2.1.2.4.2 *Conclusion*

In this section, we have considered a single user single SCeNBce scenario where the computing resources installed in the SCeNB are leveraged to offload mobile applications that process user generated data. We have investigated joint radio resource allocation and offloading strategies to minimize the average energy consumed by the user equipment while satisfying predefined delay constraints imposed by the applications. For this, we have studied deterministic and randomized offline and online dynamic programming strategies. The offline algorithms require perfect knowledge of the Markov chain transitions. They exploits the channel transitions to compute the optimal policy that reduces the energy consumption under fixed average delay. The proposed online policy iteration algorithm, on the other hand, does not require any information on the channel transitions and the arrival distribution. It finds a policy based on partial state information and uses post-decision state learning technique to evaluate value iteration equation updates. Both offline and online solutions have shown better energy saving with respect to static strategies. Although the online learning approach is suboptimal in terms of energy consumption compared to the offline approach, it is able to cope with transmission in an unknown environment. The precalculated offline strategies offer significant gains compared to the online strategies even if they are designed with a certain margin in order to account for imperfect knowledge of the application properties. Offline strategies offer also the advantage to require lower signalling overhead. The results of sections (5.2.1.2.2 – 5.2.1.2.4) have been proposed in [Labidi15] and [Kamoun15].

5.2.1.2.5 Extension of Dynamic programming strategies for multi-user case

We extend in this section the joint resource allocation and offloading problem to a multi-user scenario. We consider K users served by a single SCeNBce. The resources scheduling is based on time division

multiple access technique. Thus, at each time slot, only one user has access to the available bandwidth. The mobile users have different applications requirements with random data arrivals and are subject to random channel states when communicating with the SCeNBce. On the cloud side, the SCM is aware of these channel conditions and of the users' buffer status sent through buffer status reports to the SCeNBce and communicated to the SCM. Based on these channel and buffer state information and aware of the applications requirements of each user, the SCM decides at each time slot which user has to be scheduled with the optimal transmission rate while minimizing its energy consumption. It decides also if the other users have to process locally their packets or wait until next decision time. When a given user is scheduled, the application data packets are offloaded to the SCeNBce on a SISO link to process it and return back the result to the mobile user.

The main objective is to minimize the average energy consumption for all the users served by the SCeNBce under average delay constraints in order to satisfy their respective QoS and application requirements. As in the single user case, the optimization problem consists in finding the optimal scheduling-offloading policy by deciding at regularly spaced time slots between three possible actions for the users' application data processing: (a) local processing at the UE, (b) computation offloading for remote processing at the SCeNBce or (c) wait until the next decision time. Therefore, the problem is formulated as a constrained Markov decision process problem where the objective function is given by the infinite time averaged energy consumed by the UEs in order to run their application. It is resolved by applying the proposed offline and online dynamic programming strategies in the multi-user.

5.2.1.2.5.1 *System Model and Problem formulation*

As in previous section, we assume that the scheduling decision are taken at regularly spaced epochs with ΔT the time slot duration between two successive decisions. At time t_n, each user indexed by i has a buffer length Q_n^i expressed in packets. The SCM scheduler decides for each user the number of packets U_n^i $\left(U_n^i \le Q_n^i\right)$ to be processed either locally or offloaded to the SCeNBce. At users' side, we assume that the application data flow follows a Poisson distribution with a fixed arrival rate $\overline{a^i}$ and the arrivals are stored in the user's buffer until processed. The instantaneous arrivals A_{n+1}^i are observed at t_{n+1}. Thus, the users' buffer states will be equal to $Q_{n+1}^i = \max\{Q_n^i - U_n^i, 0\} + A_{n+1}^i$. We consider that the users communicate with the SCeNBce through N Rayleigh channels, with complex gains H_n^i that remain fixed during the duration ΔT. We denote Z_n^i the quantized channel states for the user i at the n^{th} time slot and W_n^i the additive complex white Gaussian noise with zero mean and variance $N_0{}^2$. If UE i is scheduled to offload data packets to the SCeNBce, by transmitting the signal y_n^i, then the received signal is given by: $R_n = H_n^i y_n^i + W_n^i$.

According to the possible decisions on processing locally or remotely a given number of data packets or waiting (no processing), at each slot t_n and for each user i, the processing delay costs are D_n^i and the instantaneous energy costs are $E_n^i = L_E\left(S_n^i, U_n^i\right)$ as defined in the single user case.

The SCM has perfect knowledge of the channel qualities, the buffer status of each user and the energy cost that a user has to spend to perform the processing decisions. Thus, the SCM aim at finding at the beginning of time slots an optimal policy μ that defines the user to be scheduled, the processing type, the rate for the local processing, and the rate allocation for the offloading. This optimal policy allows all the users to respect the QoS required (arrival rate, buffering delay) with an acceptable energy consumption budget. The optimization problem is a Markov decision process where the system can be characterized as a Markov chain with finite states and actions spaces defined as following:

The state space: The state space varies in time and takes into account the buffers and the channel conditions of the K users. The state of the system at the slot n is

$S_n = \left[S_n^1, S_n^2, ..., S_n^K \right] = \left[(Q_n^1, Z_n^1), (Q_n^2, Z_n^2), ..., (Q_n^K, Z_n^K) \right]$ where the channel state Z_n^i and the buffer state Q_n^i for user *i* take values from finite spaces Z and Q respectively . We suppose that the maximal buffer values are equal for all users. Thus, for K users the state space cardinality can be expressed as: $|S| = |\mathrm{Q}|^K \times |Z|^K$.

The state space:
$\Omega = \left[\Omega^1, \Omega^2, ..., \Omega^K \right] = \left[\left\{ -1, 0, \ldots, U_{\max}^1 \right\}, \left\{ -1, 0, \ldots, U_{\max}^2 \right\}, ..., \left\{ -1, 0, \ldots, U_{\max}^K \right\} \right]$ is also finite where the control action determines the number of packets U_n^i to be treated at time t_n by all users, $\left| U_n^i \right| \le min\ (Q_n^i, U_{\max}^i)$. -1 denotes the local processing of one packet.

The transition probabilities: $p\left(s' \middle| s, u \right)$ specifies the transition from a state s to another s' following a specific action Ω . It depends on the perturbations which could affect the system (channel transitions and random arrival for all the users). We suppose that the arrival and the channel state are independent for all the users and that the channel states are not correlated in time thus the transition probabilities could be expressed as: $p\left(s' \middle| s, u \right) = \prod_{i=1}^{K} p\left(a = q_i' - (q_i - u_i) \right)\ p(z_i)$

We define μ as the policy of this problem, μ is a mapping between the state space and the action space $\Omega = \mu(s)$. Our main objective is to design a scheduler that minimizes the average energy consumed by all the users subject to constraint on the average delay and the average packets' overflow for each one of them. Then, the problem is a CMDP with average cost and finite state and action spaces. The time averaged energy consumed, queue length and packets overflow of a user *i* can be defined respectively on an infinite horizon as:

$$\begin{aligned} \overline{E}^i &= \limsup_{N\to\infty}\ \frac{1}{N}\ \mathrm{E}\left[\sum_{n=1}^{N} L_E^i(S_n, U_n) \right];\ \ \overline{Q}^i = \limsup_{N\to\infty}\ \frac{1}{N}\ \mathrm{E}\left[\sum_{n=1}^{N} L_Q^i(S_n, U_n) \right]; \\ \overline{O}^i &= \limsup_{N\to\infty}\ \frac{1}{N}\ \mathrm{E}\left[\sum_{n=1}^{N} L_O^i(S_n, U_n) \right] \end{aligned} \tag{100}$$

As the Markov chain is ergodic the averaging in time can be converted to an expectation over the occupation of the states following the policy μ

$$\begin{aligned} \overline{E}^{\mu,i} &= E^{\mu}\left[L_E^i\left(s, \mu(s)\right) \right] = \sum_s \rho^{\mu}(s)\ L_E^i\left(s, \mu(s)\right) \\ \overline{Q}^{\mu,i} &= E^{\mu}\left[L_Q^i\left(s, \mu(s)\right) \right] = \sum_s \rho^{\mu}(s)\ L_Q^i\left(s, \mu(s)\right) \\ \overline{O}^{\mu,i} &= E^{\mu}\left[L_O^i\left(s, \mu(s)\right) \right] = \sum_s \rho^{\mu}(s)\ L_O^i\left(s, \mu(s)\right) \end{aligned} \tag{101}$$

If $s = (q, x)$ the state of the user *i* then:

$L_E^i\left(s, \mu(s)\right)$ is the energy cost for user *i* at state *s* for the action $\mu(s)$

$L_Q^i\left(s, \mu(s)\right)$ is the delay/queuing cost with $L_Q^i\left(s, \mu(s)\right) = L_Q^i(q) = q$

$L_O^i\left(s, \mu(s)\right)$ is the overflow cost with $L_O^i\left(s, \mu(s)\right) = L_O^i(q, u)$

Then, the optimal scheduling policy μ^* is a solution of the problem:

$$\begin{cases} \min\limits_{\mu} \quad \overline{E}^{\mu} = \sum\limits_{i=1}^{K} \overline{E}^{\mu,i} \\ \text{subject to } \underline{\overline{Q}^{\mu}} = \left[\overline{Q}^{\mu,1}, \overline{Q}^{\mu,2}, ..., \overline{Q}^{\mu,K} \right] \le \underline{\delta_Q} = \left[\delta_Q^1, \delta_Q^2, ..., \delta_Q^K \right] \\ \underline{\overline{O}^{\mu}} = \left[\overline{O}^{\mu,1}, \overline{O}^{\mu,2}, ..., \overline{O}^{\mu,K} \right] \le \underline{\delta_O} = \left[\delta_O^1, \delta_O^2, ..., \delta_O^K \right] \end{cases} \tag{102}$$

This constrained Markov Decision Process problem can be converted into an unconstrained one using the Lagrangian approach to characterize the instantaneous reward or cost per stage:

$$L\left(S_n, \Omega, \underline{\lambda_1}, \underline{\lambda_2}\right) = \sum_{i=1}^{N} L_E^i\left(Z_n^i, U_n^i\right) + \sum_{i=1}^{K} \lambda_{1,n}^i \left(L_Q^i(Q_n^i) - \delta_Q^i\right) + \sum_{i=1}^{K} \lambda_{2,n}^i \left(L_O^i(Q_n^i, U_n^i) - \delta_O^i\right) \tag{103}$$

and the optimization problem is converted to find the optimal policy which minimizes the average Lagrangian expression:

$$\begin{aligned} L_{av}\left(\mu, \underline{\lambda_1}, \underline{\lambda_2}\right) = \limsup_{N\to\infty} \frac{1}{N} E\Bigg[\sum_{n=1}^{N}\sum_{i=1}^{K} L_E^i(S_n, \mu(S_n)) + \sum_{n=1}^{N}\sum_{i=1}^{K} \lambda_1^i \left(L_Q^i\left(S_n, \mu\left(S_n\right)\right) - \delta_Q^i\right) + \\ \sum_{n=1}^{N}\sum_{i=1}^{K} \lambda_2^i \left(L_O^i\left(S_n, \mu\left(S_n\right)\right) - \delta_O^i\right) \Bigg] \end{aligned} \tag{104}$$

As the energy is a convex function of the number of packet sent, the optimal policy μ^* for the constrained problem is also optimal for the unconstrained problem. μ^* minimizes the average Lagrangian $L_{av}\left(\mu, \underline{\lambda_1}, \underline{\lambda_2}\right)$ for a particular choice of the Lagrange multiplier $(\underline{\lambda_1}, \underline{\lambda_2}) = (\underline{\lambda_1^*}, \underline{\lambda_2^*})$. The Lagrange multipliers are the parameters that characterize the flexibility of the scheduler with respect to the delay constraints and the buffer overflow constraints.

5.2.1.2.5.2 *Dynamic programming strategies*

5.2.1.2.5.2.1 Offline approach

The minimization problem of the CMDP in multi-user case is handled by solving dynamic programming equations as in single-user case [Bertsekas07] with the corresponding system states and action spaces for all the users. For the offline approach, the optimal policy is obtained by using the policy iteration technique as previously. But, in this case, finding the optimal Lagrange multipliers should be done iteratively for all the users:

$$\text{For } i = \{1, \ldots, K\} \qquad \begin{aligned} \lambda_{1,n}^i &= \lambda_{1,n}^i + \xi_1 \left(\overline{Q}_n^{\mu^*_{\underline{\lambda_{1,n}}, \underline{\lambda_{2,n}}}, i} - \delta_Q^i \right) \\ \lambda_{2,n}^i &= \lambda_{2,n}^i + \xi_2 \left(\overline{O}_n^{\mu^*_{\underline{\lambda_{1,n}}, \underline{\lambda_{2,n}}}, i} - \delta_O^i \right) \end{aligned} \tag{105}$$

ξ_1 and ξ_2 are fixed update parameters and $\overline{Q}_n^{\mu^*_{\underline{\lambda_{1,n}}, \underline{\lambda_{2,n}}}, i}$ and $\overline{O}_n^{\mu^*_{\underline{\lambda_{1,n}}, \underline{\lambda_{2,n}}}, i}$ are the average user's buffer lengths and the average user's overflows, respectively.

The dynamic offline algorithm can be summarized as following:

Initialization: $n = 1, \quad \underline{\lambda_{1,0}} = \underline{0} \quad \underline{\lambda_{2,0}} = \underline{0}$

Choose an initial distribution ρ_0, choose an initial policy μ_0 and an initial state $s_0 = (0, Z_1)$

for $n = 0$ until $\left\| \underline{\lambda_{1,n+1}} - \underline{\lambda_{1,n}} \right\| < \underline{\theta_1}$ and $\left\| \underline{\lambda_{2,n+1}} - \underline{\lambda_{2,n}} \right\| < \underline{\theta_2}$

for $m = 1$ until $\mu_{m+1}^{\underline{\lambda_{1,n}}, \underline{\lambda_{2,n}}} = \mu_m^{\underline{\lambda_{1,n}}, \underline{\lambda_{2,n}}}$

* Policy evaluation step: find $\beta_{\mu_m}^{\underline{\lambda_1}, \underline{\lambda_2}}$ and $V_m^{\underline{\lambda_1}, \underline{\lambda_2}}(s); s \neq s_0$

solve

$$\begin{cases} V_m^{\underline{\lambda_1},\underline{\lambda_2}}(s) = L\left(\underline{\lambda_1},\underline{\lambda_2},s,\mu_m^{\lambda_1,\lambda_2}(s)\right) + \sum_{s'} p\left(s'\middle|s,\mu_m^{\lambda_1,\lambda_2}(s)\right) V_m^{\underline{\lambda_1},\underline{\lambda_2}}(s') - \beta_{\mu_m}^{\underline{\lambda_1},\underline{\lambda_2}} \quad for \quad s \neq s_0 \\ V_m^{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}(s_0) = 0 \end{cases}$$

*Policy improvement step to find $\mu_{m+1}^{\lambda_{1,n},\lambda_{2,n}}$

$$\mu_{m+1}^{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}(s) = \arg\min_{u\in\Omega}\left[L\left(\underline{\lambda_{1,n}},\underline{\lambda_{2,n}},s,u\right) + \sum_{s'} p\left(s'|s,u\right) V_m^{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}(s')\right] \qquad \forall s \in \mathrm{S}$$

$m = m+1$

End $\mu_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}^{*} = \mu_m^{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}$

* occupation measure calculation

Solve $\rho^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}}(s) = \sum_{s'} p\left(s\middle|s',\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}(s')\right)\rho^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}}(s') \quad \forall\, s \in \mathrm{S}$

* Evaluation of the users performances $\forall\ \mathrm{i} \in \{1,...,K\}$

$$\overline{Q}_n^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}},i} = \sum_s \rho^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}}(s)\ L_Q^i\left(s,\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}(s)\right)$$

$$\overline{O}_n^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}},i} = \sum_s \rho^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}}(s)\ L_O^i\left(s,\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}}(s)\right)$$

* Update the Lagrange multipliers $\forall\ \mathrm{i} \in \{1,...,K\}$

$$\lambda_{1,n}^i = \lambda_{1,n}^i + \xi_1\left(\overline{Q}_n^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}},i} - \delta_Q^i\right)$$

$$\lambda_{2,n}^i = \lambda_{2,n}^i + \xi_2\left(\overline{O}_n^{\mu^*_{\underline{\lambda_{1,n}},\underline{\lambda_{2,n}}},i} - \delta_O^i\right)$$

end

5.2.1.2.5.2.2 Online approach

The online policy is based on the post-decision state learning technique proposed for single user case. However, for multi-user case, the algorithm becomes very complex since the state space cardinality $|\mathbf{S}| = |\mathbf{Q}|^N \times |\mathbf{Z}|^N$ is very large. In order to reduce the number of states, hence the complexity, we divide the global optimization problem into *K* parallel individual problems formulated as:

$$\begin{cases} \min\ \overline{E}_i \\ s.t\ \overline{Q}_i \le \delta_{Qi} \quad \forall i \in \{1,..,K\} \\ \overline{O}_i \le \delta_{Oi} \end{cases} \tag{106}$$

The *K* problems are closely dependent as all the users share the same radio link. In fact, only one user can access the channel at each time slot to offload its data for remote processing. Thus, this forces decisions of local processing or waiting for the other users and may increase their buffer levels. The problem division allows to reduce the number of states from $|\mathbf{S}| = |\mathbf{Q}|^N \times |\mathbf{Z}|^N$ to $|\mathbf{S}| = N|\mathbf{Q}| \times |\mathbf{Z}|$ which reduces considerably the convergence time of the online algorithm. The action space for each user can be expressed as: $\Omega = [-1,\ 0\ ,...,\ U_{\max}]$. In fact at each time slot, a user *i* scheduler makes a decision whether to process locally at the UE or remotely at the SCeNBce $|U_n^i| \le min\ (Q_n^i, U_{\max}^i)$ packets in order to minimize its own energy consumption. At the same time,

SCeNBce/ SCM scheduler should decide to which user the radio resources are allocated in case of offloading decision. The non-scheduled users are forced either to process locally their data or to wait until next decision time slot.

We adopt a scheduling strategy which makes the number of packets to be offloaded as the principle decision criterion. At each time slot, after deciding the number of packets to be processed and the processing way, the users inform the SCeNBce and the SCM about their decision. The scheduling manager gives the channel access to the user having the highest number of packets to offload. In addition, if two users have the same number to transmit, the user who has the better channel condition is allocated the radio resources. This method will promote users who exceed the buffer/delay constraint and have the better radio link quality.

Therefore, the main objective is to design for each user, a packet scheduling policy that minimizes its own average energy consumed while satisfying average queuing/delay constraints and tolerated buffer overflow based on the knowledge on their own applications characteristics and the requirements of other users' applications. This knowledge can be acquired online after a sufficient number of experiences in which the users undergoes not only their own arrival distribution and channel transitions but also those of the other users. It can be approximated via the scheduling policy imposed by the small cell manager.

The proposed online algorithm can be summarized as:

Initialize: $\tilde{V}_0^i(\tilde{s}) = 0 \ \ \forall \tilde{s} \in S \ ; \ \ \lambda_{01}^i = B \ ; \ \tilde{Q}_0^i = B \quad \forall i \in [1,..,K]$

For n : 1 to N (N number of iterations; n is the time slot index)

1- Compute for each user a random packets A_{n+1}^i arriving at the beginning of each time slot and the channel state Z_{n+1}^i

2- Compute at each user the number of packets U_{n+1}^i to be processed at the slot $n+1$ and the way of processing from the possible action set. U_{n+1}^i is the solution that minimizes the following expression based on Bellman equation.

$$U_{n+1}^i = \min_{u \in \Omega} \left[L\left(\lambda_{1n}^i, \lambda_{2n}^i, \left(Q_n^i + A_{n+1}^i, Z_{n+1}^i\right), u\right) + \tilde{V}_n\left(\tilde{Q}_n^i + A_{n+1}^i - |u|, Z_{n+1}^i\right)\right]$$

The SC manager schedules the user $i = \underset{i \in [1,..,K]}{\operatorname{argmax}} \left(U_{n+1}^i\right)$.

If two users i_1 and i_2 have the same number of packets to offload, the scheduler chooses the user that undergo better channel conditions.

For all the users $j \neq i$, a decision is made whether to process one packet locally or wait for the next slot, then $u \in [-1, 0]$ in the previous equation.

3- Compute the energy spent by each user

4- Update the value function of the actual visited post decision state at the slot n $\tilde{S}_n^i = (\tilde{Q}_n^i, Z_n^i) \ \ \forall i \in [1,..,K]$ while keeping all the other value functions unchanged.

$$\tilde{V}_{n+1}^i\left(\tilde{S}_n^i\right) = (1-\alpha_n)\tilde{V}_n^i\left(\tilde{S}_n^i\right) + \alpha_n\left[\left[L\left(\lambda_{1n}^i, \lambda_{2n}^i, \left(Q_n^i + A_{n+1}^i, Z_{n+1}^j\right), U_{n+1}^i\right) + \tilde{V}_n\left(\tilde{Q}_n^i + A_{n+1}^i - \left|U_{n+1}^i\right|, Z_{n+1}^i\right)\right] - \tilde{V}_n^i\left(\tilde{S}_0\right)\right]$$

$$\tilde{V}_{n+1}^i\left(\tilde{S}_n^{i\prime}\right) = \tilde{V}_n^i\left(\tilde{S}_n^i\right) \quad \forall \tilde{S}_n^{i\prime} \neq \tilde{S}_n^i$$

Update the Lagrange multipliers for all the users:

$$\lambda_{1n+1}^{i} = \lambda_{1n}^{i} + \zeta_{1n}^{i}\left(\tilde{Q}_{n}^{i} + A_{n+1}^{i} - \delta_{Q}^{i}\right)$$
$$\lambda_{2n+1}^{i} = \lambda_{2n}^{i} + \zeta_{2n}^{i}\left(\overline{O}^{i} - \delta_{O}^{i}\right) \qquad \forall i \in [1,..,K]$$

5- $n \leftarrow n+1,\ \tilde{Q}_{n+1}^{i} \leftarrow \tilde{Q}_{n}^{i} + A_{n+1}^{i} - \left|U_{n+1}^{i}\right| \quad Z_{n}^{i} \leftarrow Z_{n+1}^{i} \quad \forall i \in [1,..,K]$

End

5.2.1.2.5.3 *Simulation results*

In this section, we compare the performances of the described offline and online dynamic approaches with classical scheduling policies. For these simulation results, we consider 2 users and we take the same parameters values as previous single user section. For simplicity, we consider that the channel is discretized into 3 values $Z = \{-8.47 \text{ dB};\ -3.28 \text{ dB};\ 1.42 \text{ dB}\}$. The maximum buffer size is 15 packets for both users.

In Table 33, we present the channel state distribution and its capacity for offloading case:

Average CSI (dB)	Z= -8.57 dB	Z= -3.28 dB	Z= 1.42 dB	Maximal average individual offloading capacity
-8.57 dB	96.33 %	3.67 %	0.001 %	1.067 packet/slot
-3.28 dB	63.21 %	31.56 %	5.23 %	1.368 packet/slot
1.42 dB	28.74 %	34.47 %	36.79 %	1.713 packet/slot
Maximal number of offloaded packets	1	2	2	

Table 33. The channel distribution and maximal number of offloaded packets for different average CSI

We have observed from simulation results that both offline and online policies respect well the buffering and overflow constraints for both users for any arrival rates under the maximal processing capacity of both users. This maximal capacity is equal to the sum of the maximum between users of their maximal average individual (offloading) capacity (Table 33) and the local processing capacity which is equal to 1.

a- Policy characteristics variations relatively to different arrival rates:
Figure 127 represents the performances of the optimal (offline) policy and online policy for two users and different arrival rate constraints with the same queue length (delay) constraint of 5 packets and an average overflow constraint of 0 packets for both users. We fix the average arrival rate of user 2 to 0.5 and 1 packets/slot and we vary user 1 rate to {0.25 , 0.5 , 0.75 , 1} packets/slot for the same average channel conditions -3.28 dB. We study the effect of the rate variation on the energy consumed by time slot duration for both users.

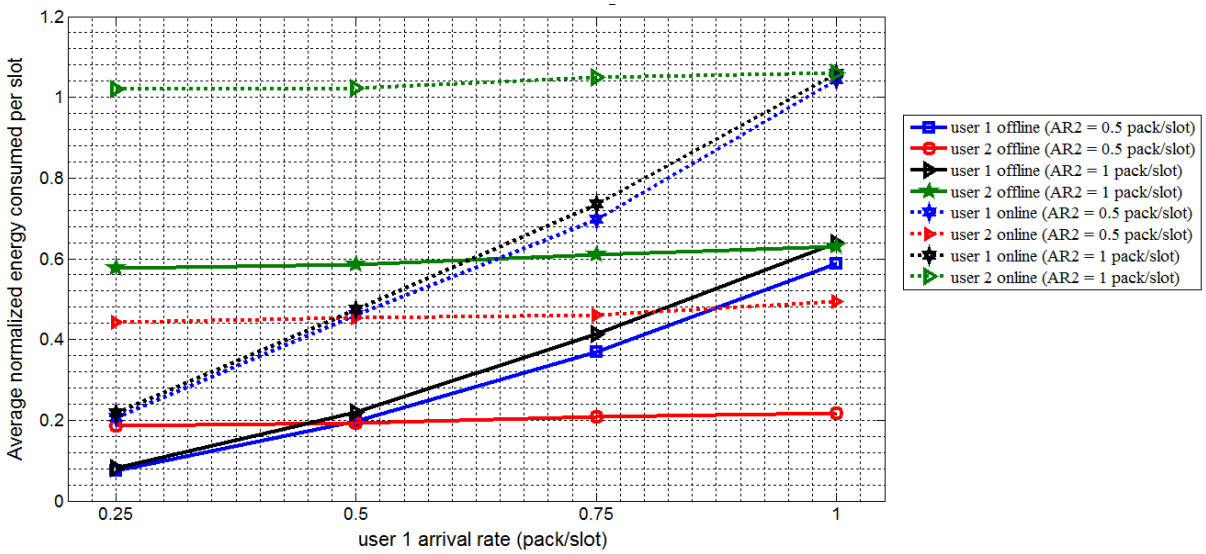


Figure 127. Average energy consumed by both users for different environment scenarios following the offline and online policies.

We can observe that both users consume approximately twice energy when adopting the online policy compared to the offline algorithm. This is due to the fact that the offline policy has a perfect knowledge of the Markov chain transitions while the online policy is based on learning these transitions from a limited number of past experiences that can explore all the states of the Markov chain. This reduces considerably the action space compared to the offline policy that exploit perfectly the radio link transitions yielding a better energy saving.
We note also that the average energy of a user depends closely on the requirements of the other users' applications. This result is observed for both policies. In fact, we can see that user 1 consumes less energy when user's 2 arrival rate is low. In fact, users are more scheduled to offload the data for remote processing exploiting the channel conditions while satisfying the average delay and overflow constraints. Besides, we can note that user 2 consumes more energy with arrival rate of 1 packet/slot when the user's 1 arrival increases. In this case, user 1 can be scheduled to process its data remotely imposing local processing on user 2 since the optimal policy consists in minimizing the instantaneous energy cost based on the sum of the energy consumed by both users.

In the sequel, we study the distributions of the decisions made by the small cell manager to schedule the users for different scenarios.

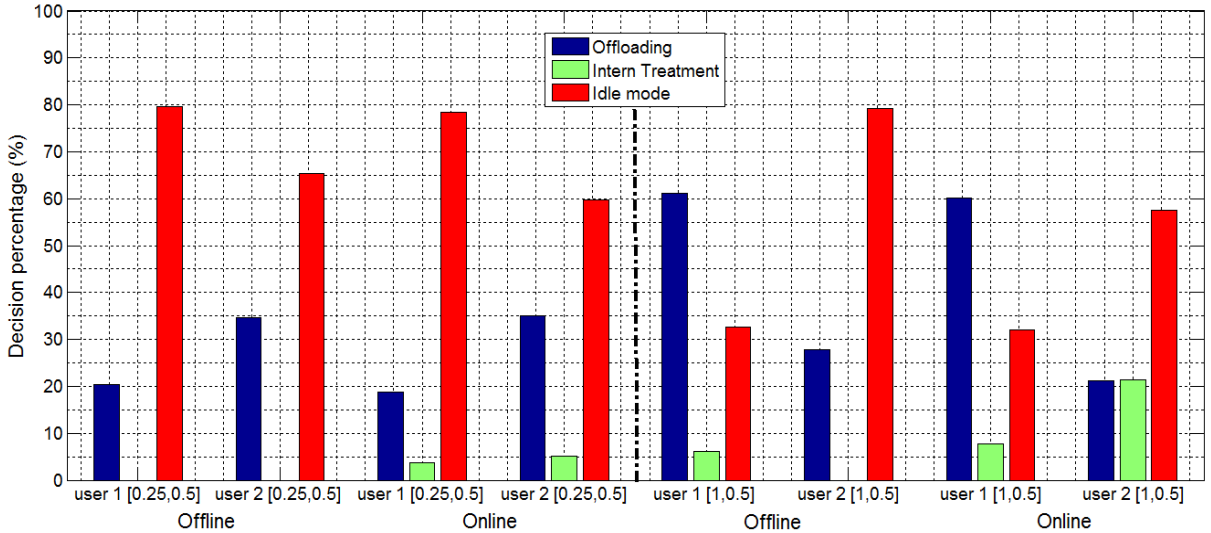


Figure 128. Processing decision distribution for both users with offline and online policies and user's 2 arrival rate = 0.5 pack/slot.

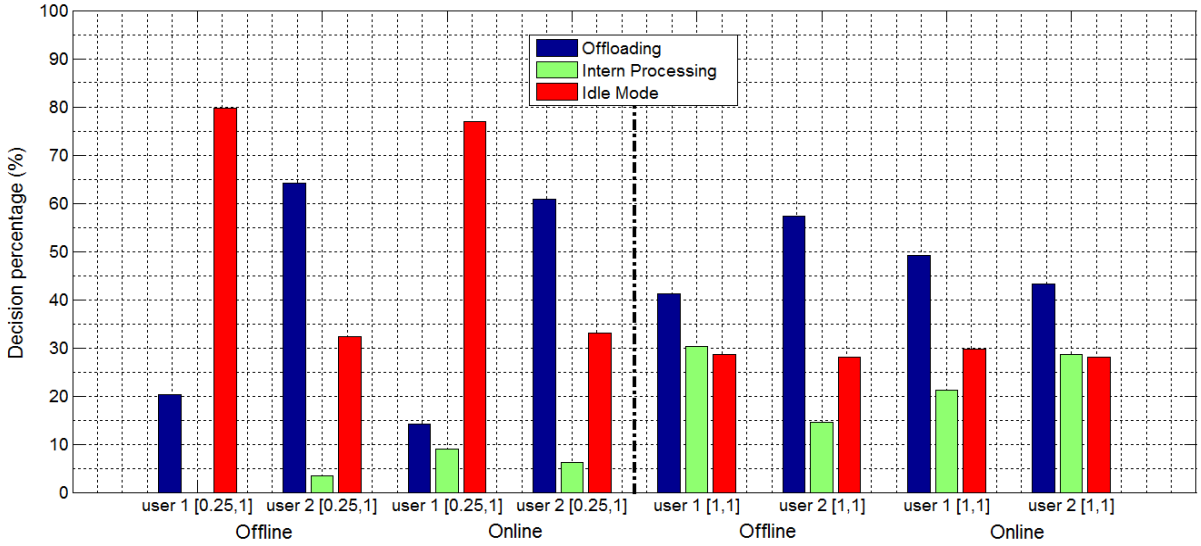


Figure 129. Processing decision distribution for both users with offline and online policies and user's 2 arrival rate = 1 pack/slot.

For both figures, we can notice that for all cases the offline policy have more actions of offloading than local processing compared to the online policy allowing thus to achieve higher energy saving. We can observe that offline policy imposes to both users to be in the Idle mode (waiting for next decision) for many time slots especially for low packet arrivals. This is due to the fact that even if there is instantaneously no arrival, the offline policy forces users to offload the maximal number of packets under good channel conditions. This will keep their queue length far enough from the queue constraint and avoid as much as possible local processing. Local processing is then decided only when the sum of their arrival rates is close to the maximal offloading capacity 1.368 packet/slot in our case.

On the other side, the online policy was not able to avoid local processing even if the arrival rates are low [0.25,0.5] pack/slot. Indeed, the online policy does not have a complete knowledge on the transitions between the states. Therefore, it cannot explore states below a certain margin satisfying the queue constraint. Hence, it forces the user who is not scheduled to process packets locally without waiting for next decision slots where he could be scheduled to offload more packets with lower energy consumption.

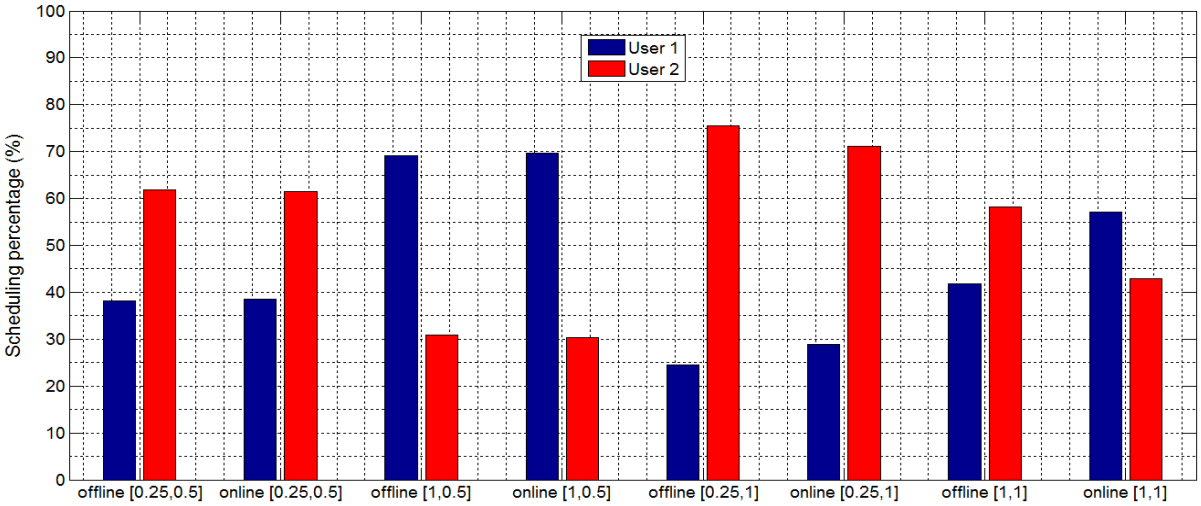


Figure 130. Scheduling decision distribution for both users with offline and online policies and different users' arrival rates.

We can see in Figure 130 that the user with the highest arrival rate is the one that is the most scheduled to offload its data packets for remote processing at the SCeNBce, allowing him to reduce

his energy consumption. For the case where the arrival rates are [1,1] pack/slot, we observe a non-fairness of the scheduler as it favored one user over the other to access the channel forcing the other to process locally more packets, since the channel capacity is limited to support the users requirements. We note also that the scheduler at SCeNBce for the online policy gives similar scheduling percentage to the offline scheduler computed by the dynamic programming algorithm. This proves the efficiency of the online scheduler to find a trade-off between both online individual users' problems. Thus, each user can learn through the past states and past actions on the requirements of the other user in order to compute its packet scheduling policy. Thus, both users can respect the arrival rates, the queuing and overflow constraints imposed by their respective applications.

b- Variation of the channel conditions

In this section, we present the impact of the channel conditions variation on the offline policy. We vary user 1 average channel and we observe the energy consumed by both users at equal arrival rates with maintaining the user 2 average CSI to -3.28 dB.

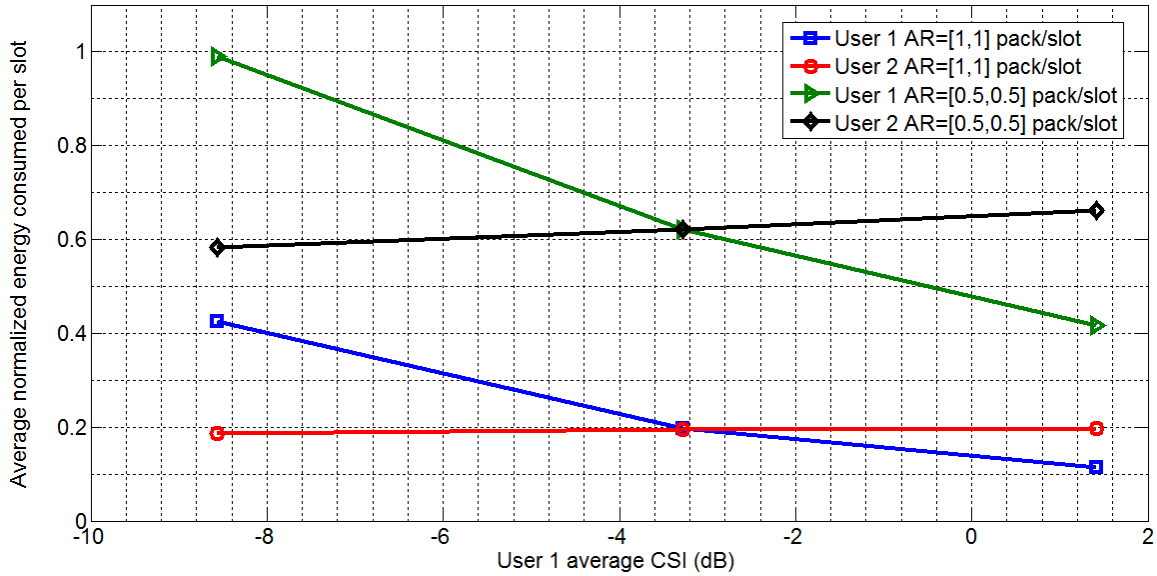


Figure 131. Energy consumption under different CSI for UE 1 and equal arrival rates for both UEs.

It is obvious that when the channel conditions for user 1 are better, more packets can be offloaded resulting in less average energy consumed per slot. In this case, the energy consumption of user 2 increases especially when the arrival rates are high. In fact, for higher user 1 average CSI, he will be scheduled more than user 2 who is forced to process locally its data in order to satisfy its buffering/delay constraints imposed by its application.

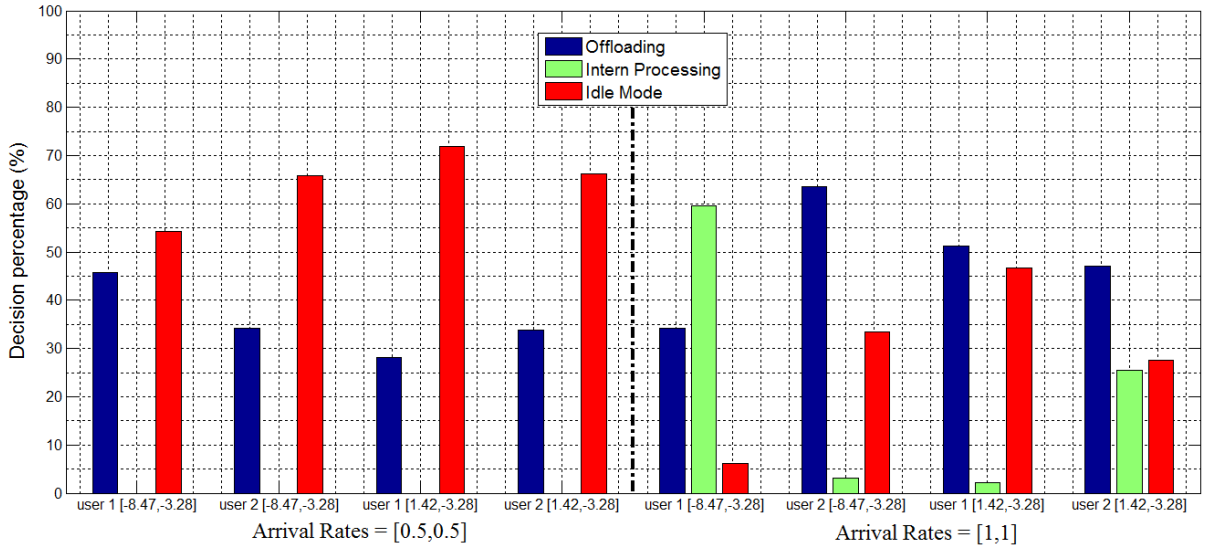


Figure 132. Processing decision distribution for both users under different CSI for UE 1.

Figure 132 illustrates the processing decision distribution on both users for an arrival rate of 0.5 and 1 packet/slot while varying user 1 average CSI. For 0.5 packet/slot case, we can see that user 1 offloads more data packets when his average CSI is low [-8.57 dB]. In fact, according to the channel state distribution (Table 33), we can see that when the average CSI is bad, the number of slots in which the user 1 can offload instantaneously with the maximal rate of 2 packets decreases. This pushes the user to offload only one packet to adjust its buffer to the imposed arrival rate. For better average radio link conditions, the user can offload 71,26 % of the time 2 packets while minimizing its energy consumption. It is then in Idle mode when the channel is in bad states. For user 2, we note that his own processing distribution is similar for both CSI conditions as he can be scheduled to offload according to the channel maximal rate without being in the need to process locally his data.

For 1 pack/slot case, we can observe that for bad channel state CSI= -8.47 dB, user 1 actions are more to process locally than to offload. The time spent in Idle mode is limited to 6% and only 3.67% of the time he can offload 2 packets pushing him to process locally in order to adjust its buffer length to its arrival rate. In this scenario, there is a tradeoff between local processing and offloading for both users. This can be seen also in Figure 133.

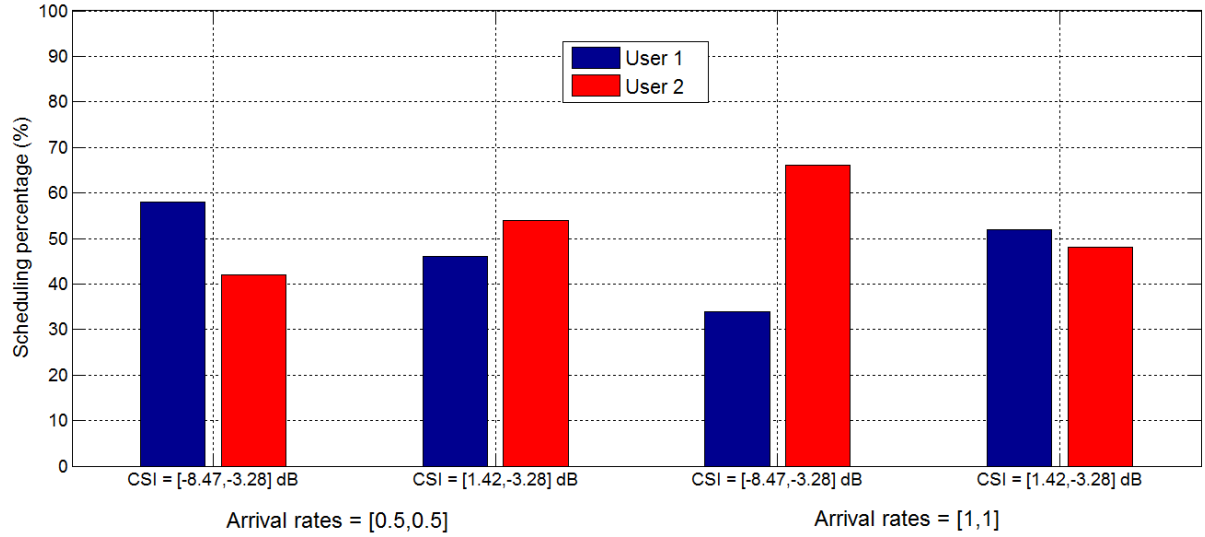


Figure 133. Scheduling decision distribution for both users under different CSI for UE1.

In fact, for lower arrival rate, we can notice that the scheduler favors the user with the worst channel state to offload its packets such that it can adjust its queue length to the arrival rate while satisfying the

delay constraints. For higher arrival rate, as the channel quality cannot satisfy the user's requirements data rate for offloading, the scheduler allocates the radio link for offloading to the user with better channel states and that can offload more data packets. Indeed, this user can offload 2 packets/slot and the other user is forced to process 1 packet/slot locally since it cannot offload more than 1 if it was allocated the radio resources.

5.2.1.2.5.4 *Conclusion*

In this section, we have investigated the offline and online strategies for multiple users, single small cell scenario. The optimal scheduling-computation offloading strategy consists in scheduling the user that has the highest number of packets to offload and the best channel conditions while minimizing the energy consumption of all the users under delay constraints imposed by their applications. Three decisions are possible as for single user case for each user optimization: local processing, offloading or staying idle. We have shown that the offline strategy improves significantly in terms of energy saving compared to the online solution. This is due to the perfect knowledge of state transitions (channel statistics and application properties), while the online policy is based on limited learning on these state information. Moreover, we have noted that the average energy of a user depends closely on the requirements of the other users' applications. In general, processing decisions favor offloading packets by users for remote processing at the SCeNBce when channel conditions are good while satisfying the average delay and overflow constraints. However, only one user is scheduled at a time slot imposing local processing or idle mode on the other users depending on their high or low arrival rates. Thus, this may lead to higher energy consumption at the other users, but still satisfies the optimal energy cost that is the sum of the energy consumed by all users.

5.2.2 **Program-oriented offloading**

The aim of this section is to propose a computation offloading strategy in order to minimize the energy expenditure at the mobile handset necessary to run an application under a latency constraint. We exploit the concept of call graph, which models a generic computer program as a set of procedures related to each other through a weighted directed graph. Our goal is to derive the partition of the call graph establishing which procedures are to be executed locally or remotely. The main novelty of our work is that the optimal partition is obtained jointly with the selection of the transmit power and constellation size, in order to minimize the energy consumption at the mobile handset, under a latency constraint taking into account transmit time, packet drops, and execution time [PDL-Barb13].

5.2.2.1 ***Joint optimization of radio resources and code partitioning in mobile cloud computing***

Our objective in this section is to minimize the energy consumption at the mobile site, under a power budget constraint at the mobile handset, and a latency constraint taking into account the time to transfer the execution and the time necessary to execute the module itself. This constraint is what couples the computation and communication aspects of the problem. We assume that the set of instructions to be executed is available at both MUE and SCeNBce. If not, they are supposed to be downloaded by the server through a high speed wired link.

We start by introducing the notion of call graph of a program, which will be instrumental to formulate our joint optimization of radio and computation resources. The representation of a program through its call graph is instrumental to design an offloading strategy that decides to offload only some modules of the program, depending on the computation requirements of each module, the size of the program state that needs to be exchanged to transfer the execution from one site to the other, and the channel conditions. Then, we formulate the optimization problem as the minimization of the overall energy spent to run a program, under latency and power constraints imposed by the application and the radio interface, respectively. We consider both a single channel and a multi-channel transmission strategy, thus proving that a globally optimal solution can be achieved in both cases with affordable complexity. The theoretical findings are corroborated by numerical results and are aimed to show under what conditions, in terms of call graph topology, communication strategy, and computation parameters, the proposed offloading strategy can provide a significant performance gain.

5.2.2.1.1 Call graph

A *call graph* is a useful representation that models the relations between procedures of a computer program into the form of a directed graph $G=(V,E)$. The call graph represents the call stack as the program executes. Each vertex $v \in V$ represents a procedure in the call stack, and each directed edge $e=(u,v)$ represents an invocation of procedure v from procedure u . The call graph contains all the relationships among the procedures in a program and, in general, it includes auxiliary information concerning for instance the number of instructions within each procedure and the global data shared among procedures. For non-recursive languages with reasonable assumptions on the program structure [Ryder79], the call graph is a directed, acyclic graph.

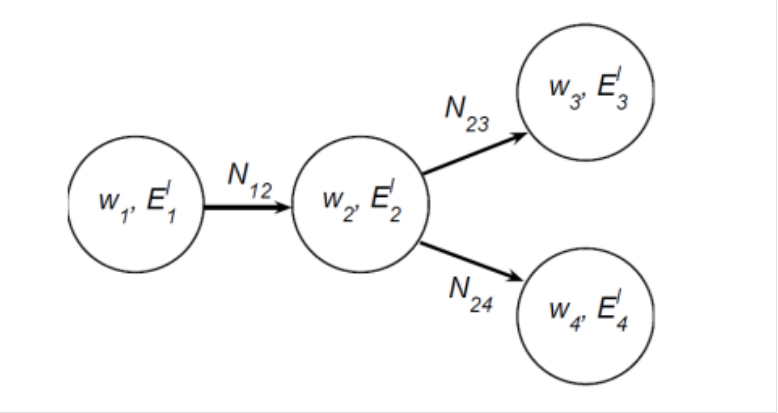


Figure 134. Example of call graph.

In our computation offloading framework, we label each vertex $v \in V$ with the energy E_v^l it takes to execute the procedure locally, and with the overall number of instructions w_v (CPU cycles), which the procedure is composed of. Then, we denote by T_v^l and T_v^r the time it takes to execute the program module locally or remotely, respectively. They are directly related to w_v through the following expressions

$$T_v^l = \frac{w_v}{f_l}, \qquad \text{and} \qquad T_v^r = \frac{w_v}{f_s}, \tag{107}$$

where f_l and f_s are the CPU clock speeds (cycles/seconds) at the MUE and at the server, respectively.

At the same time, each edge $e=(u,v)$ is characterized by a label describing the number of bits $N_{u,v}$ representing the size of the program state that needs to be exchanged to transfer the execution from node u to node v.

In general, some procedures cannot be offloaded, like, e.g., the program modules controlling the user interface or the interaction with I/O devices. Examples of these categories include codes that determine the location of a smartphone by reading from the GPS device, or code that uses a network connection to perform an e-commerce transaction, e.g., to purchase an item from an online store.

The set of procedures that should be executed locally, at the mobile site, is denoted by V_l. An example of call graph of a generic application, composed of $V=4$ nodes and $E=3$ edges, is shown in Figure 134. The call graph has a tree structure, where node 1 is the root node.

Our goal is now to formulate an optimization problem aimed at determining which modules of an application's call graph should be executed locally and which ones should be executed remotely. Intuitively speaking, the modules more amenable for offloading are the ones requiring intensive computations and limited exchange of data to transfer the execution from one site to the other. Our goal now is to make this intuition the result of an optimization procedure. To this end, we formulate the offloading decision problem jointly with the selection of the transmit power and the constellation size used for transmitting the program state necessary to transfer the execution from the mobile

handset to the cloud or vice versa. The objective is to minimize the energy consumption at the mobile site, under a power budget constraint at the mobile handset, and a latency constraint taking into account the time to transfer the execution and the time necessary to execute the module itself. This constraint is what couples the computation and communication aspects of the problem. We assume that the set of instructions to be executed is available at both MUE and SCeNBce. If not, they are supposed to be downloaded by the server through a high speed wired link.

Let us indicate with I_v the indicator variable, which is equal to one, if the procedure v of the call graph is executed remotely, or zero, if it is executed locally. To incorporate the fact that the program initiates and terminates at the mobile site, we introduce two auxiliary vertices, namely the initial and terminating vertices, whose indicator variables are set to zero by default. The addition of these two nodes gives rise to an extended edge set that comprises all the edges of the original call graph, plus the edges from the initiating node to the call graph, and the edges from the call graph to the terming node. We also denote by $p_{u,v}$ the power spent to transmit the program state between the procedures u and v. The set of all powers is collected in the vector $\mathrm{p} = \{\mathrm{p_{u,v}}\}_{(\mathrm{u,v})\in \mathrm{E}} \in \mathrm{R}^{\mathrm{card(E)}}$, where $\mathrm{card}(E)$ is the cardinality of set E. To decide which modules of the call graph should be executed remotely, we need to solve the following optimization problem:

$$
\begin{aligned}
[\mathbf{P.1}] \quad & \min_{I,\mathbf{p}} \quad \sum_{v\in V}(1-I_v)\cdot E_v^l + \sum_{(u,v)\in E} f_{u,v}\left(I_u, I_v, p_{u,v}\right) \\
& \text{s.t.} \sum_{v\in V}\left[(1-I_v)T_v^l + I_v T_v^r\right] + \sum_{(u,v)\in E} g_{u,v}\left(I_u, I_v, p_{u,v}\right) \le L \\
& \qquad I_v \in \{0,1\}, \qquad I_v = 0, \quad \forall v \in V_l, \\
& \qquad 0 < p_{u,v} \le P_T \qquad \forall\,(u,v)\in E,
\end{aligned}
\tag{108}
$$

where $I = \{I_v\}_{v\in V}$. The objective function in (108) represents the total energy spent by the MUE for executing the application. In particular, the first term in (108) is the sum (over all the vertices of the call graph) of the energies spent for executing the procedures locally, whereas the second term is the sum (over the edges of the call graph) of the energies spent to transfer the execution from the MUE to the SCeNBce. The function $f_{u,v}\left(I_u, I_v, \mathrm{p}_{u,v}\right)$ in (108), reported in Table 34a, describes the energy spent by the MUE when procedure u calls procedure v and its dependency on the indices u and v. Note that an energy cost occurs only if the two procedures u and v are executed at different locations, i.e. $I_u \neq I_v$. More specifically, if $I_u = 0$ and $I_v = 1$, the energy cost is equal to the energy $J_{u,v}(p_{u,v})$ needed to transmit the program state $N_{u,v}$ from the MUE to the SCeNBce, whereas, if $I_u = 1$ and $I_v = 0$, the cost is equal to the energy $\varepsilon_{u,v}$ needed by the MUE to decode the $N_{u,v}$ bits of the program state transmitted back by the SCeNBce. The cost $\varepsilon_{u,v}$ is not a function of the MUE's transmitted power and it depends only on the size of the program state $N_{u,v}$.

The constraint in (108) is a latency constraint and it contains three terms: the first term is the time needed to compute the procedures locally, the second is the time needed to compute the procedures remotely, and the third term is the delay resulting from transferring the program state from one site (e.g., the MUE) to the other (e.g., the cloud). The constant L represents the maximum latency dictated from the application.

The delay function $g_{u,v}\left(I_u, I_v, \mathrm{p}_{u,v}\right)$ in (108), reported in Table 34b, represents the delay associated to letting procedure u call procedure v. From Table 34b, we note that no delay occurs if the two procedures u and v are both executed in the same location, i.e. $I_u = I_v$. Furthermore, if $I_u = 0$ and $I_v = 1$, the delay is equal to the time $D_{u,v}(p_{u,v})$ needed to transmit the program state $N_{u,v}$ from the MUE to the SCeNBce, whereas, if $I_u = 1$ and $I_v = 0$, the delay is equal to the time $\gamma_{u,v}$ needed by the MUE to get the $N_{u,v}$ status bits back from the SCeNBce. The delay function $D_{u,v}(p_{u,v})$ in

Table 34 will be made explicit later on. The term $\gamma_{u,v}$ is not a function of the MUE's transmitted power and it depends only on the size of the program state $N_{u,v}$ to be transferred. The second constraint in (108)specifies that the variables I_v are binary and that for all procedures contained in the set V_I which is the set of procedures that are to be executed locally, $I_v = 0$. The last constraint, in (108), is a power budget constraint on the maximum transmit power P_T .

I_u	I_v	$f_{u,v}(I_u, I_v, p_{u,v})$
0	0	0
0	1	$J_{u,v}(p_{u,v})$
1	1	0
1	0	$\varepsilon_{u,v}$

a)

I_u	I_v	$g_{u,v}(I_u, I_v, p_{u,v})$
0	0	0
0	1	$D_{u,v}(p_{u,v})$
1	1	0
1	0	$\gamma_{u,v}$

b)

Table 34. a) Energy cost function $f_{u,v}\left(I_u, I_v, p_{u,v}\right)$; b) Delay function $g_{u,v}\left(I_u, I_v, p_{u,v}\right)$.

The problem formulation in (108) is an extension of the MAUI strategy from [Maui10], where it was proposed an integer linear program aimed at finding the optimal program partitioning strategy that minimizes the MUE's energy consumption, subject to a latency constraint. The difference with respect to our work is that in [Maui10], the quantities $D_{u,v}(p_{u,v})$ and $J_{u,v}(p_{u,v})$ were assumed to be constant, whereas in our case those quantities are optimized jointly with the program partition, acting on the transmit power and on the constellation size.

The quantities $D_{u,v}(p_{u,v})$ and $J_{u,v}(p_{u,v})$ are functions of channel state and transmit power. We assume an adaptive modulation scheme that selects the QAM constellation size as a function of channel conditions and computational requirements. As a consequence, the minimum time $D_{u,v}(p_{u,v})$ necessary to transmit $N_{u,v}$ bits of duration T_b over an additive white Gaussian noise (AWGN) channel is

$$D_{u,v}(p_{u,v}) = \frac{N_{u,v}T_b}{\log_2\left(1+\frac{p_{u,v}|h|^2}{\Gamma(\mathrm{BER})d^{\beta}N_0}\right)}, \tag{109}$$

where the denominator represents the number of bits/symbol, $p_{u,v}$ is the transmit power, d is the distance between MUE and SCeNBce, β is the path loss exponent, N_0 denotes the noise power and h is the channel fading coefficient (normalized to distance); $\Gamma(\mathrm{BER}) = -(2/3)\log(5\mathrm{BER})$ represents the so called SNR margin (whenever the base is unspecified, $\log(x)$ has to be intended as the natural base logarithm), introduced to meet the desired target bit error rate (BER) with a QAM constellation. Note that the gap factor $\Gamma(\mathrm{BER})$ is valid under the assumption $\Gamma(\mathrm{BER}) > 0$, i.e. $\mathrm{BER} < 1/5$.

In practice, the time necessary to let the server receive the correct input bits is often greater than (109) because of retransmission of erroneous packets. Let us denote by S the random variable indicating the number of retransmissions over the wireless link and with P_e the corresponding packet error rate.

Assuming independent errors over different packets, the probability of transmitting a packet S times is given by $\mathbb{P}\{S = s\} = P_e^{s-1}(1-P_e)$. The expected number of transmissions is then

$$\bar{S} = \sum_{s=1}^{\infty} s\, P_e^{s-1}(1-P_e) = \frac{1}{1-P_e}. \tag{110}$$

The relation between BER and P_e is dictated by the channel coding scheme adopted in the link between the MUE and the SCeNBce. The average delay associated to the correct reception of $N_{u,v}$ bits at the server side, incorporating packet duration and retransmissions, is then

$$\bar{D}_{u,v}(p_{u,v}) = \bar{S}\, D_{u,v}(p_{u,v}) = \frac{N_{u,v}T_b}{(1-P_e)\log_2\left(1+\dfrac{p_{u,v}\,|h|^2}{\Gamma(\mathrm{BER})d^{\beta}N_0}\right)} \tag{111}$$

Expression (111) shows that offloading may be advantageous if the distance d between MUE and SCeNBce is sufficiently small, so that the transmission rate is high. This is a further justification for favoring the proximity radio access to the cloud through SCeNBce's. Now, setting

$$a = \frac{|h|^2}{\Gamma(\mathrm{BER})d^{\beta}N_0} \tag{112}$$

and adopting natural logarithms, (111) can be rewritten as

$$\bar{D}_{u,v}(p_{u,v}) = \frac{N'_{u,v}}{\log\left(1+ap_{u,v}\right)}, \tag{113}$$

where $N'_{u,v} = \dfrac{N_{u,v}T_b\log 2}{(1-P_e)}$. Expression (113) represents the average delay associated to the data transfer required to execute procedure v remotely, when called by procedure u . Thus, exploiting (113) , the average energy $\bar{C}_{u,v}$, associated to the transfer of the program state $N_{u,v}$, is given by

$$\bar{J}_{u,v}(p_{u,v}) = p_{u,v}\bar{D}_{u,v}(p_{u,v}) = \frac{N'_{u,v}p_{u,v}}{\log\left(1+ap_{u,v}\right)} \tag{114}$$

Thus, using expressions (107), (113) and (114) the solution of problem [P.1] provides a *joint* optimization of radio resources (power and bits/symbol) and code partitioning for mobile computation offloading.

5.2.2.1.2 Single channel transmission

Let us consider the optimization problem in (108), i.e.

$$
\begin{aligned}
[\mathbf{P.1}] \quad & \min_{I,\,\mathbf{p}} \quad \sum_{v\in V}(1-I_v)\cdot E_v^l + \sum_{(u,v)\in E} f_{u,v}\left(I_u, I_v, p_{u,v}\right) \\
& \text{s.t.} \sum_{v\in V}\left[(1-I_v)T_v^l + I_v T_v^r\right] + \sum_{(u,v)\in E} g_{u,v}\left(I_u, I_v, p_{u,v}\right) \le L \\
& \quad I_v \in \{0,1\}, \qquad I_v = 0, \quad \forall v \in V_l, \\
& \quad 0 < p_{u,v} \le P_T \qquad \forall\,(u,v)\in E,
\end{aligned}
\tag{115}
$$

where $I = \{I_v\}_{v\in V}$. The objective function in (115) represents the total energy spent by the MUE for executing the application. In particular, the first term in (115) is the sum (over all the vertices of the call graph) of the energies spent for executing the procedures locally, whereas the second term is the sum (over the edges of the call graph) of the energies spent to transfer the execution from the MUE to the SCeNBce.

Since the state variables I_u are integer, problem [P.1] is inherently a mixed integer programming problem [Nemh-Wolsey88]. Hence, its solution in principle requires to explore all possible combinations in $\boldsymbol{I}$, solve for the power vector $\mathbf{p}$, and then compare the final values of the objective function in order to choose the best configuration. The size of the problem might explode because, in principle, the number of combinations grows exponentially with the number of vertices V of the call graph. Nevertheless, a series of simplifications are possible. The first useful remark is that, with respect to the integer variables I_v , the problem is a *linear* (binary) integer programming problem. This remark is useful because there exist efficient algorithms, such as the branch and bound algorithm for example, to solve a linear integer problem with reasonable complexity [Nemh-Wolsey88]. An alternative approach to handle the integer part is to resort to a backward induction approach to find out the optimal partition [Fuden-Tir91]. To this end, starting from the call graph, we build a tree containing all possible offloading choices, for each module. Then, starting from the leaves of the tree, we explore the tree back up to the root, by removing all the branches which are suboptimal. Of course, the overall complexity depends on the granularity of the call graph construction: A fine-grain model provides better performance, but with a much higher cost; conversely, a coarse-grain call graph provides worse performance, but with affordable complexity. An important simplification of the problem comes from observing that, for any set of integer values I_v , the remaining optimization over the power coefficients is a convex problem. This means that its solution is unique and it can be achieved with very efficient numerical algorithms. We prove now the previous statement.

Let us consider a generic combination $c \in C$, where C is the set of all possible combinations of the binary variables I_v , $v \in V$. For each combination $\boldsymbol{C}$, the value of the variables I_v is fixed to some value I_v^c , and problem [P.1] becomes a radio resource allocation problem where we are attempting to minimize the average energy spent for transmission subject to a constraint on the average transmission delay, i.e.,

$$
\begin{aligned}
[\mathbf{P.2}] \quad & \min_{\mathbf{p}_c} \quad \sum_{(u,v)\in I_c} \frac{N_i' p_{u,v}}{\log\left(1+ap_{u,v}\right)} \\
& \text{s.t.} \quad \sum_{(u,v)\in I_c} \frac{N_{u,v}'}{\log\left(1+ap_{u,v}\right)} \le L' \\
& \qquad 0 < p_{u,v} \le P_T, \qquad \forall\,(u,v)\in I_c.
\end{aligned}
\tag{116}
$$

where, for any combination $\boldsymbol{C}$, I_c is the set of edges $(u,v)\in E$ for which $I_u^c = 0$ and $I_v^c = 1$; $\mathbf{p}_c = \{p_{u,v}\}_{(u,v)\in I_c}$ is vector containing all the powers, and

$$L' = L - \sum_{v \in V} \left[(1 - I_v^c) \frac{w_v}{f_l} + I_v^c \frac{w_v}{f_s} \right] - \sum_{(u,v) \in I_c'} \gamma_{u,v} \tag{117}$$

where I_c' is the set of edges $(u,v) \in E$ for which $I_u^c = 1$ and $I_v^c = 0$. Going from [P.1] to [P.2], we have eliminated all the terms that do not depend explicitly on the transmit powers, since they do not affect the optimization problem [P.2]. The problem in [P.2] is still nonconvex because the objective function is actually concave in the power allocation vector $\mathbf{p}_c$. Nevertheless, we can prove the following result:

Theorem 1. If the following conditions

$$L' > 0 \qquad \text{and} \qquad a \geq \frac{e^{\frac{\sum_{(u,v) \in I_c} N'_{u,v}}{L'}} - 1}{P_T}, \tag{118}$$

are satisfied, then: (a) the feasible set of problem [P.2] is non-empty; (b) the constrained energy minimization problem in [P.2] can be reduced to an equivalent convex problem (see [P.5] in [PDL-Barb13]); (c) the latency constraint in [P.2] is always satisfied with strict equality.
See [PDL-Barb13] for the proof.

Remark 2: By virtue of Theorem 1 (see [PDL-Barb13]), the delay constrained energy minimization problem in [P.2] is equivalent to a convex problem with strictly convex objective function. Thus, if the feasible set is non-empty, the problem admits a unique global solution that can be found using numerically efficient algorithms [Boyd04].

The main steps for solving the joint optimization of radio resources and code partitioning for mobile cloud computation offloading are summarized in the following Algorithm.

Algorithm: Joint Optimization of Radio Parameters and Call Graph's Partitioning

1. Given a generic call graph, compute the set of combinations $C = \{c\}$.
2. For every combination $\boldsymbol{C}$, check the feasibility of problem [P.2] through the relation (118);
   a) if the problem is feasible, compute its optimal solution $\mathrm{p}_c^* = \left\{\mathrm{p}_{u,v}^*\right\}_{(u,v) \in \mathrm{I}_c} \in \mathrm{R}^{\mathrm{card}(\mathrm{I}_c)}$ by solving the problem [P.5]. Furthermore, evaluate the overall energy spent to run the application, for the given combination $\boldsymbol{C}$, as:

   $$E_c = E_c^l + E_c^r(\mathbf{p}_c^*) = \sum_{v \in V} (1 - I_v^c) \cdot E_v^l + \sum_{(u,v) \in E} f_{u,v}\left(I_u^c, I_v^c, \mathbf{p}_{u,v}^*\right)$$

   where $f_{u,v}\left(I_u^c, I_v^c, \mathbf{p}_{u,v}^*\right)$ is defined as in Table 34 (a), with $\bar{J}_{u,v}(\mathbf{p}_{u,v})$ given by (114) .
   b) if the problem is not feasible, set $E_c = E_g^l$, where E_g^l is the energy spent by the MUE to run the entire application locally.
3. Select the pair $(c^*, \mathbf{p}_c^*) = \arg\min_c E_c$.

5.2.2.1.3 Multiple channel transmission

We now extend the previous offloading optimization strategy to the wideband channel case, where the MUE transmits over a set of K parallel sub bands. In this case, we have the extra degree of freedom on how to distribute the transmit power across the parallel channels. In the multi-channel case, the

average delay associated to the correct reception of $N_{u,v}$ bits at the server side, incorporating packet duration and retransmissions, is

$$\bar{D}_{u,v}(\mathbf{p}_{u,v}) = \frac{N_{u,v}T_b}{(1-P_e)\sum_{k=1}^{K}\log_2\left(1+\frac{p_{u,v}^k |h_k|^2}{\Gamma(\mathrm{BER})d^{\beta}N_0}\right)} = \frac{N_{u,v'}}{\sum_{k=1}^{K}\log\left(1+a_k p_{u,v}^k\right)} \tag{119}$$

with

$$N_{u,v'} = \frac{N_{u,v}T_b \log 2}{(1-P_e)} \quad \text{and} \quad a_k = \frac{|h_k|^2}{\Gamma(\mathrm{BER})d^{\beta}N_0}, \tag{120}$$

where h_k and $p_{u,v}^k$ denote the fading coefficient and the power transmitted over the k-th sub channel, respectively, and $\mathbf{p}_{u,v} = [p_{u,v}^1, \ldots, p_{u,v}^K]^T$. Exploiting (119), the average energy cost $\bar{J}_{u,v}(\mathbf{p}_{u,v})$, associated to the transfer of the program state $N_{u,v}$, is given by

$$\bar{J}_{u,v}(\mathbf{p}_{u,v}) = \bar{D}_{u,v}(\mathbf{p}_{u,v})\sum_{k=1}^{K} p_{u,v}^k = \frac{N_{u,v'}\sum_{k=1}^{K} p_{u,v}^k}{\sum_{k=1}^{K}\log\left(1+a_k p_{u,v}^k\right)} \tag{121}$$

Let us define $\mathrm{p} = \{\mathrm{p_{u,v}}\}_{(\mathrm{u,v})\in\mathrm{E}} \in \mathrm{R}^{\mathrm{K}\cdot card(\mathrm{E})}$ .Using the expressions (119) - (121), the joint optimization of radio resources and code offloading, in the case of transmission over a wideband channel, can be written as:

$$\begin{aligned}
[\mathrm{P.3}] \quad & \min_{I,\mathbf{p}} \sum_{v\in V}(1-I_v)\cdot E_v^l + \sum_{(u,v)\in E} f_{u,v}\left(I_u, I_v, \mathbf{p}_{u,v}\right) \\
& \text{s.t.} \sum_{v\in V}\left[(1-I_v)T_v^l + I_v T_v^r\right] + \sum_{(u,v)\in E} g_{u,v}\left(I_u, I_v, \mathbf{p}_{u,v}\right) \le L \\
& I_v \in \{0,1\}, \qquad I_v = 0, \quad \forall v \in V_l, \\
& \sum_{k=1}^{K} p_{u,v}^k \le P_T, \qquad \forall\,(u,v)\in E,
\end{aligned} \tag{122}$$

where $f_{u,v}\left(I_u, I_v, \mathbf{p}_{u,v}\right)$ and $g_{u,v}\left(I_u, I_v, \mathbf{p}_{u,v}\right)$ are defined as in Table 34a) and b), respectively, and P_T denotes the power budget constraint on the sum of powers that the MUE can transmit over all the sub channels. As for [P.1], the optimization problem in [P.3] is a mixed integer problem. Then, let us consider a generic combination $c \in C$, where C is the set of all possible combinations of the binary variables I_v, $v \in V$. For each combination c, the value of the variables I_v is fixed to some value I_v^c, and problem [P.3] becomes the radio resource allocation problem

$$[P.4] \quad \min_{\mathbf{p}_c} \quad \sum_{(u,v)\in I_c} \frac{N_{u,v'} \sum_{k=1}^{K} p_{u,v}^{k}}{\sum_{k=1}^{K} \log\left(1+a_k p_{u,v}^{k}\right)}$$

$$\text{s.t.} \quad \sum_{(u,v)\in I_c} \frac{N_{u,v'}}{\sum_{k=1}^{K} \log\left(1+a_k p_{u,v}^{k}\right)} \le L' \qquad (123)$$

$$\sum_{k=1}^{K} p_{u,v}^{k} \le P_T, \qquad \forall\,(u,v)\in I_c,$$

where I_c is the set of edges $(u,v)\in E$ for which $I_u^c = 0$ and $I_v^c = 1$ given the combination $\boldsymbol{C}$. Furthermore, the optimization vector is $\mathbf{p}_c = \{\mathbf{p}_{u,v}\}_{(u,v)\in I_c}$, where $\mathbf{p}_{u,v} = [p_{u,v}^1,\ldots,p_{u,v}^K]^T$. The delay constraint L' is the same as (117). Again, the problem in [P.4] is not convex. However, it can be proved the following result:

Theorem 2. If the following feasibility conditions are satisfied:

$$L' > 0 \qquad \text{and} \qquad \sum_{k=1}^{K} \log(1+a_k p_k^*) \ge \frac{\sum_{(u,v)\in I_c} N'_{u,v}}{L'}, \qquad (124)$$

where $\mathbf{p}^* = \{p_k^*\}_{k=1}^K$ is the water-filling solution (see Appendix B in [PDL-Barb13]), (a) problem [P.4] admits a non-empty feasible set;(b) problem [P.4] can be reduced to an equivalent convex problem (see the problem [P.1] in Appendix B [PDL-Barb13]), which is such that any local optimum is also globally optimal.
For the proof see Appendix B in [PDL-Barb13].

To solve the problem in [P.3], we propose an algorithm totally similar to the one we have proposed in the previous subsection. The only difference is that, for each combination $\boldsymbol{C}$, we check the feasibility condition (124), and we solve the optimization problem in [P.4], instead of [P.2] .

5.2.2.1.4 Numerical results

In this section we provide some numerical results to validate our theoretical findings and to assess the performance of the proposed algorithms. First, we show the effect of the call graph on the performance of the optimization algorithm. Second, we illustrate the role played by the random wireless channel, thus showing how the performance of the algorithm is affected by radio parameters such as, e.g., the distance and the number of independent channels. Finally, we apply the proposed method to the call graph of a face recognition application, in the case the MUE transmits over a set of parallel sub bands.

5.2.2.1.4.1 *Numerical Example 1-Performance*

The performance of the proposed offloading strategy is affected by several factors. The first important factor is of course the call graph topology and its parameters. As an example, let us consider three different examples of call graphs, as illustrated in Figure 135. To grasp the dependence of the performance on the graph's parameters, we consider for simplicity three directed call graphs having the same topology, but different parameters. We consider the case where the root node must be computed only locally. This represents the case where the node handles the interface with the user. All other nodes of the graphs can be all offloaded, if needed. All the nodes in the three graphs have the same set of energies E_v^l (mJoule) and number of instructions w_l (Mcycles), associated to the local

computation of a method. What is different from one graph to the other is the number $N_{u,v}$ of KByte associated to the size of the program state at each edge of the graph, or the parameters $\varepsilon_{u,v}$ and $\gamma_{u,v}$, which are the energy and the time needed by the MUE to get the $N_{u,v}$ bits of the program state transmitted back by the SCeNBce, respectively. In particular, Graph 2 has different $N_{u,v}$'s with respect to Graph 1, but same $\varepsilon_{u,v}$'s and $\gamma_{u,v}$'s, whereas Graph 3 has different $\varepsilon_{u,v}$'s and $\gamma_{u,v}$'s with respect to Graph 2, but same $N_{u,v}$'s.

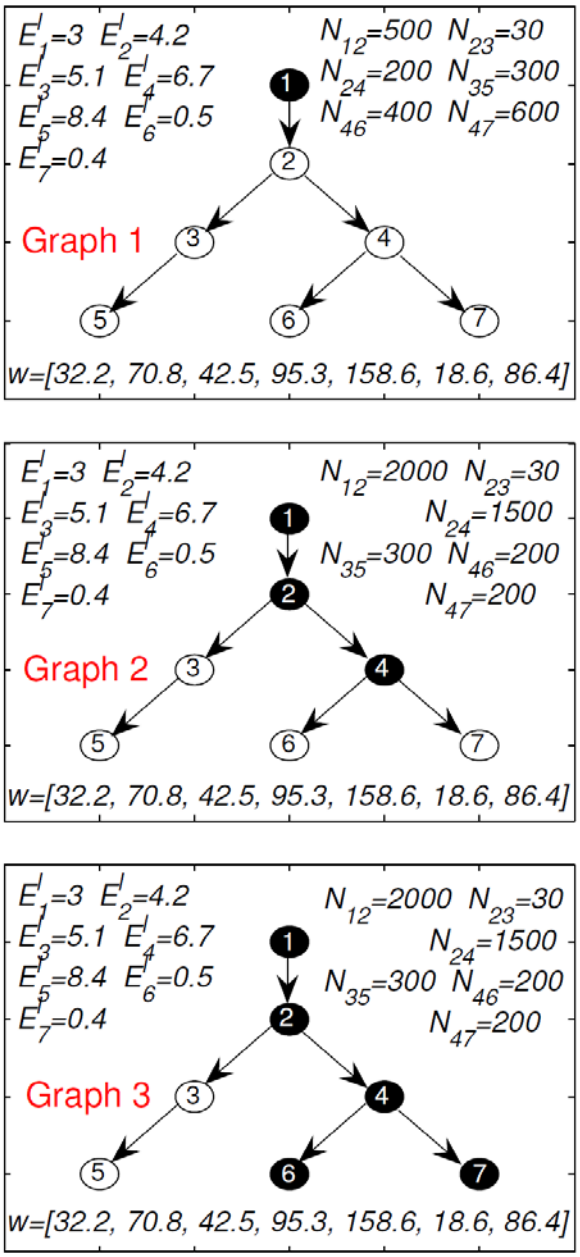


Figure 135. Examples of optimal graph's partitions.

Let us indicate with 0 the index of the MUE, which acts equivalently as an end-node for the call graph. For simplicity and to grasp the main features, for graphs 1 and 2, we have $\varepsilon_{u,v} = \gamma_{u,v} = 0.05$ for all $(u,v) \in E$. The parameters of Graph 3 are chosen as before except that $\varepsilon_{u,v} = \gamma_{u,v} = 0.5$ for $(u,v) = \{(6,0),(7,0)\}$. This means that the output of the procedures 6 and 7, which must be

transmitted back to the MUE if the procedures are computed remotely, is much larger than before. The colors associated to the nodes in Figure 135 denotes the final result of the proposed joint optimization algorithm for the specific graph. In particular, the black nodes are computed locally, whereas the white nodes are offloaded to the server. In this simulation, we consider the presence of a single channel between MUE and SCeNBce. For the joint optimization problem on [P.1], the latency constraint L is chosen equal to the time needed by the MUE to compute the entire program locally, and the power budget constraint is $P_T = 0.01$ Watt. The local CPU clock speed is chosen as $f_l = 10^8$ cycles/s, whereas the server speed is equal to $f_s = 10^{10}$ cycles/s. The normalized channel coefficient a in (112) is set equal to 500, and the bit duration T_b is 10^{-6}.

As we can notice from Figure 135, slightly different call graphs (in terms of number of bits $N_{u,v}$, or parameters $\varepsilon_{u,v}$'s and $\gamma_{u,v}$'s) lead to completely different results. In the first case (Graph 1), all the nodes (except the root of course) are offloaded to the server. This happens because the overall energy that the MUE should have spent for computing the entire application locally is much greater than the energy needed to transmit the initial state of size $N_{12} = 500$ KByte. In this case, it might be that, for each individual method after node 2, its remote execution is more expensive than its local execution, and yet remote execution can save energy by offloading all the program. It is indeed important to remark that the result of the optimization is globally optimal (i.e., across the entire program) rather than locally optimal (i.e., relative to a single method invocation).

Let us consider now the case of Graph 2 in Figure 135, where the size N_{12} and N_{24} of the program state are larger than before. In this case, the optimal solution is the one shown in Figure 135 (Graph 2), where only four methods are offloaded. This happens because, from an energetic point of view, it has become non convenient to transmit the large states N_{12} or N_{24} to the SCeNBce. Furthermore, once a computation is offloaded, the result must be transmitted back to the SCeNBce. If the result yields too many bits, it may not be convenient to offload that procedure. This is exactly what happen in the case of Graph 3 in Figure 135. Indeed, increasing the values of the output size of procedures 6 and 7 (by acting on $\varepsilon_{u,v}$'s and $\gamma_{u,v}$'s) with respect to the case of Graph 2, the optimal solution changes as shown in Figure 135 (Graph 3).

A fundamental factor affecting the performance of the joint optimization algorithm is the wireless fading channel between MUE and SCeNBce. For this reason, we check the performance of the optimization algorithm as a function of the distance between MUE and SCeNBce, considering a Multiple-Input-Multiple-Output (MIMO) transmission scheme. In Figure 136, we report the average energy spent for processing versus the distance between MUE and SCeNBce, for different MIMO configurations.

The results are averaged over 200 independent realizations. In the presence of flat fading, the normalized channel coefficient can be written as

$$a = \frac{\alpha}{\Gamma(\mathrm{BER}) d^{\beta} N_0} \tag{125}$$

where α is a random variable, whose probability density function (pdf) depends on the MIMO communication strategy.

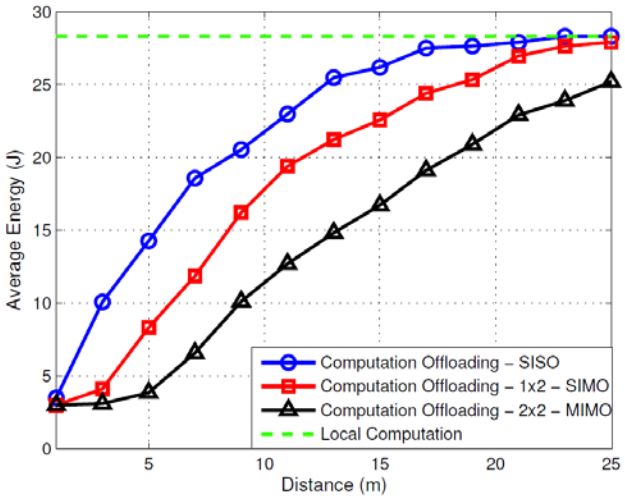


Figure 136. Average energy spent for processing versus distance (from MUE to SCeNBce), for different communication strategies.

Thus, assuming the presence of M statistically independent spatial channels, the pdf of the coefficient α is given by

$$p_A(\alpha) = \frac{1}{(M-1)!\,\bar{\alpha}^M}\,\alpha^{M-1}e^{-\alpha/\bar{\alpha}}u(\alpha) \tag{126}$$

where $\bar{\alpha}$ is the variance over the single channel coefficient and $u(\alpha)$ denotes the unitary step function. In particular, in Figure 136, we assume the use of SISO (i.e., $M=1$), 1x2 SIMO (i.e., $M=2$), and 2x2 MIMO (i.e., $M=4$) communication strategies. The BER is chosen equal to 10^{-3} , the path-loss coefficient β is equal to 2, the noise power is given by $N_0 = 5\times10^{-5}$, and the variance $\bar{\alpha}=1$.

Further, we consider the Graph 1 in Figure 135, as a call graph for this example. As expected, from Figure 136, we notice how computation offloading is more convenient if the distance between MUE and SCeNBce is sufficiently low, in order to allow the offloading of the entire program with high probability. This is a further numerical justification for favoring the access to the cloud through small cells. Further, it is possible to see how, by using MIMO strategies, which increase the probability to have a large normalized channel coefficient, we get a larger energy saving with respect to the SISO strategy, thanks to the increased offloading of instructions to the SCeNBce.

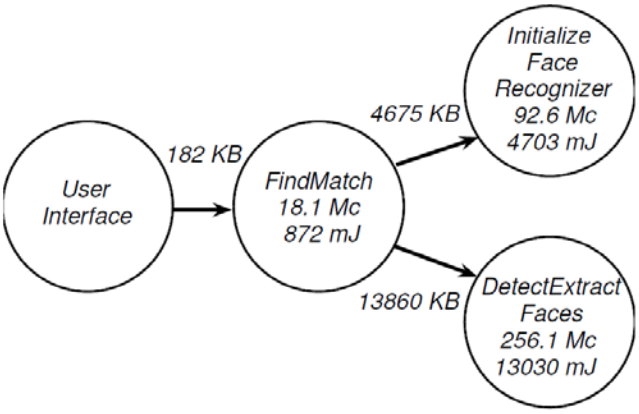


Figure 137. Example of call graph of a face recognition application from [Maui10].

5.2.2.1.4.2 *Numerical Example 2-Application to a face recognition program*

We now apply our proposed joint optimization approach to the case of a realistic call graph of a program, representing a face recognition application [Maui10]. The application's call graph is shown in Figure 137 . The call graph has a tree structure, where the “user interface” procedure is the root node, and all the links are unidirectional such that the direction is from left to right. The “user interface” procedure cannot of course be offloaded and must necessarily be evaluated locally. All the other methods can be offloaded. For this application, we illustrate an example of optimal power allocation and call graph's partition, in the case the MUE is transmitting by using multiple channels. For this purpose, in Figure 138, we report the result of our joint optimization algorithm, which is applied to the call graph of the face recognition application in Figure 137.

We consider an OFDM system with $K = 8$ subcarriers. For the problem in [P.3], the latency constraint L is chosen equal to the time needed by the MUE to compute the entire program locally, and the power budget constraint is $P_T = 0.02$ Watt. The local CPU clock speeds f_l and f_s , and the bit duration T_b are chosen as in previous simulations. The BER is chosen equal to 10^{-3} , the path-loss coefficient β is equal to 2, the noise power is given by $N_0 = 5 \times 10^{-5}$. The fading over the single carrier is given by the output of a random variable with pdf given by (126) with $M = 1$ and variance $\overline{\alpha} = 1$. The resulting normalized wireless channels between MUE and SCeNBce are shown in Figure 138 (middle).

The optimal graph's partition for this parameter setting is shown in Figure 138 (bottom), where, again, the white nodes denote procedures computed at the SCeNBce side. As we can notice from Figure 138 (bottom), all the remoteable nodes are offloaded to the SCeNBce. This means that the optimization has found that the most convenient solution in terms of energy is to transmit the 182 KB of the program state between the “user interface” and the “FindMatch” procedures, and then compute all the rest of the program at the server. Thus, only one link in the call graph is selected to be used for the transmission of data. In particular, the top plot in Figure 138 (top) shows the optimal power allocation over the multiple channels, achieved as a result of the optimization problem [P.4]. As we can see from Figure 138 (top and middle), the power allocation shows a water-filling behavior, where all the power is concentrated over the best channels, while no bits are transmitted over the worse channels. The energy saving is potentially huge in this case. Indeed, by computing locally nodes 2, 3, and 4 of the graph in Figure 137, the MUE would have spent 18.6 Joule, while, by offloading the entire program as in Figure 138 (bottom), the MUE would spend only 25 mJoule. The gain in terms of energy saving is about 740 times. Since the size of the program states on the other links is much bigger than the one on the first link, it is clear that, for this application, there are only two possible optimal graph's configurations: offload all the remoteable procedures or compute all the program locally. Indeed, if the normalized channels in Figure 138 (middle) were uniformly worse, the optimization problem [P.4] might have been unfeasible, thus leading to the impossibility to offload the computation to the server.

On the other side, even improving the channels, there is no possibility to improve the energy spent for processing by varying the graph's partitioning.

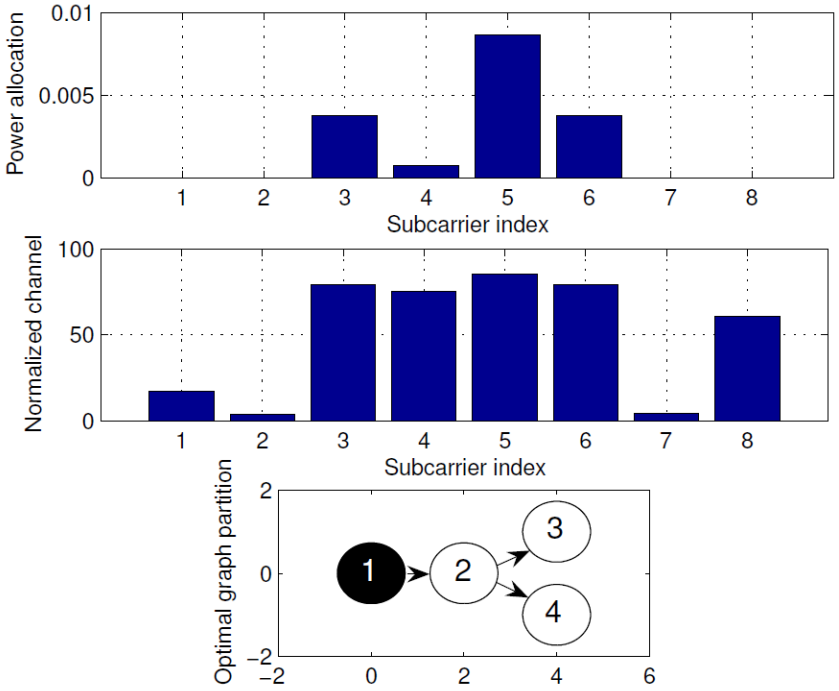


Figure 138. Result of the joint optimization over a wideband channel. (Top) Optimal power allocation. (Middle) Normalized channel coefficients. (Bottom) Optimal graph partition.

In summary in this section, we have proposed a method to optimize the allocation of communication resources and the call graph's partition of a computer program jointly, in a mobile cloud computing context, with the aim of minimizing the energy consumption at the mobile user, while satisfying a delay constraint imposed by the application.

The problem turns out to have a combinatorial complexity that increases with the granularity of the call graph structure. Nevertheless, we have proved that, for any given partition, the remaining optimization problem with respect to the radio parameters is convex and then it can be solved with numerically efficient methods. Furthermore, with respect to the integer variables, the problem is a binary linear programming problem, for which several efficient algorithms exist, with affordable complexity. In general, the granularity of the call graph will be selected as a tradeoff between computational optimality and complexity. Simulation results corroborate our theoretical findings and illustrate for what kind of application and channel conditions, computation offloading can provide a significant performance gain with respect to local computation.

5.2.3 **Task-oriented offloading**

In some practical cases, it may difficult to have the call graph of the application running on the mobile handset, or there might be concurrent applications running. In all these cases, rather than starting from the call graph of each application, it is more meaningful to model the computational load as a queue of instructions that need to be executed. The offloading decision has to do, in such a case, with the length of the queue and the application requirements, in terms of delay constraints, for example. In this case, the length of the computational queue is a state parameter that takes into account how the applications (virtually) running on each mobile handset are being served through time. Of course, the queue length changes with time, so that the offloading decision has to be made dynamically, as a function of application requirements, channel conditions and computational load at the cloud side. In the following section, we propose a dynamic approach to computation offloading in a multitask environment where the MUE has a queue of instructions to be executed, which can come from multiple applications running concurrently. A rule to deal with this dynamical nature of the problem,

the time axis is supposed to be split in slots and in each time slot we need to decide if it is advantageous to offload computations or not, depending on queue status, channel conditions, server state, delay constraints, and so on. This requires a cross-layer optimization strategy, which encompasses application, MAC and physical layer into a joint framework the details of which will be described in the incoming sections [BarbSardDiLor].

The offloading decision rule aims to minimize the energy consumption at the mobile handset: the user has to find out the minimum transmitted energy spent to offload computations in order to compare it with the local energy consumption. To save energy computation, offloading is convenient if the energy spent to transmit the program state to the intended server is less than the energy consumption to run the application locally. On the other hand the minimum energy spent to offload computation has to meet the constraint on the application-dependent delay since every application specifies a maximum latency that should not be exceeded, to provide a satisfactory quality of experience (QoE) to the user. Hence, the decision on whether offloading or not must take into account the latency constraint.
Additionally the optimization strategy has to guarantee the stability of the queue of instructions to be executed at the mobile handset.

Taking into account all these issues we propose a novel strategy to jointly optimize the radio and computational capabilities by considering the following scenarios:

- Single Small Cell Cloud Manager (SCM) serving a single cell
- Single SCM serving multiple cells
- Multiple-SCM serving multiple cells with congested or available backhaul

In the incoming sections the following offloading optimization strategies are proposed:

- Task-oriented partitioning offloading in the multi user case under power budget constraint and the constraint of guaranteeing both the stability of the queue of instructions to be executed at the mobile terminal and the application delay constraint.
- Joint optimization of server computational resources and mobile users transmit powers in the single cell case under delay and power budget constraints.
- Joint optimization of radio and computational resources in the multi-cell MIMO scenario under delay and power budget constraints.
- Optimal distributed allocation of communication/computation resources in the multiple SCM MIMO scenario under delay and power budget constraints.
- The offloading strategies proposed in this section encompass also the full-offloading strategy as in this case it is sufficient to assume that a single task (whole program) has to be executed.

5.2.3.1 *Task-oriented partitioning offloading with computational stability constraint*

In this section we propose an optimization strategy based on task-oriented partitioning offloading to find out the optimal power allocation which minimizes the energy expenditure at the mobile terminal, under the following constraints:
a) A power budget constraint at the mobile handset;
b) Delay constraint;
c) Computational stability constraint.
The third constraint takes into account that the net computing rate must include the time wasted to transfer the program execution from the mobile device to the server. The final net rate must exceed the rate required to run the application within the latency constraints needed to meet the user quality of experience.
To better explain the proposed strategy we consider first the single mobile case and afterwards we extend the approach to the multi-user case [Barbarossa14].

5.2.3.1.1 Single user case

Let us consider a time slotted system with the following parameters:

- f_{loc} is the local computational rate (instructions/sec);

- f_s is the server computational rate (instruction/sec);
- T is the duration of the time slot (it could also represent a latency constraint);
- B is the bandwidth allocated to transmission from MUE to SCeNBce;
- p_{proc} is the power spent for local processing;
- N is the dimension of the program state;
- λ is the computational rate (instructions/sec) pertaining to the application.

The parameter λ is a parameter that is negotiated between the MUE and the cloud.
An application can be offloaded only if the server is able to meet the computational needs required by the application.

In principle, the degrees of freedom are:

- $d \in \{0,1\}$ is the decision variable equal to 0 if the program is executed locally;
- $p \in [0, P_T]$ is the power spent for transmitting the program state from mobile to server;
- $\tau \in [0, T]$ is the duration of the interval used for transmitting the program state to the server (to enable the execution transfer).

Nevertheless, these variables can be reduced to one, exploiting the analytic expressions relating them. First of all, the duration τ is related to the transmit power p by

$$\tau = \frac{N}{B\log(1+\alpha p)} \Rightarrow p = \frac{1}{\alpha}\left(2^{N/B\tau} - 1\right) \tag{127}$$

where $\alpha := |h|2/(\Gamma(BER)N_0)$ is the (equivalent) channel coefficient, having denoted by h the channel coefficient, whereas $\Gamma(BER)$ is the SNR margin necessary to meet a desired target BER, and N_0 is the noise power. Furthermore, since offloading occurs only if the energy spent locally is greater than the energy required for offloading, the decision variable d is related to p and τ through

$$d = u\left(p_{proc}T - p\tau\right) \tag{128}$$

where $u(\cdot)$ is the unit step function.
Since the transmit power is upper bounded by a budget constraint P_T, the duration τ is bounded as follows

$$\frac{N}{B\log(1+\alpha P_T)} \le \tau \le T. \tag{129}$$

Whenever the computation is offloaded, an inevitable waste of time occurs to transfer the program status from the mobile device to the server. This "communication" time translates into a computation rate loss: If within an overall time interval T, τ seconds are dedicated to data transfer, the real computing rate at the server is $f_S\left(1-\frac{\tau}{T}\right)$. The computational stability constraint can be formulated as follows

$$f_R := f_{loc} + \left[f_s\left(1-\frac{\tau}{T}\right) - f_{loc} \right] u\left(p_{proc}T - p\tau\right) \ge \lambda \tag{130}$$

Exploiting the relations among the free variables, we can express the objective function in terms of the single variable τ, as

$$\mathcal{E}_s := \min\left[p_{proc}T, \frac{\tau}{\alpha}\left(2^{N/B\tau}-1\right)\right] := p_{proc}T + \left[\frac{\tau}{\alpha}\left(2^{N/B\tau}-1\right) - p_{proc}T\right]\cdot u\left(p_{proc}T - \frac{\tau}{\alpha}\left(2^{N/B\tau}-1\right)\right) \tag{131}$$

The optimization problem can then be formulated as follows

$$\min_{\tau} \min\left[p_{proc}T, \frac{\tau}{\alpha}\left(2^{N/B\tau}-1\right)\right] \quad \text{[P.5]} \tag{132}$$

subject to

$$\begin{aligned} &\textbf{C.1} \qquad \frac{N}{B\log\left(1+\alpha P_T\right)} \le \tau \le T \\ &\textbf{C.2} \qquad f_{loc} + \left[f_S\left(1-\frac{\tau}{T}\right) - f_{loc}\right] u\left(p_{proc}T - p\tau\right) \ge \lambda. \end{aligned} \tag{133}$$

After some algebraic manipulations, it is possible to show that this problem is feasible if $f_S \ge \lambda$ and the (equivalent) channel coefficient exceeds a minimum value, i.e.,

$$\alpha > \alpha_{\min} \tag{134}$$

where

$$\alpha_{\min} = \max\left[\frac{1}{P_T}, \frac{\left(1-\frac{\lambda}{f_S}\right)}{p_{proc}}\right]\left(2^{\frac{N}{BT(1-\lambda/f_S)}}-1\right) \tag{135}$$

This last condition states, in closed form, that offloading takes place only if the channel is sufficiently good, as expected. The interesting point is that the minimum channel value is dictated by parameters related to both radio and computational parameters. If the feasible set is non-empty, the optimal solution of problem [P.5] is given in closed form by:

$$p = \frac{1}{\alpha}\left(2^{\frac{N}{BT(1-\lambda/f_S)}}-1\right) \tag{136}$$

Then the following offloading algorithm can be performed.

Single user minimum energy offloading strategy
If $\alpha > \alpha_m$ and $f_S > \lambda$ then *enable computation offloading* with optimal power in (136) else *run the applications locally* by satisfying the stability condition **C.2** if $f_{loc} > \lambda$.

As a numerical example we plot in Figure 139 the optimal energy spent by the mobile user (upper subplot) corresponding to the time-varying channel profile drawn in the lower subplot. We have assumed $T = 5$ msec, $\lambda = 10^4$ cycles/s, $f_S = 10^7$ cycles/s, $p_{proc} = 1$ mWatt, $P_T = 10$ mWatt, $N = 850$ Mbyte, $B = 200$ MHz.

It can be observed that as the channel coefficient is sufficiently high so that condition (134) holds true then the mobile user performs the offloading of computations otherwise to save energy consumption it runs computations locally.

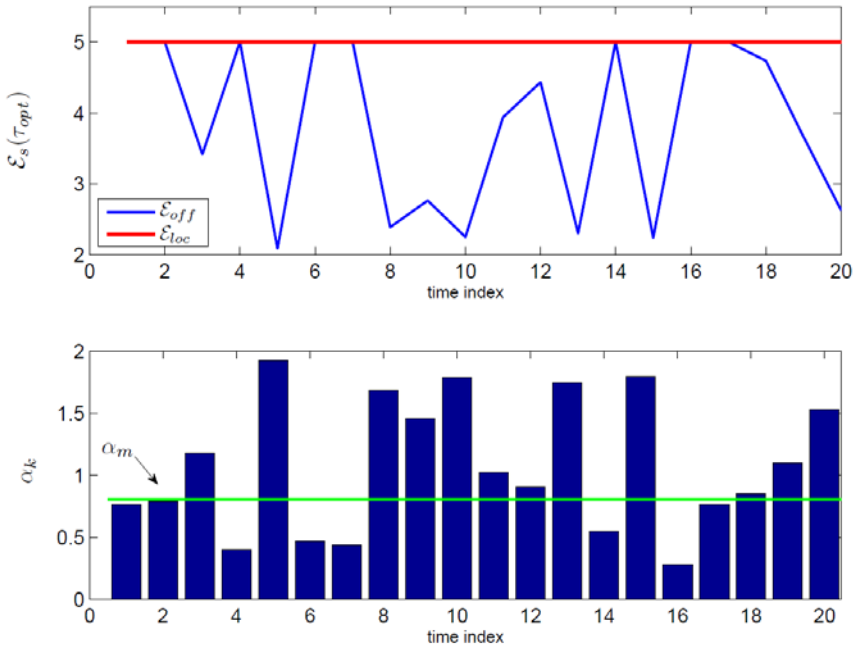


Figure 139. Optimal energy expenditure versus time (top subplot) assuming the channel profile in the bottom subplot.

In practice, since the wireless channel is random, there is a nonnull probability that offloading takes place or not, depending on both channel status and computational requests. The consequence is that the real computing rate experienced over a fading channel is going to depend on the channel statistics and typically will be lower than the rate achievable under ideal channel conditions. However, building on previous expressions, we can derive the equivalent computing rate λ^*, to be asked by the mobile device in order to ensure the desired rate λ, in the presence of fading, provided that the fading statistics are known. More specifically, the probability of offloading is simply:

$$\mathcal{P} := \text{Prob}\{\alpha > \alpha_{\min}\} = 1 - D_A(\alpha_{\min}) \tag{137}$$

where $D_A(\alpha)$ denotes the cumulative distribution function of α and $\alpha_{\min}$ is given in (135), with λ^* instead of λ. The expected value of the rate is then

$$f_{ave} = \left(1 - \mathcal{P}\left(\lambda^*\right)\right) f_{loc} + \mathcal{P}\left(\lambda^*\right)\lambda^* \tag{138}$$

where we made explicit the dependence of $\mathcal{P}$ from λ^* (through $\alpha_{\min}$). Imposing this average rate to be equal to the target rate λ, we end up with a nonlinear equation in λ^*. This equation admits a unique solution. As a numerical example, Figure 140 (top) shows λ^* as a function of the distance between mobile user and base station, for different antenna configurations. The simulation parameters are: f_s = 1010, f_{loc} = 107, p_{proc} = 0.1, P_T = 0.1, N = 5× 103, B = 106, T = 10−2, λ = 108. As expected, at short distances, λ^* tends to coincide with λ, because the channel attenuation is negligible and offloading occurs most of the times. However, as the distance exceeds a certain value, λ^* starts increasing considerably to compensate for the missing opportunities to offload. In parallel, Figure 140 (bottom) shows the average energy spent for offloading as a function of the distance between mobile user and base station, for different antenna configurations. As expected, the energy increases at larger distance until reaching a constant value dictated by the energy required to run the application locally. It is interesting to see how MIMO transceivers yield a larger saving and then, ultimately, widen the area over which offloading is beneficial.

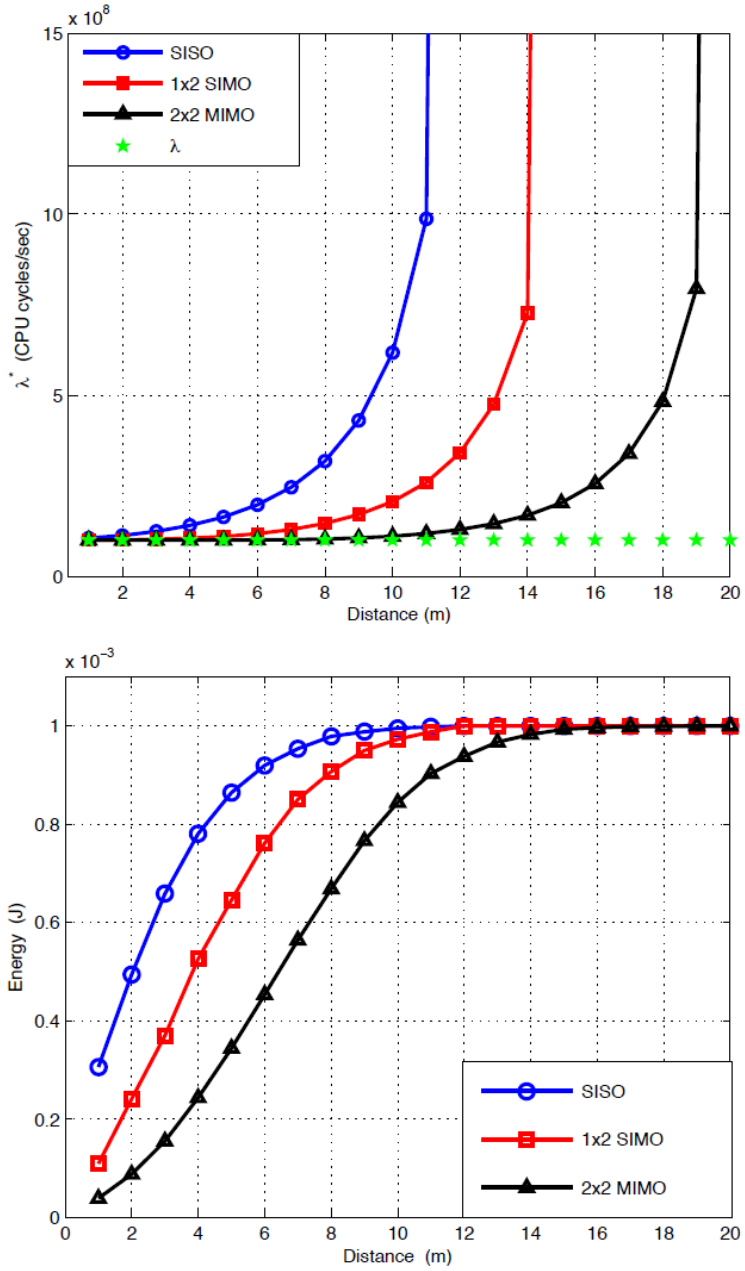


Figure 140. (Top) Equivalent rate λ^* and (bottom) Average Energy spent for offloading, vs. distance between mobile user and access point, for different communication strategies.

5.2.3.1.2 Multi-user case

In the multiuser case, the optimization can be cast as the minimization of the sum of the energies spent from the mobiles to run their applications under the stability conditions and the maximum server capacity as follows

$$\min_{\boldsymbol{\tau},\mathbf{f}} \sum_{k=1}^{K} \min\left[\left(p_{proc}^{k}\, T, \frac{\tau_k}{\alpha_k}\left(2^{N_k/B\tau_k}-1\right)\right)\right] \quad [\mathbf{P.6}] \tag{139}$$

subject to

$$\textbf{C.3} \quad \frac{N_k}{B\log\left(1+\alpha_k P_T\right)} \le \tau_k \le T, k=1,\ldots,K$$

$$\textbf{C.4} \quad f_{loc}^k + \left[f_k\left(1-\frac{\tau_k}{T}\right) - f_{loc}^k \right] \cdot u\left(p_{proc}^k T - \frac{\tau_k}{\alpha_k}\left(2^{N_k/B\tau_k}-1\right)\right) \ge \lambda_k \tag{140}$$

$$\textbf{C.5} \quad \sum_{k=1}^{K} f_k \le f_S$$

where $\boldsymbol{\tau} = \left[\tau_1, \ldots, \tau_K\right]$ and $\boldsymbol{f} = \left[f_1, \ldots, f_K\right]$. Let us first check the problem feasibility, assuming $f_S > \lambda_k \quad \forall k$.
Denoting with $\tau_k^* = \tau^*(\alpha_k)$ the unique solution of the following equation associated to user k

$$p_{proc}^k T = \frac{\tau_k^*}{\alpha_k}\left(2^{N_k/(B\tau_k^*)}-1\right). \tag{141}$$

user k is enabled to offload if the following necessary conditions are both verified

$$\begin{aligned} &\textbf{F.3} \quad \frac{N_k}{B\log\left(1+\alpha_k P_T\right)} \le T \\ &\textbf{F.4} \quad \tau^*(\alpha_k) \le T(1-\lambda_k / f_S) \end{aligned} \tag{142}$$

After some algebra, conditions **F.3** and **F.4** can be reduced to the following necessary single condition on the channel coefficients:

$$\textbf{F.5} \quad \alpha_k > \alpha_m^k \tag{143}$$

with

$$\alpha_m^k = \max\left[\frac{\left(2^{\frac{N_k}{BT}}-1\right)}{P_T}, \frac{\left(1-\frac{\lambda_k}{f_S}\right)\cdot\left(2^{\frac{N_k}{BT(1-\lambda_k/f_S)}}-1\right)}{p_{proc}^k} \right]. \tag{144}$$

In case condition **F.5** is not verified user k performs its computations locally . Let us denote with $\mathcal{S}$ the set of users satisfying **F.5** and define $|\mathcal{S}|$ as the cardinality of $\mathcal{S}$. From condition **C.4** , in case of offloading we get $f_k \ge f_{k,min}$ with $f_{k,min} = \lambda_k / \left[1 - \frac{N_k}{BT\log\left(1+\alpha_k P_T\right)}\right]$. Then, to ensure that the feasibility set is non-empty, the further global necessary condition must be verified by all users in $\mathcal{S}$:

$$\textbf{F.6} \quad \sum_{k\in\mathcal{S}} f_{k,min} < f_S. \tag{145}$$

Hence if conditions **F.5** and **F.6** are verified for a given subset $\overline{\mathcal{S}} \subseteq \mathcal{S}$ then the feasible set in (140) is non-empty and problem **[P.6]** can be formulated as

$$\min_{\boldsymbol{\tau}_{\bar{S}},\mathbf{f}_{\bar{S}}} \sum_{k\notin\bar{\mathcal{S}}} \mathcal{E}_{loc}^{k} + \sum_{k\in\bar{\mathcal{S}}} \frac{\tau_k}{\alpha_k}\left(2^{N_k/(B\tau_k)}-1\right) \quad [\mathbf{P}.7]$$

$$s.t. \quad \frac{N_k}{B\log\left(1+\alpha_k P_T\right)} \le \tau_k \le T, \forall k \in \bar{\mathcal{S}}$$

$$f_k\left(1-\frac{\tau_k}{T}\right) \ge \lambda_k, \forall k \in \bar{\mathcal{S}} \tag{146}$$

$$f_{loc}^{k} \ge \lambda_k, \forall k \notin \bar{\mathcal{S}}$$

$$\sum_{k\in\bar{\mathcal{S}}} f_k \le f_S$$

where denoting with $|\bar{\mathcal{S}}|$ the cardinality of the set $\bar{\mathcal{S}}$ we have $\tau_{\bar{S}}, f_{\bar{S}} \in \mathrm{R}_{+}^{|\bar{S}|}$ and $\mathcal{E}_{loc}^{k} = p_{proc}^{k} T$.

Let us denote by $\mathcal{F}$ the collection of all, let us say L, subsets $\bar{\mathcal{S}}_i$ of $\mathcal{S}$ satisfying conditions **F.5** and **F.6** for $i = 1,\ldots,L$ with $L \le 2^{|\mathcal{S}|}-1$.

Our goal is to find the optimal users subset $\bar{\mathcal{S}}^{*} \subseteq \mathcal{F}$ and the optimal vectors $\tau_{\bar{S}}^{*}, f_{\bar{S}}^{*}, \in \mathrm{R}_{+}^{|\bar{S}^{*}|}$ which solve problem **[P.6]** under the constraints in (140). To this end we perform an exhaustive search on all the possible subsets of $\mathcal{F}$ by solving for each of them the optimization problem **[P.7]**. Assuming that the inequalities $f_{loc}^{k} \ge \lambda_k$ are verified $\forall k \notin \bar{\mathcal{S}}$, **[P.7]** can be rewritten for each $\bar{\mathcal{S}} \subseteq \mathcal{F}$ as

$$\min_{\boldsymbol{\tau}_{\bar{S}},\mathbf{f}_{\bar{S}}} \sum_{k\in\bar{\mathcal{S}}} \frac{\tau_k}{\alpha_k}\left(2^{N_k/(B\tau_k)}-1\right) \quad [\mathbf{P}.8]$$

$$s.t. \quad \tau_{k,min} \le \tau_k \le T, \forall k \in \bar{\mathcal{S}}$$

$$f_k\left(1-\frac{\tau_k}{T}\right) \ge \lambda_k, \forall k \in \bar{\mathcal{S}} \tag{147}$$

$$\sum_{k\in\bar{\mathcal{S}}} f_k \le f_S$$

with $\tau_{k,min} = \dfrac{N_k}{B\log\left(1+\alpha_k P_T\right)}$. It is easy to see that problem **[P.8]** is convex. First, the objective function is sum of convex functions of τ_S. Then the second constraint is a log-concave function of (τ_k, f_k) for each k , thus guaranteing that all its superlevel sets $\left\{\left(\tau_k, f_k\right) \in \mathrm{R}_{+}^{2} \middle| f_k\left(1-\frac{\tau_k}{T}\right) \ge \lambda_k\right\}$ are convex.

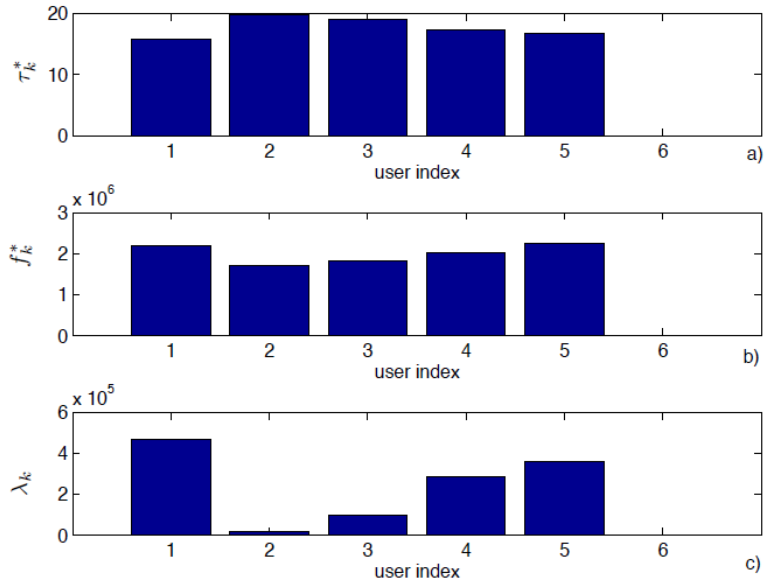


Figure 141. a) Optimal transmission vs. user index; b) optimal per user computational rate; c) rate requirement for each application.

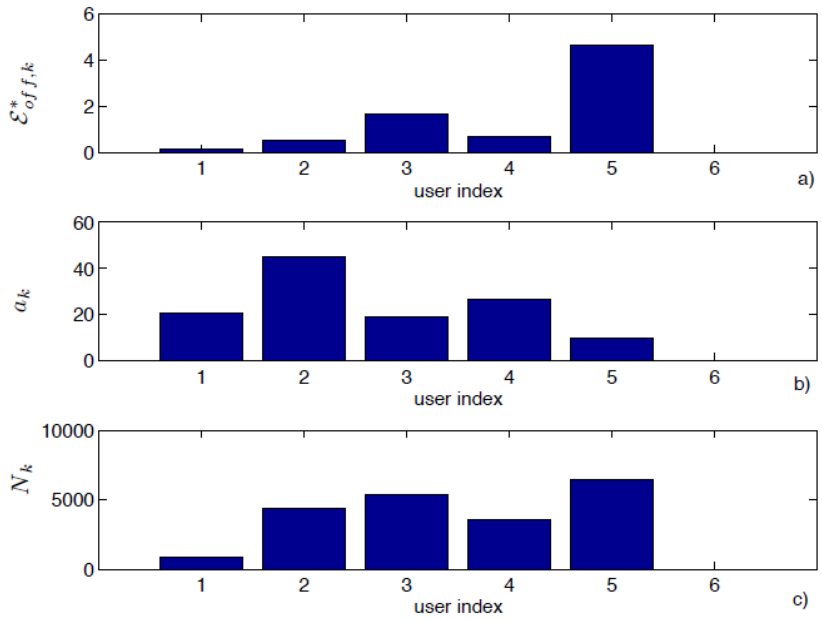


Figure 142. a) Optimal per user energy consumption corresponding to the channel profile in b) and to the program states in c).

Thus, it follows that the set in (147) is feasible, when conditions **F.5** and **F.6** hold true, and it is also convex as non-empty intersections of convex sets. In summary, the multiuser offloading strategy is reported in the following Algorithm.

Multi-user minimum energy offloading strategy
1. Find $\mathcal{F}$, i.e. the collection of all the subsets $\overline{\mathcal{S}}_i$ satisfying **F.5** , **F.6** ; 2. if $\mathcal{F}=\varnothing$ then all the users run computations locally else for each subset $\overline{\mathcal{S}}_i$ o compute the optimal pair $(\boldsymbol{\tau}^*_{\overline{S}_i}, \mathbf{f}^*_{\overline{S}_i})$ by solving [**P.8**] o evaluate the optimal energy spent as

$$\mathcal{E}_s(\overline{\mathcal{S}}_i, \boldsymbol{\tau}^*_{\overline{\mathcal{S}}_i}, \mathbf{f}^*_{\overline{\mathcal{S}}_i}) = \sum_{k \notin \overline{\mathcal{S}}_i} \mathcal{E}^k_{loc} + \sum_{k \in \overline{\mathcal{S}}_i} \frac{\tau^*_k}{\alpha_k}\left(2^{N_k/(B\tau^*_k)} - 1\right) \quad (148)$$

3. Find the optimal subset $\overline{\mathcal{S}}^*$ and the optimal pair $(\boldsymbol{\tau}^*_{\overline{\mathcal{S}}_i}, \mathbf{f}^*_{\overline{\mathcal{S}}_i})$ such that

$$\mathcal{E}_s(\overline{\mathcal{S}}^*, \boldsymbol{\tau}^*_{\overline{\mathcal{S}}_i}, \mathbf{f}^*_{\overline{\mathcal{S}}_i}) = \arg\min_{\{\overline{\mathcal{S}}_i\}_{i=1}^L} \mathcal{E}_s(\overline{\mathcal{S}}_i) \quad (149)$$

Numerical Example - Optimal Allocation of communication and computation resources: Some numerical examples are useful to assess the performance of the proposed algorithm. In our numerical simulations we set $p^k_{proc} = 10$ mW, $\forall k$, $T = 20$ msec, $f_S = 10^7$ cycles/s, $P_T = 10$ mWatt, $B = 200$ MHz. Figure 141 shows the effect of the stability constraint on the optimization strategy. It can be noted that, when λ_k is large, the optimal computational rate f^*_k tends to be large as well, while the optimal transmission delay τ^*_k decreases. This behavior can be explained by observing that the stability constraint is always active. Moreover note that user 6 is excluded from computation offloading since its channel coefficient does not meet the conditions F.5, F.6. Finally Figure 142 (a) shows the optimal offloading energy per user $\mathcal{E}^*_{off}$, associated to the optimal variables shown in Figure 141. The corresponding channel coefficient a_k and program states N_k are reported in subplots (b) and (c), respectively. From Figure 142, we can observe the joint effect on the optimal energy consumption of both channel conditions and class of running application. In particular note that user 5 entails the maximum energy consumption since it must transmit a large program state, with a stringent stability constraint, over a wireless channel inducing a strong fading.

5.2.3.1.3 Conclusions

We have proposed a computation offloading framework for minimizing the energy consumption at the mobile handset, incorporating a computational stability constraint dictated by the application. The approach has been also extended to the multiuser case, where a single SCceNB is serving multiple mobile handsets. The proposed strategy solves, jointly, an admission control (evaluating the set of users admitted to offloading), and a combined radio and computing resource allocation for each mobile user. The resource allocation is dynamic, depending on channel conditions. If some users are not admitted, the computation resources are distributed among the admitted ones. Simulation results show how MIMO transceivers contribute to augmenting the computation capabilities of each user. Furthermore, we have also shown how the application features, in terms of size of program state and latency constraints, play a key role in the offloading decisions. In summary, an effective offloading mechanism can be setup only by taking into account, jointly, communication and computation aspects, together with user quality of experience.

5.2.3.2 ***Joint optimization of computational/communication resources in multi-user case***

5.2.3.2.1 Single cell case SISO scenario

In this section we consider an alternative task-oriented partitioning offloading aiming to find the optimal radio and computational resource allocation under delay constraint [Barb-Sard-PDL13]. We consider a multiuser scenario where a set of K mobile users are served by one SCeNBce. To avoid unnecessary interference, the SCeNBce assigns orthogonal channels to the Mobile User Equipments (MUE). The SCeNBce is capable of running f_S instructions per second. Thanks to its multi-tasking capability, it is possible to serve the K users concurrently, by allocating a number f_k of instructions per second to each user, under the constraint that $\sum_{k=1}^{K} f_k \leq f_S$. Of course, to offload the computations

to the remote server, the MUE has to send all the information to the server. Under a very general setting, we assume that the program to be executed is structured as a graph composed by computational components, each one characterized by the following parameters: a) the number of input bits per second; b) the set of instructions to be run to perform the computations; c) the number of output bits per second. We assume that the instructions to be executed are available at the server or, if not, they can be downloaded by the server through a high speed wired link. We consider a batch processing and we denote by N_k the number of input bits and by $w_k = w_k(N_k)$ the number of instructions to be executed, for that given set of N_k input bits.

The MUE has the possibility to offload the computations or not, depending on several factors, such as the number of instructions to be executed, the conditions of the wireless channel between MUE and SCeNBce, and the availability of the server, which typically runs several applications concurrently. More specifically, every application specifies a maximum latency that should not be exceeded, to provide a satisfactory quality of experience (QoE) to the user. Hence, the decision on whether offloading or not must take into account the latency constraint. In case of computations performed locally, at the mobile side, the latency is simply the ratio between the number of instructions to be executed and the number f_k^{loc} of instructions per second that can be run locally. In case of offloading, the analysis is more complicated as the latency must incorporate the time to transmit the input bits to the server, with no decoding errors, the time necessary for the server to execute the instructions and the time to send the result back to the MUE. More specifically, the minimum time T_{tk} necessary to transmit N_k bits of duration T_b over an additive white Gaussian noise (AWGN) channel is

$$T_{tk} = \frac{N_k T_b}{\log_2\left(1+\frac{p_{tk}\,|H_k|^2}{\Gamma(\mathrm{BER}_k) d_k^{\beta} N_0}\right)}, \tag{150}$$

where p_{tk} is the transmit power of the k -th MUE, d_k is the distance between the k -th MUE and the SCeNBce, β is the path loss exponent, N_0 denotes the noise power and H_k is the channel fading coefficient (normalized to distance); $\Gamma(\mathrm{BER}_k) = -(2/3)\log(5\mathrm{BER}_k)$ represents the so called SNR margin (whenever the base is unspecified, $\log(x)$ has to be intended as the natural base logarithm), introduced to meet the desired target BER_k of the k -th user with a QAM constellation. In practice, the time necessary to let the server receive the correct input bits is often greater than (150) because of transmission errors that require the retransmission of the erroneous packets. If we denote by S_k the random variable indicating the number of retransmissions over the k -th link and with P_{ek} the corresponding packet error rate, the average number of retransmissions, assuming independent errors on each packet, is $\bar{S}_k = \frac{1}{1-P_{ek}}$.

The relation between BER_k and P_{ek} is dictated by the channel coding scheme adopted in the link between the k -th user and the SCeNBce.
The average delay associated to the correct reception of N_k bits at the server side, incorporating packet duration and retransmissions, is then

$$\Delta_{tk} = \bar{S}_k T_{tk} = \frac{N_k T_b}{(1-P_{ek})\log_2\left(1+\frac{p_{tk}\,|H_k|^2}{\Gamma(\mathrm{BER}_k) d_k^{\beta} N_0}\right)}. \tag{151}$$

Expression (151) shows that offloading may be advantageous if the distance d_k between MUE and SCeNBce is sufficiently small, so that the transmission rate is high. This is a further justification for favoring the access to the cloud through small cells.

The overall (average) latency associated to computation offloading from the k -th MUE is then

$$L'_k = \Delta_{tk} + \frac{w_k}{f_k} + T_{rk} \tag{152}$$

where w_k / f_k is the time necessary to run w_k instructions at the server side and T_{rk} denotes the time necessary for the server to send the results back to the MUE. This last time does not depend on parameters associated to the MUE and we incorporate it in the latency limit by introducing, for the sake of simplicity of notation, the variable $L_k := L'_k - T_{rk}$. Denoting by $\mathbf{p} := (p_1, \dots p_K)$ and $\mathbf{f}_S := (f_1, \dots f_K)$ the transmit powers and the computation percentages associated to the MUE's, the optimization problem can then be cast as follows:

$$\begin{aligned} \min_{\mathbf{p},\mathbf{f}_S} \quad & \sum_{k=1}^{K} p_k, \qquad\qquad [\mathbf{P}.\mathbf{9}] \\ \text{s.t.} \quad & \frac{c_k}{\log(1+p_k a_k)} + \frac{w_k}{f_k} \le L_k, \forall\, k = 1, \dots, K \\ & 0 < p_k \le P_T, \quad f_k > 0, \quad \forall\, k = 1, \dots, K \\ & \sum_{k=1}^{K} f_k \le f_S \end{aligned} \tag{153}$$

where $c_k = \dfrac{N_k T_b \log 2}{(1-P_{ek})}$ and $a_k = \dfrac{|H_k|^2}{\Gamma(\mathrm{BER_k}) d_k^{\beta} N_0}$ .Let us define the maximum rates over each channel $R_k^{max} := \log(1 + a_k P_T)$. The main result of this work is the following.

Theorem 3. *If*

$$R_k^{max} > c_k / L_k, \quad k = 1, \dots, K, \tag{154}$$

and

$$\sum_{k=1}^{K} \frac{w_k}{L_k - c_k / R_k^{max}} \le f_S, \tag{155}$$

then problem $[\mathbf{P}.\mathbf{9}]$ *admits a non-empty feasible set satisfying all the constraints in (153) and the problem is convex.*

Proof. See [Barb-Sard-PDL13].

Since Slater's constraints qualifications hold true and having proved the convexity of problem $[\mathbf{P}.\mathbf{9}]$, we may impose the KKT conditions to find out important relations among the variables. The Lagrangian associated to (153) is

$$\mathcal{L}(\mathbf{p},\mathbf{f}_S)=\sum_{k=1}^{K}p_k+\sum_{k=1}^{K}\gamma_k\left(\frac{c_k}{\log(1+a_k p_k)}+\frac{w_k}{f_k}-L_k\right)-\sum_{k=1}^{K}\nu_k p_k-\sum_{k=1}^{K}\tau_k f_k+\sum_{k=1}^{K}\theta_k(p_k-P_T)+\mu\left(\sum_{k=1}^{K}f_k-f_S\right) \quad (156)$$

where the variables $\gamma_k,\nu_k,\tau_k,\theta_k,\mu$ are all nonnegative coefficients representing the Lagrange multipliers. The KKT conditions are, for $n=1,\ldots,K$,

$$\frac{\partial\mathcal{L}}{\partial p_n}=1-\frac{\gamma_n a_n c_n}{(1+a_n p_n)\log^2(1+a_n p_n)}-\nu_n+\theta_n=0 \quad (157)$$

$$\frac{\partial\mathcal{L}}{\partial f_n}=-\frac{\gamma_n w_n}{f_n^2}+\mu-\tau_n=0 \quad (158)$$

$$\nu_n p_n=0 \quad (159)$$

$$\tau_n f_n=0 \quad (160)$$

$$\gamma_n\left(\frac{c_n}{\log(1+p_n a_n)}+\frac{w_n}{f_n}-L_n\right)=0 \quad (161)$$

$$\theta_n(p_n-P_T)=0 \quad (162)$$

$$\mu\left(\sum_{n=1}^{K}f_n-f_S\right)=0. \quad (163)$$

We can notice that, if the MUE n is served, i.e. $f_n>0$ and $p_n>0$, then $\tau_n=0$ and $\nu_n=0$. Then, from (158), it follows that μ must be strictly positive. As a consequence, all the γ_n are strictly positive. In fact, if μ were equal to zero, from (158) we would have $\gamma_n=0$, $\forall n$ and then, from (157), $1+\theta_n=0$, which is impossible since $\theta_n\geq 0$. The fact that $\mu\neq 0$, implies, from (162) that all the computational capability is fully used, i.e. $\sum_{n=1}^{K}f_n=f_S$. This should be expected, because it is clear that any choice such that $\sum_{n=1}^{K}f_n<f_S$ would be sub-optimal, as it gives the possibility to increase at least one value of f_n to obtain a lower p_n . At the same time, the fact that $\gamma_n>0$, $\forall n$, implies that the latency constraint is always active, i.e.

$$\frac{c_n}{\log(1+p_n a_n)}+\frac{w_n}{f_n}=L_n. \quad (164)$$

This equation establishes a one-to-one relation between the transmit power p_n of MUE n and the number of CPU cycles/sec f_n that the server assigns to the same MUE. This relation establishes a strict, yet important, link between the communication and computation resources. Since wireless channels are typically affected by fading, the validity of conditions (154) depends on the value assumed by the channel coefficient a_k present in the definition of R_k^{max} . Modeling the coefficients a_k as statistically independent random variables, we can modify the previous theorem in order to incorporate an admission strategy. In general, it may happen that the conditions (154) are satisfied only for a subset of users.

Let us denote by I the set of user indices such that (154) is true. Clearly, all users whose indices do not belong to I are not admitted to computation offloading. For all other users, the global condition becomes

$$\sum_{k\in I}\frac{w_k}{L_k-c_k/R_k^{max}}\leq f_S. \quad (165)$$

In practice, the set of admissible users is going to change with time, as the channel coefficients assume different values. As a consequence, the admissibility check has to be carried out dynamically. The time axis is divided in blocks and the decision whether to offload computations is performed over each block: For every block, a new set of admissible users is defined and the communication/computation resources are allocated according to the channel values and the computational needs of each user.

As an example, in Figure 143, we illustrate the joint optimization of communication (transmit power, first subfigure) and computation resources (number of CPU cycles, second subfigure) in the case of $K = 4$ MUE's served by a single SCeNBce, for specific values of number of bits N_k (third subfigure) and equivalent channels a_k (fourth subfigure). For the sake of simplicity, the number of instructions to be executed has been simply related to the N_k input bits through a linear function, i.e., $w_k = \alpha \cdot N_k$, with $\alpha = 40$. The other parameters are $f_S = 10^7$, $\mathrm{BER} = 10^{-3}$, $T_b = 1/(64 \cdot 10^3)$, $L_k = 2.3 \cdot 10^{-3}$, $\forall k$, and $P_T = 1$. As we can notice from Figure 143, the optimization assigns, as expected, a larger amount of power and CPU cycles to those MUE's having a larger number of input bits N_k.

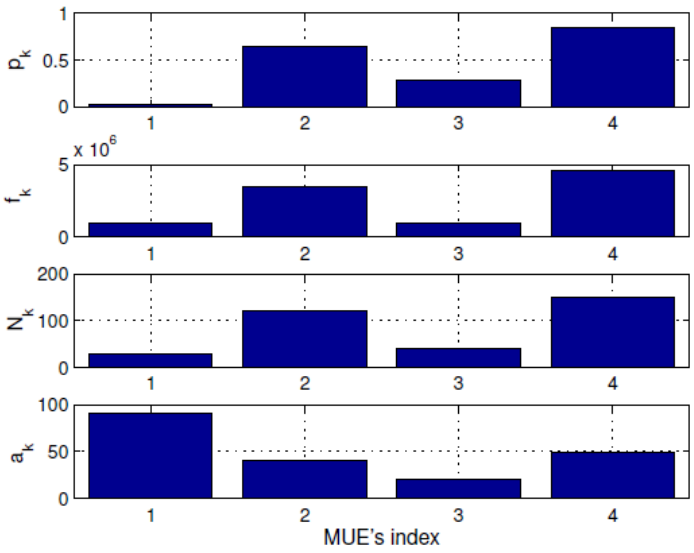


Figure 143. Optimal transmitted power and CPU cycles, for specific values of number of bits N_k and channels a_k.

5.2.3.2.1.1 *Computation scheduling*

We address now more specifically the problem of dynamically scheduling the instructions to be executed either locally or remotely, with the aim of augmenting the capability of the mobile handset to run applications requiring a serving time smaller than that available on the mobile handset.

As mentioned above, the decision is taken in each time slot of Δ seconds. We denote by $A_k(n)$ the number of new instructions to be executed accumulated in an interval of duration Δ seconds starting at time $t = n\Delta$. We model $A_k(n)$ as a random process and we assume $\mathbb{E}[A_k(n)] = \lambda_k$ and $\mathbb{E}[A_k^2(n)] < \infty$. Then we define $Q_k(n)$ as the queue of instructions to be executed at time slot n; $Q_k(n)$ is a state variable that takes into account how user k has been served up to slot n. For each slot n, every user verifies if its channel coefficient $a_k(n)$ satisfies the individual feasibility condition

$$a_k(n) > \frac{e^{c_k/L_k} - 1}{P_T}. \tag{166}$$

All users satisfying this equation notify the SCceNB consequently. The FCM gets this information about the set I of admissible users and checks if $\sum_{k\in I}\frac{w_k}{L_k - c_k / R_k^{max}} \le f_S$ is also satisfied. If yes, the FCM solves the optimization problem and finds out the optimal $p_k(n)$ and $f_k(n)$ to be associated with the admitted users, at time slot n. If it is not satisfied, the FCM starts removing from the set I the users having the shortest queue of instructions, in order to privilege the ones having more load. The check is repeated until finding the set of users satisfying all the feasibility conditions. In principle, in some slot, this set could be empty, if all channels are strongly attenuated. However, the probability of this event is rather low.
In any case, all the users that are not admitted for offloading run their instructions locally. Our goal now is to investigate the stability of the queue of computations accumulating on every device. More specifically, the stability region is defined as the set of arrival rates for which there is a service scheduling policy stabilizing the queues, i.e. such that $\lim_{t\to\infty}\sum_k \mathbb{E}[Q_k(t)] < \infty$ [Georgiadis06]. The proposed scheduler for user k is very simple and can be cast as follows:

$$Q_k(n+1) = \begin{cases} \left[Q_k(n) - f_k(n)\Delta\right]^+ + A_k(n), & k \in I(n); \\ \left[Q_k(n) - f_k^{loc}\Delta\right]^+ + A_k(n), & \text{otherwise} \end{cases} \tag{167}$$

where, for each slot n, the frequencies $f_k(n)$ are obtained as the solution of the optimization problem [**P.1**] and $[x]^+ := \max[0, x]$. It is useful to highlight that, if the local speed of the MUE is such that $f_k^{loc} > \lambda_k$, the user is capable to compute the instructions locally while always guaranteeing the stability of its queue. In such a case, there is no need for offloading. The interesting situation is represented by the case where $f_k^{loc} > \lambda_k$, when the user is unable to execute the computations by itself, within the latency constraint, and it needs help from the cloud server to stabilize its queue. It is then of interest to investigate under what conditions, computation offloading increases the number of applications (set of instruction arrival rates) that can be run on the mobile handset, while still ensuring the stability of the queues.
In the following, we illustrate some numerical results aimed at assessing the behavior of the queues. A key aspect affecting offloading is channel conditions, i.e. the statistical characterization of fading. Different properties can be obtained depending on the number of antennas at the transmit and receive sides. Using a Multiple-Input-Multiple-Output (MIMO) communication system, with L degrees of freedom (i.e. L statistical independent channels), the pdf of the coefficient a_k is

$$p_A(a) = \frac{1}{(L-1)!\,\bar{a}^L} a^{L-1} e^{-a/\bar{a}} u(a) \tag{168}$$

where $\bar{a}$ is the variance over each single channel coefficient and $u(a)$ denotes the unitary step function.
In Figure 144, we report the behavior of the average length of the queues of $K = 3$ MUE's, achieved through the scheduling policy versus the instruction arrival rate λ_k generated by a generic application, supposed to be the same for all users. The results are averaged over 100 independent simulations. In particular, we assume the use of different communication strategies: SISO (i.e., $L = 1$), 1x2 SIMO (i.e., $L = 2$), and 2x2 MIMO (i.e., $L = 4$). Let us assume $\Delta = 1$ at every time step. The channel parameters are $d_k = 5$, $N_0 = 10^{-3}$, $\bar{a} = 10$, and $T_b = 1/(64 \cdot 10^3)$. For the optimization problem

(153), we also consider $w_k = \alpha N_k$, with $\alpha = 420$, $L_k = N_k \cdot T_b$, $f_S = 10^8$, and $P_T = 0.01$, $\forall k$. The local computation capability of the mobile handsets is set equal to $f_k^{loc} = 2 \cdot 10^6$, for all k, which is much lower than the overall capability f_S of the server. The local computation case, which corresponds to the case where all computations are carried out locally, at the mobile handset side, is shown for comparison purposes (green curve). For each curve, the abscissa corresponding to the vertical asymptote indicates the minimal rates λ that the system is unable to support.

In case of local computation, the queues remain stable if $\lambda_k < f_k^{loc}$, for all k What is interesting to notice from Figure 144 is that computation offloading makes possible to enlarge the set of instruction arrival rates that can be handled by the system, while guaranteeing the stability of the computational queues. We can also notice how, adopting MIMO communication strategies, the stability region of the queues increases.
In fact, having more antennas enables the possibility to reduce the minimum power needed by each MUE to satisfy the delay constraint in (153). This increases the probability that, fixing a power budget constraint P_T, the optimization set in (153) is feasible, thus increasing the possibility to offload the computation to the server. In summary, offloading might considerably widen the set of applications that can be run on a mobile handset, especially adopting MIMO communication strategies, which increase the probability of offloading computations to the server.

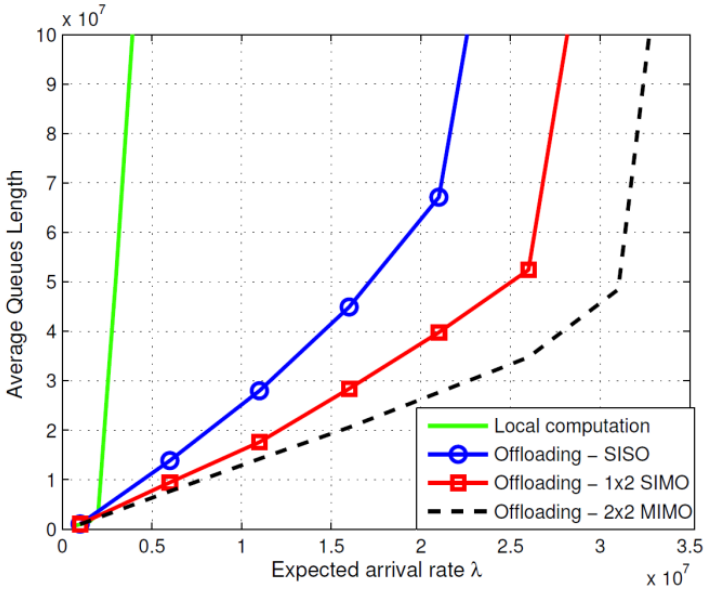


Figure 144. Average queue length vs. the number of instructions per second to be executed, with and without offloading.

5.2.3.2.1.2 *Offloading protocol*

In this section we describe the exchange of messages and information among the mobile users and the serving SCM needed to implement the offloading strategy in (153). A preliminary protocol is devised and its block diagram is shown in the following figure.

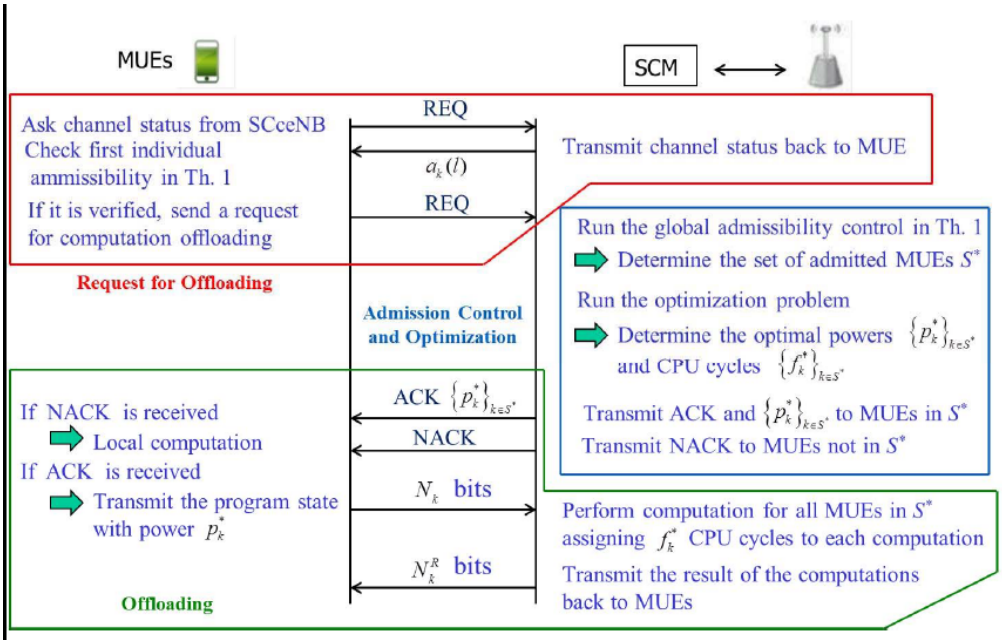


Figure 145. Block diagram for the protocol implementation.

The main steps can be resumed as follows:
Step 0. The MU requests information about the channel status to the serving SCceNB
Step 1. The SCceNB transmits the channel status to the MUE after estimating it

Step 2. The MU verifies the individual condition $a_k(n) > \frac{e^{c_k/L_k}-1}{P_T}$. If it holds the MU sends a request for computation offloading and the required parameters to the SCceNB otherwise the computation is locally performed

Step 3. The SCceNB determines the set of admitted users S^* by running the global admissibility control (165).

Step 4. The SCceNB runs the optimization problem in (153) to find out the optimal power $\{p_k^*\}_{k\in S^*}$ and CPU cycles $\{f_k^*\}_{k\in S^*}$

Step 5. The SCceNB transmits the offloading decisions to all MUEs and the optimal power $\{p_k^*\}_{k\in S^*}$ to the users admitted to offloading

Step 6. The MUs admitted to computation offloading transmit the input bits N_k to the serving SCceNB. The other MUs perform computation locally

Step 7. The SCceNB transmits the result of the computation back to MUEs.

5.2.3.2.2 Multi-cell MIMO scenario with single SCM

In this section we extend the previous offloading strategy to a MIMO multicell interfering system wherein several Mobile Users (MUs) ask for computation offloading to a common cloud server through their femto-access points [SardBarb14],[SardScutBarb]. We formulate the computation offloading problem as a *joint* optimization of the radio resources-the transmit precoding matrices of the MUs-and the computational resources-the CPU cycles/second assigned by the cloud to each MU-in order to minimize the overall users' energy consumption while meeting the latency constraints imposed by the applications running on the MUs. The resulting optimization problem is nonconvex (in the objective function and the constraints), and there are constraints coupling all the optimization variables.

To cope with the nonconvexity, we hinge on successive convex approximation techniques and propose an iterative algorithm converging to a local optimal solution of the original nonconvex problem.

Hence we devise an iterative algorithm which can be implemented in a centralized [SardScutBarb] or distributed manner [SardScutBarb14]. In this last case the distributed algorithm is performed across the access points through dual/primal decomposition techniques requiring limited coordination/signaling with the cloud. Numerical results show that the proposed joint optimization yields significant energy savings with respect to more traditional schemes performing a separate optimization of the radio and computational resources.

5.2.3.2.2.1 *Offloading over femtocloud networks*

Let us consider a femto-cell cloud network composed of N_c femto-cells, each of them served by one SCeNB, and K_n MUs in each cell $n = 1, \ldots, N_c$. We denote by i_n the i-th user in the cell n, and by $\mathrm{I} = \{\mathrm{i_n} : \mathrm{i} = 1, \ldots, \mathrm{K_n}, \mathrm{n} = 1, \ldots, \mathrm{N_c}\}$ the set of all the users. Each MU i_n and SCeNB n are equipped with $n_{T_{i_n}}$ transmit and n_{R_n} receive antennas, respectively. The femto-cells are connected to a common cloud provider that is able to serve concurrently multiple users accessing through the associated femto-cell. We assume that MUs in the same cell transmit over orthogonal channels, whereas users of different cells may interfere against each other.

In this scenario, each MU i_n is willing to run an application within a given time T_{i_n} (imposed by the application) while minimizing the resulting energy consumption. The application is structured as a graph composed by computational components whose parameters are: i) the number b_{i_n} of input bits; ii) the number of CPU cycles $\omega_{i_n} = \omega(b_{i_n})$ associated with the b_{i_n} bits; iii) the number $b_{i_n}^{\mathrm{o}}$ of output bits (the result of the computation).

We assume that the instructions to be executed are available at the server, otherwise they can be downloaded by the server through a high speed wired link. The MU can perform its computations locally or offload them to the cloud, based on which strategy requires less energy, while satisfying the latency constraint. In case of offloading, the overall latency experienced by each MU i_n is given by

$$\Delta_{i_n} = \Delta_{i_n}^{\mathrm{t}} + \Delta_{i_n}^{\mathrm{exe}} + \Delta_{i_n}^{\mathrm{tx/rx}} \tag{169}$$

where $\Delta_{i_n}^{\mathrm{t}}$ is the time necessary for the MU i_n to transfer the input bits b_{i_n} (associated with the set of instructions to be run on the cloud) to its SCeNB; $\Delta_{i_n}^{\mathrm{exe}}$ is the time for the server to execute ω_{i_n} CPU cycles; and $\Delta_{i_n}^{\mathrm{tx/rx}}$ is the round-trip time of the information (b_{i_n} and $b_{i_n}^{\mathrm{o}}$) between SCeNB n and the cloud through the backhaul link. We assume that this link is a dedicated high speed connection (e.g., fiber optics) with constant latency, resulting thus in a constant $\Delta_{i_n}^{\mathrm{tx/rx}}$. We derive next an explicit expression of $\Delta_{i_n}^{\mathrm{t}}$ and $\Delta_{i_n}^{\mathrm{exe}}$ as a function of the radio and computational resources (to be optimized).

Radio resources: The optimization variables at radio level are the users' transmit covariance matrices $\mathbf{Q} = (\mathbf{Q}_{\mathrm{i_n}})_{\mathrm{i_n} \in \mathrm{I}}$, subject to power budget constraints

$$\mathcal{Q}_{i_n} = \left\{ \mathbf{Q}_{i_n} \in C^{n_{T_{i_n}} \times n_{T_{i_n}}} : \mathbf{Q}_{i_n} \succeq \mathbf{0}, \mathrm{tr}\left(\mathbf{Q}_{i_n}\right) \leq P_{i_n} \right\}, \tag{170}$$

where P_{i_n} is the average transmit power of user i_n. We will denote by $\mathcal{Q}$ the joint set $\mathrm{Q} = \left(\prod_{\mathrm{i_n} \in \mathrm{I}} \mathrm{Q_{i_n}}\right)$. For any given profile $\mathbf{Q} = (\mathbf{Q}_{\mathrm{i_n}})_{\mathrm{i_n} \in \mathrm{I}}$, the maximum achievable rate of the MU i_n is:

$$r_{i_n}(\mathbf{Q}) = \log_2 \det\left(\mathbf{I} + \mathbf{H}_{i_n n}^H \mathbf{R}_{i_n}(\mathbf{Q}_{-n})^{-1} \mathbf{H}_{i_n n} \mathbf{Q}_{i_n},\right) \tag{171}$$

where

$$\mathbf{R}_{i_n}(\mathbf{Q}_{-n}) = \mathbf{R}_n + \sum_{n \neq m=1}^{N_c} \sum_{j=1}^{K_m} \mathbf{H}_{j_m n} \mathbf{Q}_{j_m} \mathbf{H}_{j_m n}^H, \tag{172}$$

is the covariance matrix of the noise (with $\mathbf{R}_n \succ \mathbf{0}$) plus the inter-cell interference at the SCeNB n (treated as additive noise); $\mathbf{H}_{i_n n}$ is the channel matrix of the (up)link i in the cell n, and $\mathbf{H}_{j_m n}$ is the (cross-)channel matrix between the interferer MU j in the cell m and the SCeNB of cell n; and we denoted by $\mathbf{Q}_{-n} = ((\mathbf{Q}_{j_m})_{j=1}^{K_m})_{n \neq m=1}^{N_c}$ the tuple of the covariance matrices of all users interfering with the SCeNB n.

Given each $r_{i_n}(\mathbf{Q})$, the time $\Delta_{i_n}^{\mathsf{t}}$ necessary for user i to transmit the input bits b_{i_n} of duration T_b to its SCeNB can be written as

$$\Delta_{i_n}^{\mathsf{t}} = \Delta_{i_n}^{\mathsf{t}}(\mathbf{Q}) = \frac{c_{i_n}}{r_{i_n}(\mathbf{Q})} \tag{173}$$

where $c_{i_n} = b_{i_n} T_b$.

The overall energy consumption of the MU i_n due to offloading is then

$$E_{i_n}(\mathbf{Q}_{i_n}, \mathbf{Q}_{-n}) = \mathrm{tr}(\mathbf{Q}_{i_n}) \cdot \Delta_{i_n}^{\mathsf{t}}(\mathbf{Q}), \tag{174}$$

which depends also on the covariance matrices $\mathbf{Q}_{-n}$ of the users in the other cells (due to the intercell interference).

Computational resources. The cloud provider is able to serve concurrently multiple users (accessing through the associated SCeNB). The computational resources made available by the cloud and shared among all the users are quantified in terms of the number of CPU cycles/second, set to f_T; let $f_{i_n} \geq 0$ be the fraction of f_T assigned to each user i_n. All the f_{i_n} are thus (nonnegative) optimization variables (to be determined) subject to the budget constraint $\sum_{i_n \in \mathcal{I}} f_{i_n} \leq f_T$. Given the resource assignment f_{i_n}, the time $\Delta_{i_n}^{\mathsf{exe}}$ needed to remotely run ω_{i_n} CPU cycles of user i_n is then

$$\Delta_{i_n}^{\mathsf{exe}} = \Delta_{i_n}^{\mathsf{exe}}(f_{i_n}) = \omega_{i_n} / f_{i_n}. \tag{175}$$

The expression of the overall latency Δ_{i_n} clearly shows the interplay between radio access and computational aspects, which motivates a *joint* optimization of the radio resources, the transmit covariance matrices $\mathbf{Q} = (\mathbf{Q}_{i_n})_{i_n \in I}$ of the MUs, and the computational resources, the cloud frequency assignment $\mathbf{f} = (\mathbf{f}_{i_n})_{i_n \in I}$.

System design. We formulate the offloading problem as minimization of the overall energy spent by the MUs to remotely run their applications, under latency and power constraints. More formally, we have the following:

$$\min_{\mathbf{Q},\mathbf{f}} \quad E(\mathbf{Q}) = \sum_{i,n} E_{i_n}(\mathbf{Q}_{i_n}, \mathbf{Q}_{-n})$$

$$s.t. \quad a) \quad g_{i_n}(\mathbf{Q}, f_{i_n}) = \frac{c_{i_n}}{r_{i_n}(\mathbf{Q}_{i_n}, \mathbf{Q}_{-n})} + \frac{\omega_{i_n}}{f_{i_n}} - \tilde{T}_{i_n} \le 0, \forall i_n \in I \qquad (P) \qquad (176)$$

$$b) \quad \sum_{i_n} f_{i_n} \le f_T, \quad f_{i_n} \ge 0, \forall i_n \in I$$

$$c) \quad \mathbf{Q}_{i_n} \in \mathcal{Q}_{i_n}, \quad \forall i_n \in I$$

where a) reflects the users' latency constraints $\Delta_{i_n} \le T_{i_n}$ with $\tilde{T}_{i_n}$ in a) capturing all the constant terms, i.e., $\tilde{T}_{i_n} := T_{i_n} - \Delta_{i_n}^{\mathrm{tx/rx}}$; b) imposes a limit on the cloud computational resources made available to the users and c) is a constraint on the radio resources. We denote by $\mathcal{X}$ the feasible set of the optimization problem (176).

Feasibility: Depending on the system parameters, Problem (P) may be feasible or not. In the latter case, offloading is not possible and thus the users will perform their computations locally. The following conditions are sufficient for $\mathcal{X}$ to be nonempty and thus the offloading be feasible: $\tilde{T}_{i_n} > 0$, and there exists a $\overline{\mathbf{Q}} = (\overline{\mathbf{Q}}_{i_n})_{i_n \in \mathcal{I}} \in \mathcal{Q}$ such that

$$\tilde{T}_{i_n} > \frac{c_{i_n}}{r_{i_n}(\overline{\mathbf{Q}})}, \forall i_n \in \mathcal{I}, \quad \text{and} \quad \sum_{i_n \in \mathcal{I}} \frac{\omega_{i_n}}{\tilde{T}_{i_n} - \frac{c_{i_n}}{r_{i_n}(\overline{\mathbf{Q}})}} \le f_T. \qquad (177)$$

Hereafter we will assume that Problem (P) is feasible. Problem (P) is nonconvex, due to the nonconvexity of the objective function and the constraints a). In what follows we will exploit the structure of (176) and building on some recent Successive Convex Approximation (SCA) techniques proposed in [ScutariFacch14],[ScutFacchLamp] we develop an efficient iterative inner approximation algorithm converging to a local optimal solution of (176). Numerical results show that the proposed algorithm converges in a few iterations to "good" local solutions of (P); furthermore, it is suitable for a distributed implementation across the SCeNBs, with limited coordination and signalling with the cloud.

5.2.3.2.2.2 *Closed form solutions in MIMO single-user case*

In this section we focus on the single user case by comparing the min-energy optimization strategy in (P) with an alternative strategy where the goal of the optimization strategy becomes to minimize the transmit power consumption.

We have proved that both these strategies provide the same optimal solution in the single user case. This is a remarkable result allowing to find out in closed form the optimal covariance matrix and computational rate which minimize the MU transmit power as well the transmit energy consumption. In both the proposed offloading methods we find out jointly the optimal precoding matrix (equivalently, the covariance matrix of the transmitted symbols) and the server computational rate subject to the following constraints: i) latency constraint; ii) transmit power less than available power budget; iii) computational rate at the server side less than the maximum rate made available by the server for computation.

Minimum power offloading strategy

Assuming that the goal of the offloading strategy is to minimize the transmit power consumption we can formulate the optimization problem as

$$\begin{aligned} &\min_{\mathbf{Q},f} \quad \mathrm{tr}(\mathbf{Q}) \qquad (\mathcal{P}) \\ &s.t. \quad \text{a)} \ \frac{c}{\log_2 \det\left(\mathbf{I}+\mathbf{H}\mathbf{Q}\mathbf{H}^H\mathbf{R}_w^{-1}\right)}+\frac{\omega}{f}-\tilde{T}\le 0 \\ &\quad \text{b)} 0\le f\le f_T \\ &\quad \text{c)} \mathrm{tr}(\mathbf{Q})\le P_T, \quad \mathbf{Q}\succeq\mathbf{0} \end{aligned} \Bigg\} = \mathcal{X} \tag{178}$$

In the following theorem whose proof is given in [SardBarb14], we provide the conditions under which the set $\mathcal{X}$ is non-empty and convex by finding out a closed form solution for problem $(\mathcal{P})$.

Theorem 4. *Let* $\mathbf{H}^H\mathbf{R}_n^{-1}\mathbf{H}$ *a positive definite hermitian matrix of rank* r *whose eigendecomposition is defined as* $\mathbf{H}^H\mathbf{R}_n^{-1}\mathbf{H}=\mathbf{U}\mathbf{D}_c\mathbf{U}^H$, *where* $\mathbf{U}\in C^{n_T\times n_T}$ *is a unitary matrix with the eigenvectors of* $\mathbf{H}^H\mathbf{R}_n^{-1}\mathbf{H}$. *The eigenvalues matrix* $\mathbf{D}_c\in R_+^{n_T\times n_T}$ *is a block diagonal matrix defined as* $\mathbf{D}_c=\mathrm{blkdiag}(\mathbf{0}_{n_T-r},\mathbf{D})$ *where the* $r\times r$ *-dimensional diagonal matrix* $\mathbf{D}$ *has entries the eigenvalues* d_i *for* $i=1,\ldots,r$ *sorted in increasing order. Given problem* $(\mathcal{P})$ *it admits a non-empty solution set if and only if both the following conditions hold true*

$$\log_2 \det\left(\mathbf{I}+\mathbf{H}\mathbf{Q}_w\mathbf{H}^H\mathbf{R}_n^{-1}\right)\ge \frac{c}{L^{'}}, \quad \frac{w}{\tilde{T}}<f_T \tag{179}$$

where $\mathbf{Q}_w=\mathbf{U}\left[\mathrm{blkdiag}(\mathbf{0}_{n_T-r},\frac{1}{\nu}\mathbf{I}_r-\mathbf{D}^{-1})\right]^+\mathbf{U}^H$, $L^{'}=\tilde{T}-\frac{w}{f_T}$, *with* $[x]^+=\max(0,x)$. *The Lagrangian multiplier* ν *is derived as solution of the equality constraint* $\mathrm{tr}(\mathbf{Q}_w)=P_T$. *Problem* $(\mathcal{P})$ *is convex and it admits a globally optimal solution* $(\mathbf{Q}^\star, f_s^\star)$ *with* $f_s^\star=f_T$. *The optimal covariance matrix* $\mathbf{Q}^\star$ *solving* $(\mathcal{P})$ *is given by*

$$\mathbf{Q}^\star=\mathbf{U}\left[\mathrm{blkdiag}(\mathbf{0}_{n_T-r},\alpha\mathbf{I}_r-\mathbf{D}^{-1})\right]^+\mathbf{U}^H \tag{180}$$

where $\alpha=\dfrac{\mu}{\log(2)(\lambda+1)}$ *is a positive constant, function of the Lagrangian multipliers* λ,μ *that can be found applying Algorithm I.*

Interestingly, it can be noted that the optimal solution is like water-filling with a water level depending on all the parameters involved in the offloading strategy, i.e. maximum tolerable delay, size of the program state, number of CPU cycles. In the following section it will be show that the optimal solution (180) of $(\mathcal{P})$ is also the optimal one which minimizes the energy consumption.

Algorithm I: MIMO Single User Algorithm
(S.1): Set $r=\mathrm{rank}(\mathbf{H}^H\mathbf{R}_n^{-1}\mathbf{H})$ sort d_i in decreasing order and $L^{'}=\tilde{T}-\omega/f_T$
(S.2): $r_e=r$ Repeat (a) Compute $\alpha=2^{\frac{c}{r_e L^{'}}-\frac{1}{r_e}\sum_{i=1}^{r_e}\log_2(d_i)}$

(b): If $p_i = (\alpha - 1/d_i) \geq 0$, $\forall i = 1,\ldots,r_e$, and $\sum_{i=1}^{r_e} p_i \leq P_T$,

then STOP else $r_e = r_e - 1$;

until $r_e \geq 1$.

Minimum energy offloading strategy

An alternative offloading optimization criterion is to minimize the MU transmit energy consumption by solving the following optimization problem

$$\begin{aligned} \min \quad & E(\mathbf{Q}) \quad (\mathcal{P}_e) \\ \text{s.t.} \quad & (\mathbf{Q}, f) \in \mathcal{X} \end{aligned} \tag{181}$$

where observe that $\mathcal{X}$ is the same set of problem $(\mathcal{P})$. In [SardBarb14] we have proved the following theorem.

Theorem 5. *Given problem* $(\mathcal{P}_e)$ *we can state that: a) it admits a non-empty solution set if and only if both the conditions in Th. 4 hold true; b) the feasible set* $\mathcal{X}$ *is convex and the objective function* $E(\mathbf{Q})$ *is a pseudo-convex function; c) the optimal solution of* $(\mathcal{P}_e)$ *is a global minimum of* $(\mathcal{P}_e)$*; d) the optimal solution of* $(\mathcal{P}_e)$ *coincides with the optimum solution of* $(\mathcal{P})$.

According to this result in the following we focus on problem $(\mathcal{P}_e)$ being the offloading decision based on a minimum energy consumption principle.

Numerical results

To assess the effectiveness of the proposed strategies let us discuss in this section some numerical results. As first numerical example we report in Figure 146 the mobile energy consumption which is $\mathcal{E}_{off} = E(\mathbf{Q}^\star)$ in case of offloading, i.e. if $\mathcal{E}_{off} \leq \mathcal{E}_{loc}$ and $\mathcal{E}_{loc} = \frac{w}{f_{loc}} P_{loc}$ otherwise, where P_{loc} represents the power spent by the mobile user to run the application locally. The curves have been averaged over 100 statistically independent channel realizations comparing the numerical results obtained by solving $(\mathcal{P})$ through the optimization solver cvx (circle markers) or by using Algorithm I (star markers). The parameters have been set as $f_T = 10^8$, $f_{loc} = 10^4$, $L = 0.1$, $w = 10^5$, $\mathbf{R}_n = N_0 \mathbf{I}_{n_r}$, $N_0 = 10^{-1}$, $P_T = 1$ and $P_{loc} = 0.3$. The type of application is captured by the ratio w/b which represents the number of instructions to be executed at the server side for an input bit sequence of length b transmitted by the mobile user to transfer the program state to the server. It can be observed the perfect matching between the analytical and numerical results.

In Figure 147 we report the average energy consumption for different MIMO configurations and several applications. From the figure it can be noted as for short distance the offloading is more advantageous and the use of MIMO configurations with higher receiver antennas effectively expands the offloading region. Additionally, it can be observed by the bottom figure that offloading is beneficial for those computational intensive applications which incur a low data volume to be transferred.

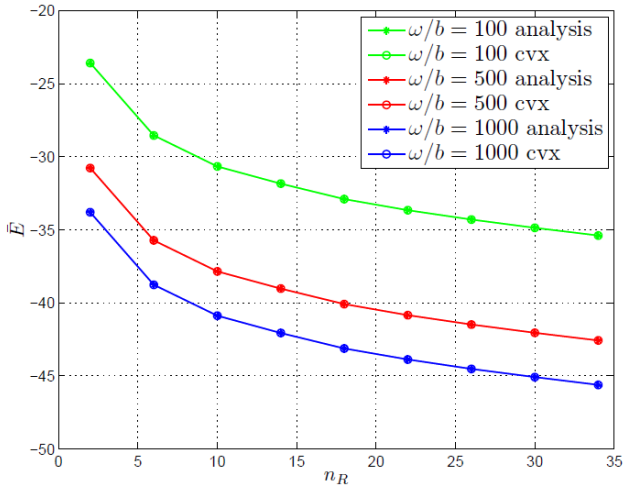


Figure 146. Average energy consumption spent for offloading vs the number of receiving antennas n_r for different applications: $w/b = 100$, $w/b = 500$, $w/b = 1000$.

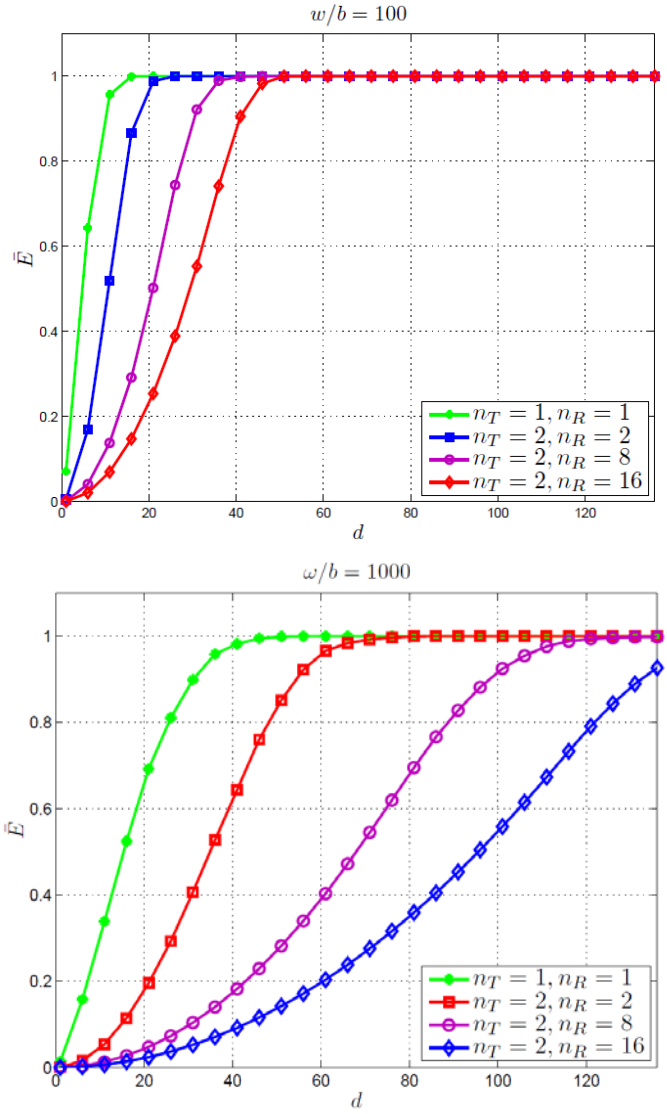


Figure 147. Average energy consumption spent for offloading vs the distance d between the mobile user and the radio access point for different applications: (top) $w/b = 100$, $w/b = 1000$ (bottom).

Multiple channels SISO case

As particular case of problem $(\mathcal{P})$ we can consider the multichannel SISO OFDM transmission. Assuming that the mobile handset transmits over a set of M sub bands, the channel matrix $\mathbf{H}$ is an $M \times M$ diagonal matrix with entries the transfer function of the channel evaluated on the i-th sub channel $H(i) = \sum_{k=0}^{L} h(k) \exp(-j2\pi ki / M)$, for $i = 0, \ldots, M-1$ where $h(k)$ represent the $L+1$ coefficients describing the finite impulse channel response. Hence the optimal covariance matrix $\mathbf{Q}$ is a diagonal matrix with entries the transmit powers p_k, allocated to each sub channel and the transmission rate can be written as $\log_2 \det\left(\mathbf{I}_M + (N_0 d^{\eta})^{-1} \mathbf{H}\mathbf{H}^H \mathbf{Q}\right) = \sum_{k=1}^{M} \log_2\left(1 + a_k p_k\right)$ where $a_k = \frac{|H(k)|^2}{N_0 d^{\eta}}$, d is the distance between the MUE and the SCceNB and η is the path loss coefficient. Denoting with $\mathbf{p} = (p_1, \ldots, p_M)$ the set of powers, problem $(\mathcal{P})$ is reduced to

$$\min_{\mathbf{p} \geq \mathbf{0},\, f \geq 0} \quad \frac{\sum_{k=1}^{M} p_k}{\sum_{k=1}^{M} \log_2\left(1 + a_k p_k\right)} \qquad (\mathcal{P}_m)$$

$$\text{s.t.} \qquad (\mathbf{p}, f) \in \mathcal{X}_m \tag{182}$$

The normalization by M in the rate term $\frac{1}{M} \sum_{k=1}^{M} \log_2\left(1 + a_k p_k\right)$ has been introduced to measure the rate in terms of number of bits per channel use. Hence Theorem 4 and Algorithm I can be straightforward applied.

To assess the effectiveness of the proposed offloading strategy we report in Figure 148 the MUE energy consumption $\overline{E}$ averaged over 100 channel realizations versus the distance d between the mobile terminal and the radio access point in the single and multi-channel case. The parameters have been set as $M = 10$, $f_{loc} = 10^3$, $f_T = 10^8$, $\omega = 10^5$, $\eta = 2$, $P_T = 1$, $\tilde{T} = 0.1$. It can be noted as expected that in the multi-channel case the energy consumption is lower since the mobile user is able to exploit channel diversity to decrease its energy consumption. Additionally, in Figure 149 we can observe in the upper plot the optimal per channel power allocation corresponding to the channel profile in the bottom subplot. Note that the proposed algorithm tends to allocate more power where the channel gain is higher in order to increase the transmission rate.

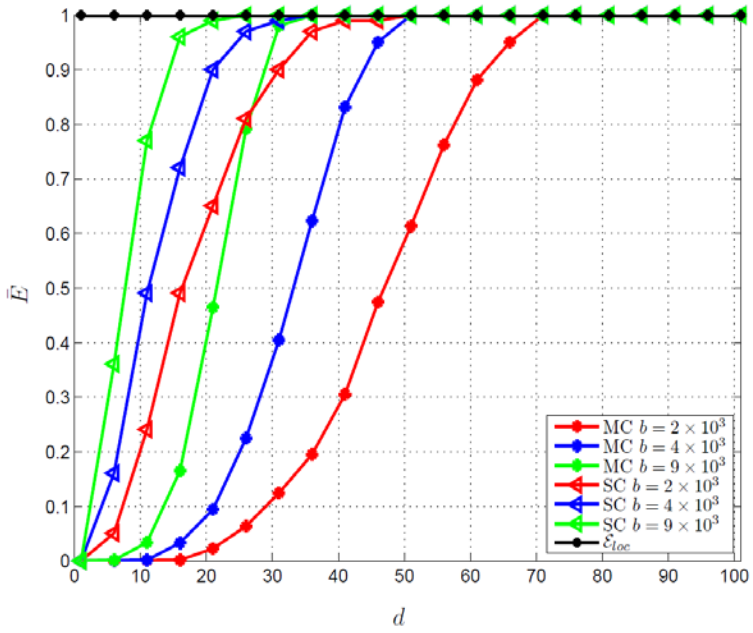


Figure 148. Single channel and multi-channel average energy consumption vs the distance d .

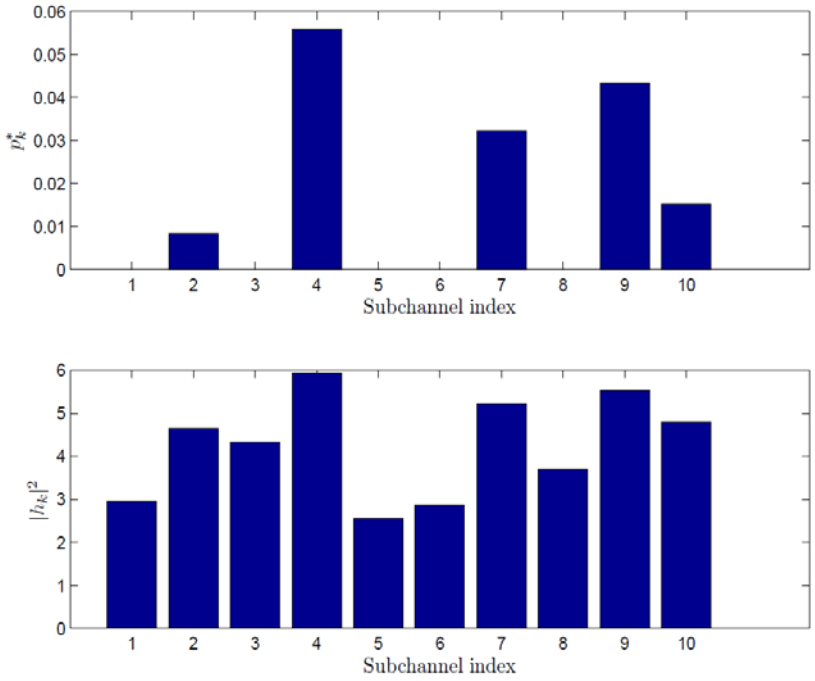


Figure 149. Optimal per-channel power allocation (upper subplot) corresponding to the channel profile in the bottom subplot.

5.2.3.2.2.3 *Algorithmic design in MIMO multi-cell networks*

In this section we focus on the MIMO multi-cell scenario. To solve the non-convex problem (P) efficiently, we develop a SCA-based method where the original problem (P) is replaced by a sequence of strongly convex problems. At the basis of the proposed technique there is a suitably chosen convex approximations of the nonconvex objective function $E(\mathbf{Q})$ and the constraints $g_{i_n}(\mathbf{Q}, f_{i_n})$, which are preliminary discussed next.

5.2.3.2.2.3.1 Approximant of $E(\mathbf{Q})$

The main idea is to approximate around the current (feasible) iterate $\mathbf{Q}^{\nu}=(\mathbf{Q}_{i_n}^{\nu})_{i_n\in I}\in\mathcal{X}$ the original nonconvex nonseparable objective function $E(\mathbf{Q})$ with a strongly convex function, say $\tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu})$, that is separable in the MUs' variables and has the same first order behavior of $E(\mathbf{Q})$ at $\mathbf{Q}^{\nu}$ [ScutariFacch14],[ScutFacchLamp]. To motivate the proposed choice of $\tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu})$, observe preliminary that, for any given $\mathbf{Q}_{-n}=\mathbf{Q}_{-n}^{\nu}$, each term $E_{i_n}(\mathbf{Q}_{i_n},\mathbf{Q}_{-n}^{\nu})$ in $E(\mathbf{Q})$ is bi-convex in $\mathbf{Q}_{i_n}$ [cf. (174)], and the other terms of the sum, $\sum_{n\neq m=1}^{N_c}\sum_{j=1}^{K_m}E_{j_m}(\mathbf{Q}_{i_n},\mathbf{Q}_{-i_n,j_m}^{\nu})$ with $\mathbf{Q}_{-i_n,j_m}^{\nu}=(\mathbf{Q}_{j_m}^{\nu},(\mathbf{Q}_{l_q}^{\nu})_{\forall l,q\neq m,l_q\neq i_n})$, are not convex in $\mathbf{Q}_{i_n}$. Exploiting such a structure, a convex approximation of the sum-energy function $E(\mathbf{Q})$ for each MU i_n can be obtained by convexifying the bi-convex term in $E_{i_n}(\mathbf{Q}_{i_n},\mathbf{Q}_{-n}^{\nu})$ [cf. (174)], and linearizing the nonconvex part $\sum_{n\neq m=1}^{N_c}\sum_{j=1}^{K_m}E_{j_m}(\mathbf{Q}_{i_n},\mathbf{Q}_{-i_n,j_m}^{\nu})$. More formally, for each i_n we introduce the "approximation" function $\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu})$:

$$\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu})=\frac{c_{i_n}\cdot\mathrm{tr}(\mathbf{Q}_{i_n})}{r_{i_n}(\mathbf{Q}_{i_n}^{\nu},\mathbf{Q}_{-n}^{\nu})}+\frac{c_{i_n}\cdot\mathrm{tr}(\mathbf{Q}_{i_n}^{\nu})}{r_{i_n}(\mathbf{Q}_{i_n},\mathbf{Q}_{-n}^{\nu})}+\sum_{n\neq m=1}^{N_c}\sum_{j=1}^{K_m}\left\langle\nabla_{\mathbf{Q}_{i_n}^{*}}E_{j_m}(\mathbf{Q}^{\nu}),\mathbf{Q}_{i_n}-\mathbf{Q}_{i_n}^{\nu}\right\rangle+\tau_{i_n}\left\|\mathbf{Q}_{i_n}-\mathbf{Q}_{i_n}^{\nu}\right\|^2 \tag{183}$$

where the first term on the right-hand side is the aforementioned convexification of the bi-convex term in (174); and the second term comes from the linearization of $\sum_{n\neq m=1}^{N_c}\sum_{j=1}^{K_m}E_{j_m}(\mathbf{Q}_{i_n},\mathbf{Q}_{-i_n,j_m}^{\nu})$, with $\langle\mathbf{A},\mathbf{B}\rangle=\mathrm{tr}(\mathbf{A}^H\mathbf{B})$ and $\nabla_{\mathbf{Q}_{i_n}^{*}}E_{j_m}(\mathbf{Q})$ denoting the conjugate gradient of $E_{j_m}(\mathbf{Q})$ with respect to $\mathbf{Q}_{i_n}$ evaluated at $\mathbf{Q}^{\nu}$, and given by

$$\nabla_{\mathbf{Q}_{i_n}^{*}}E_{j_m}(\mathbf{Q})=\frac{\mathrm{tr}(\mathbf{Q}_{j_m}^{\nu})\Delta_{j_m}(\mathbf{Q}^{\nu})}{r_{j_m}(\mathbf{Q}^{\nu})}\cdot\left[\mathbf{H}_{i_nm}^H\left(\mathbf{R}_{j_m}(\mathbf{Q}_{-m}^{\nu})^{-1}-(\mathbf{R}_{j_m}(\mathbf{Q}_{-m}^{\nu})+\mathbf{H}_{j_mm}\mathbf{Q}_{j_m}^{\nu}\mathbf{H}_{j_mm}^H)^{-1}\right)\mathbf{H}_{i_nm}\right]. \tag{184}$$

Note that in (183) we also added a quadratic regularization, making $\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu})$ uniformly strongly convex on $\mathcal{Q}_{i_n}$.

Based on each $\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu})$, we can now define the candidate sum-energy approximation $\tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu})$ as: given $\mathbf{Q}^{\nu}\succcurlyeq\mathbf{0}$,

$$\tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu})=\sum_{n=1}^{N_c}\sum_{i=1}^{K_n}\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu}). \tag{185}$$

Note that $\tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu})$ has many desirable properties, such as: i) it has the same first-order behavior of the original nonconvex function $E(\mathbf{Q})$ at $\mathbf{Q}^{\nu}$, i.e., $\nabla_{\mathbf{Q}^{*}}E(\mathbf{Q}^{\nu})=\nabla_{\mathbf{Q}^{*}}\tilde{E}(\mathbf{Q}^{\nu};\mathbf{Q}^{\nu})$; ii) it is separable in the users variables $\mathbf{Q}_{i_n}$ (which is instrumental to obtain distributed algorithms, see [SardBarb14] for details); and iii) it is uniformly strongly convex on $\mathcal{Q}$ [SardBarb14]; we will denote by $c_{\tau}>0$ the constant of strong convexity, that c_{τ} is the largest positive scalar such that

$$\left\langle \mathbf{Q}^1-\mathbf{Q}^2, \nabla_{\mathbf{Q}^*}\tilde{E}\left(\mathbf{Q}^1;\mathbf{Q}^\nu\right)-\nabla_{\mathbf{Q}^*}\tilde{E}\left(\mathbf{Q}^2;\mathbf{Q}^\nu\right)\right\rangle \geq c_\tau \left\|\mathbf{Q}^1-\mathbf{Q}^2\right\|^2, \quad \forall \mathbf{Q}^1,\mathbf{Q}^2 \in \mathcal{Q}. \tag{186}$$

An explicit expression of c_τ is given in [SarBarb14] and is omitted here because of space limitation; we only observe that $c_\tau \geq \min_{i_n\in\mathcal{I}}\{\tau_{i_n}\}$.

5.2.3.2.2.3.2 Inner convexification of the constraints $g_{i_n}(\mathbf{Q}, f_{i_n})$

We aim at introducing an inner convex approximation of the constraints $g_{i_n}(\mathbf{Q}, f_{i_n})$ around $\mathbf{Q}^\nu \in \mathcal{X}$ denoted by $\tilde{g}_{i_n}(\mathbf{Q}, f_{i_n};\mathbf{Q}^\nu)$. To do so, let us exploit first the concave-convex structure of the rate functions $r_{i_n}(\mathbf{Q})$ [cf. (171)]:

$$r_{i_n}(\mathbf{Q}) = r_{i_n}^+(\mathbf{Q}) + r_{i_n}^-(\mathbf{Q}_{-n}), \tag{187}$$

with

$$\begin{aligned} r_{i_n}^+(\mathbf{Q}) &= \log_2 \det\left(\mathbf{R}_{i_n}(\mathbf{Q}_{-n}) + \mathbf{H}_{i_n n}\mathbf{Q}_{i_n}\mathbf{H}_{i_n n}^H\right) \\ r_{i_n}^-(\mathbf{Q}_{-n}) &= -\log_2 \det\left(\mathbf{R}_{i_n}(\mathbf{Q}_{-n})\right) \end{aligned} \tag{188}$$

and $\mathbf{R}_{i_n}(\mathbf{Q}_{-n})$ defined in (172). Note that $r_{i_n}^+(\bullet)$ and $r_{i_n}^-(\bullet)$ are concave on $\mathcal{Q}$ and convex on $\mathcal{Q}_{-n} = \prod_{m\neq n}\mathcal{Q}_m$, respectively. Using (187), and observing that at any feasible $\mathbf{Q},\mathbf{f}$, it must be $r_{i_n}(\mathbf{Q}) > 0$ and $f_{i_n} > 0$ for all i and n, the constraints $g_{i_n}(\mathbf{Q}, f_{i_n}) \leq 0$ in (P) can be rewritten as

$$g_{i_n}(\mathbf{Q}, f_{i_n}) = -r_{i_n}^+(\mathbf{Q}) - r_{i_n}^-(\mathbf{Q}_{-n}) + \frac{c_{i_n}\cdot f_{i_n}}{f_{i_n}\cdot\tilde{T}_{i_n} - \omega_{i_n}} \leq 0. \tag{189}$$

The desired inner convex approximation $\tilde{g}_{i_n}(\mathbf{Q}, f_{i_n};\mathbf{Q}^\nu)$ is obtained from $g_{i_n}(\mathbf{Q}, f_{i_n})$ by retaining the convex part in (189) and linearizing the concave term $-r_{i_n}^-(\mathbf{Q}_{-n})$, resulting in:

$$\tilde{g}_{i_n}(\mathbf{Q}, f_{i_n};\mathbf{Q}^\nu) = -r_{i_n}^+(\mathbf{Q}) + \frac{c_{i_n}\cdot f_{i_n}}{f_{i_n}\cdot\tilde{T}_{i_n} - \omega_{i_n}} - r_{i_n}^-(\mathbf{Q}_{-n}^\nu) - \sum_{m\neq n, j\neq i}\left\langle \nabla_{\mathbf{Q}_{jm}^*} r_{i_n}^-(\mathbf{Q}_{-n}^\nu), \mathbf{Q}_{j_m} - \mathbf{Q}_{j_m}^\nu \right\rangle \tag{190}$$

with $\nabla_{\mathbf{Q}_{jm}^*} r_{i_n}^-(\mathbf{Q}_{-n}^\nu) = -\mathbf{H}_{j_m n}^H \mathbf{R}_{i_n}(\mathbf{Q}_{-n}^\nu)^{-1}\mathbf{H}_{j_m n}$.

5.2.3.2.2.4 *SCA algorithm: Centralized approach*

We are now ready to introduce the proposed convex approximation of the nonconvex problem (176), which consists in replacing the nonconvex objective function $E(\mathbf{Q})$ and constraints $g_{i_n}(\mathbf{Q}, f_{i_n}) \leq 0$ in (176) with the approximations $\tilde{E}(\mathbf{Q};\mathbf{Q}^\nu)$ and $\tilde{g}_{i_n}(\mathbf{Q}_{i_n}, f_{i_n};\mathbf{Q}^\nu) \leq 0$, respectively. More formally, given the feasible point $\mathbf{Q}^\nu$, we have

$$\hat{\mathbf{Z}}(\mathbf{Z}^{\nu}) = \underset{\mathbf{Q},\mathbf{f}}{\operatorname{argmin}} \quad \tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu}) + \frac{c_f}{2}\left\|\mathbf{f}-\mathbf{f}^{\nu}\right\|^2$$

$$\text{s.t.} \quad a)\ \ \tilde{g}_{i_n}(\mathbf{Q}, f_{i_n};\mathbf{Q}^{\nu}) \le 0, \forall i_n \in I \qquad P^{\nu} \qquad (191)$$

$$b)\sum_{i_n \in \mathcal{I}} f_{i_n} \le f_T \text{ and } f_{i_n} \ge 0, \forall i_n \in I$$

$$c)\ \ \mathbf{Q}_{i_n} \in \mathcal{Q}_{i_n}, \quad \forall i_n \in I$$

where we denoted by $\hat{\mathbf{Z}}(\mathbf{Z}^{\boldsymbol{\nu}}) \coloneqq (\hat{\mathbf{Q}}(\mathbf{Z}^{\boldsymbol{\nu}}), \hat{\mathbf{f}}(\mathbf{Z}^{\boldsymbol{\nu}}))$ the unique solution of the strongly convex optimization problem (note that we added in the objective function a quadratic term in the f -variables, in order to make the objective strongly convex in $\mathbf{f}$, with c_f being an arbitrary positive constant).

The proposed solution method consists in solving the sequence of problems P^{ν}, starting from a feasible $\mathbf{Z}^0 = (\mathbf{Q}^0, \mathbf{f}^0)$. The formal description of the algorithm is given in Algorithm 2 below, and its convergence to local optimal solution of the original nonconvex problem (P) are stated in Theorem 6 (the proof of the theorem is omitted because of the space limitation; it consists in showing that all the conditions in [Th.1] in [ScutariFacch14] are satisfied; see [SardBarb14]). Note that in Step 3 of the algorithm we allow a memory in the update of the iterate $\mathbf{Z}^{\nu} = (\mathbf{Q}^{\nu}, \mathbf{f}^{\nu})$. A practical termination criterion in Step 2 is to stop the iterates when $|E(\mathbf{Q}^{\nu+1}) - E(\mathbf{Q}^{\nu})| \le \delta$, where $\delta > 0$ is the prescribed accuracy.

Algorithm 2: Inner SCA Algorithm for (P)
Data: $\mathbf{Z}^0 = (\mathbf{Q}^0, \mathbf{f}^0) \in \mathcal{X}$; $\{\gamma^{\nu}\}_{\nu} \in (0,1]$; $c_{\tau} > 0$; $c_f > 0$. Set $\nu = 0$
(S.1): If $\mathbf{Z}^{\nu}$ satisfies a suitable termination criterion, STOP
(S.2): Compute $\hat{\mathbf{Z}}(\mathbf{Z}^{\nu}) = (\hat{\mathbf{Q}}(\mathbf{Z}^{\nu}), \hat{\mathbf{f}}(\mathbf{Z}^{\nu}))$ [cf. (P^{ν})];
(S.3): Set $\mathbf{Z}^{\nu+1} = \mathbf{Z}^{\nu} + \gamma^{\nu}\left(\hat{\mathbf{Z}}(\mathbf{Z}^{\nu}) - \mathbf{Z}^{\nu}\right)$;
(S.4): $\nu \leftarrow \nu + 1$ and go to (S.1).

Theorem 6. *Given the non convex problem (P), choose* $c_{\tau} > 0$, $c_f > 0$, *and* $\{\gamma^{\nu}\}_{\nu}$ *such that*

$$(0,1] \ni \gamma^{\nu} \to 0, \forall \nu \ge 0, \quad \text{and} \quad \sum_{\nu} \gamma^{\nu} = +\infty. \qquad (192)$$

Then every limit point of $\{\mathbf{Z}^{\nu}\}$ *is a stationary solution of* (P). *Furthermore, none of such points is a local maximum of the energy function* E.

Theorem VI offers some flexibility in the choice of the free parameters c_{τ}, c_f, and $\{\gamma^{\nu}\}_{\nu}$ while guaranteeing convergence of Algorithm 2. For instance, c_{τ} is positive if all τ_{i_n} are positive (but arbitrary); in the case of full-column rank matrices $\mathbf{H}_{i_n n}$, one can also set $\tau_{i_n} = 0$ (still resulting in $c_{\tau} > 0$). Any $c_f > 0$ is instead admissible. Many choices are possible for the step-size γ^{ν}; a practical rule satisfying (192) that we found effective in our experiment is [ScutariFacchinei]:

$$\gamma^{\nu+1} = \gamma^{\nu}(1 - \alpha\gamma^{\nu}), \quad \gamma^0 = 1, \qquad (193)$$

with $\alpha \in [0,1)$.

5.2.3.2.2.4.1 Numerical results

In this section we present some numerical results to assess the performance of the proposed offloading strategy. Our experiments are run under the following setup. We consider a $N_c = 2$ cell MIMO network, where all the transceivers are equipped with $n_T = n_R = 2$ antennas (unless stated otherwise); in each cell there are $K_n = 3$ active users, randomly deployed. The other system parameters are set as follows: $f_T = 2 \cdot 10^7$, $\mathbf{R}_n = \sigma^2 \mathbf{I}$, with $\text{snr} = \frac{P_T}{\sigma^2} = 3\text{dB}$. We simulated Algorithm 1 using the diminishing step-size rule (193), with $\alpha = 1\text{e}{-5}$; the termination accuracy δ is set to 10^{-2}.

Example # 1: Energy consumption.

We start comparing the energy consumption of the proposed offloading strategy with that of a more traditional method where the optimization of the communication and computational resources is performed separately. We consider the optimization problem (P) wherein the optimization variables are only the MUs' covariance matrices, while the f_{i_n} are fixed and proportional to the computational load of each user, i.e. $f_{i_n} = w_{i_n} f_T / \sum_{i_n \in \mathcal{I}} w_{i_n}$ CPU cycles/second; we termed such a method Disjoint Resources Assignment (DRA) algorithm.

In Figure 150 we plot the energy consumption resulting from Algorithm 2 and DRA versus the ratio $\eta = w_{i_n} / b_{i_n}$ (assumed to be equal for all the users). The curves are averages over 100 independent realizations of Rayleigh fading channels. . The system parameters have been chosen so that offloading is feasible for all the algorithms over the simulated channel realizations.

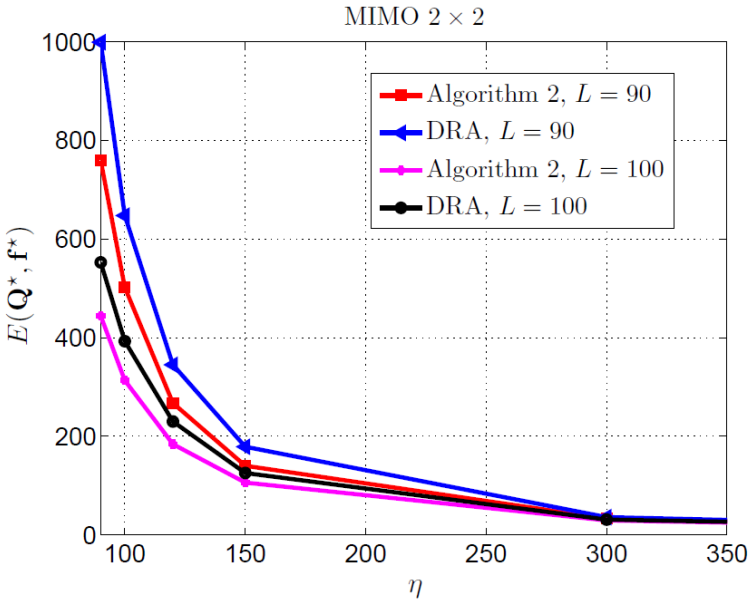


Figure 150. Energy consumption vs. $\eta = w_{i_n} / b_{i_n}$: Proposed scheme (Algorithm 2) and DRA methods.

The analysis of the figure clearly shows that our scheme yields an energy saving with respect to the DRA simulated scheme as the delay constraint becomes more stringent and the number of transmit bits is high, which validates the proposed approach of jointly optimizing radio and computational resources. Note that the energy consumption decreases as η increases, showing that the applications more suitable for offloading are those characterized by a limited input (or state) size (i.e., small b_{i_n}) and a high number of CPU cycles to be executed (i.e., large ω_{i_n}).

Example # 2: Convergence speed. In Figure 151 we test the convergence speed of the proposed scheme; we plot the optimal energy $E(\mathbf{Q}^{\nu},\mathbf{f}^{\nu})$ [cf. (P^{ν})] versus the iteration index ν of Algorithm 1, for different values of the maximum tolerable delay $\tilde{T}_{i_n}$ (assumed to be equal for all the users). The curves correspond to a single channel realization. Quite interestingly, the proposed algorithm converges in a few iterations (we experienced a similar behaviour in all our experiments on different channel realizations). Note also that the energy consumption increases as the delay constraints become more stringent.

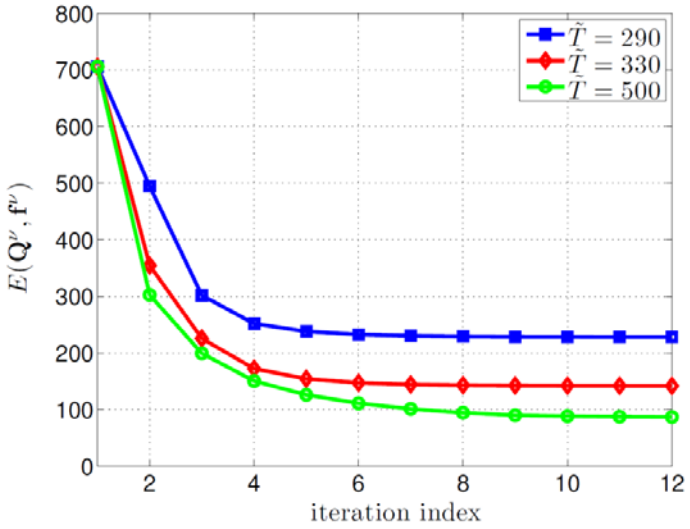


Figure 151. Convergence speed: Optimal energy vs. the iterations for different values of $\tilde{T}$.

Example # 3: Optimal resources allocation.
To grasp how the computational and radio resources are allocated across the users, in Figure 152 we show the allocation of the (normalized) CPU cycles f_{i_n} / f_T and the rates r_{i_n} resulting from Algorithm 2, for a given set of $\eta_{i_n} = w_{i_n} / b_{i_n}$ (also shown in the figure). It is interesting to observe how the proposed strategy tends to assign the highest computational rate f_{i_n} to the users with the most demanding applications (the highest ratio η_{i_n}).

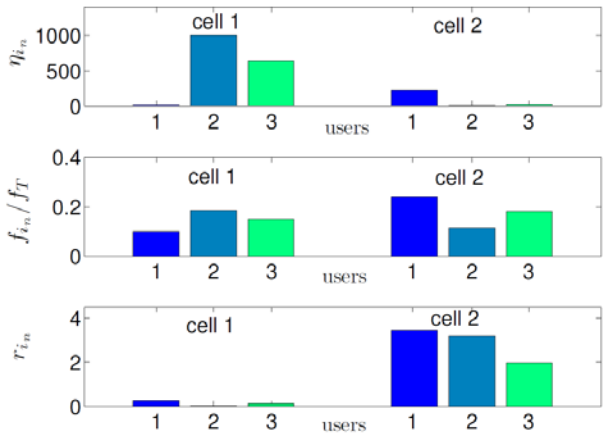


Figure 152. Radio and computational resource allocation (SISO channels): η_{i_n}, f_{i_n} / f_T, and r_{i_n} of the users in the cells.

5.2.3.2.2.5 *SCA algorithm: Cell-distributed approach*

In this section we show how to solve problem P^{ν} in a distributed way across the cells, invoking decomposition techniques for convex optimization problems. More specifically, we propose two different distributed algorithms, namely a dual and a primal-based decomposition scheme.

What prevents the direct application of primal or dual methods to P^{ν} is the presence of coupling on the MTs' covariance matrices of different cells: the constraint terms $r_{i_n}^{+}(\mathbf{Q})$ depends on all $\mathbf{Q}_{i_n}$'s. To cope with this issue we explore next a different approximation of the constraint $r_{i_n}^{+}(\mathbf{Q})$. Invoking the Lipschitz continuity of each $-r_{i_n}^{+}(\mathbf{Q})$ on $\mathcal{Q}$ with constant $L_{\nabla_{j_l}}^{i_n}$, we get

$$\begin{aligned}
&-r_{i_n}^{+}(\mathbf{Q}) \le -r_{i_n}^{\mathrm{up}}(\mathbf{Q},\mathbf{Q}^{\nu}) = -r_{i_n}^{+}(\mathbf{Q}^{\nu}) + \\
&\sum_{n\neq l=1}^{N_c}\sum_{j=1}^{K_l}\left\langle \boldsymbol{\Pi}_{j_l}^{(i_n)}(\mathbf{Q}^{\nu}), \mathbf{Q}_{j_l} - \mathbf{Q}_{j_l}^{\nu}\right\rangle + \frac{L_{\nabla_{i_n}}^{i_n}}{2}\left\|\mathbf{Q}_{i_n} - \mathbf{Q}_{i_n}^{\nu}\right\|^2 \\
&+\sum_{n\neq l=1}^{N_c}\sum_{j=1}^{K_l}\frac{L_{\nabla_{j_l}}^{i_n}}{2}\left\|\mathbf{Q}_{j_l} - \mathbf{Q}_{j_l}^{\nu}\right\|^2 + \left\langle \boldsymbol{\Pi}_{i_n}^{(i_n)}(\mathbf{Q}^{\nu}), \mathbf{Q}_{i_n} - \mathbf{Q}_{i_n}^{\nu}\right\rangle
\end{aligned} \tag{194}$$

with $\boldsymbol{\Pi}_{j_l}^{(i_n)}(\mathbf{Q}^{\nu}) = -\nabla_{\mathbf{Q}_{j_l}^{*}} r_{i_n}^{+}(\mathbf{Q}^{\nu}) = -\mathbf{H}_{j_l n}^{H}(\mathbf{R}_n(\mathbf{Q}_{-n}^{\nu}) + \mathbf{H}_{i_n n}\mathbf{Q}_{i_n}\mathbf{H}_{i_n n}^{H})^{-1}\mathbf{H}_{j_l n}$.

Then, the approximate function $\tilde{g}_{i_n}$ can be replaced by the upper bound $\tilde{q}_{i_n}$ given by

$$\tilde{q}_{i_n}(\mathbf{Q}, f_{i_n}; \mathbf{Q}^{\nu}) = -r_{i_n}^{\mathrm{up}}(\mathbf{Q},\mathbf{Q}^{\nu}) - r_n^{-}(\mathbf{Q}_{-n}^{\nu}) + \frac{c_{i_n}\cdot f_{i_n}}{f_{i_n}\cdot\tilde{T}_{i_n} - \omega_{i_n}} - \sum_{n\neq l=1}^{N_c}\sum_{j=1}^{K_l}\left\langle \nabla_{\mathbf{Q}_{j_l}^{*}} r_n^{-}(\mathbf{Q}_{-n}^{\nu}), \mathbf{Q}_{j_l} - \mathbf{Q}_{j_l}^{\nu}\right\rangle \tag{195}$$

We can then define a different sequence of subproblems (P_d^{ν}) having the form of (P^{ν}), but where the constraints $\tilde{g}_{i_n}$ are replaced by $\tilde{q}_{i_n}$. More specifically, given $\mathbf{Q}^{\nu} \in \mathcal{X}$, we can formulate the following approximation of (P^{ν})

$$\begin{aligned}
\min_{\mathbf{Q},\mathbf{f}} \quad & \tilde{E}(\mathbf{Q};\mathbf{Q}^{\nu}) + \frac{c_f}{2}\left\|\mathbf{f} - \mathbf{f}^{\nu}\right\|^2 \\
\text{s.t.} \quad & \left.\begin{array}{l} \tilde{q}_{i_n}(\mathbf{Q}, f_{i_n}; \mathbf{Q}^{\nu}) \le 0, \quad \forall i_n \in I \\ \mathbf{Q}_{i_n} \in \mathcal{Q}_{i_n}, \quad \forall i_n \in I \\ \sum_{i_n} f_{i_n} \le f_T,\ f_{i_n} \ge 0, \forall i_n \in I \end{array}\right\} = \mathcal{X}_d(\mathbf{Q}^{\nu})
\end{aligned} \tag{196}$$

Note that the objective function and the constraints $\tilde{q}_{i_n}$ are separable in the users covariance matrices $\mathbf{Q}_{i_n}$. Hence, problem (P_d^{ν}) can be solved in a distributed way using standard primal or dual decomposition techniques [Palomar07], as shown next.

5.2.3.2.2.5.1 Cell-Distributed Dual Decomposition

In this section, we propose a two layers distributed optimization strategy to solve the strongly convex problem (196) using dual decomposition (it can be easily proved that there is zero duality gap [SarBarb14]). The dual problem associated with each problem can be expressed as:

$$\max_{\boldsymbol{\lambda}\ge 0} \quad D(\boldsymbol{\lambda};\mathbf{Z}^{\nu}) \tag{197}$$

where $\mathbf{Z}^{\nu} = (\mathbf{Q}^{\nu}, \mathbf{f}^{\nu}) \in \mathcal{X}_d^{\nu}$, $\boldsymbol{\lambda} = ((\lambda_{i_n})_{i_n\in\mathcal{I}}, \lambda_f)$ is the vector of the nonnegative Lagrangian multipliers and

$$D(\boldsymbol{\lambda};\mathbf{Z}^{\nu}) = \min_{\mathbf{Q}\in\mathcal{Q},\mathbf{f}\in R_{+}^{|I|}} \quad \mathcal{L}(\mathbf{Q},\mathbf{f},\boldsymbol{\lambda};\mathbf{Z}^{\nu}) \tag{198}$$

with

$$\mathcal{L}(\mathbf{Q},\mathbf{f},\boldsymbol{\lambda};\mathbf{Z}^{\nu}) = \sum_{i_n\in I}\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu}) + \frac{c_f}{2}\left\|\mathbf{f}-\mathbf{f}^{\nu}\right\|^2 + \sum_{i_n\in I}\lambda_{i_n}\tilde{q}_{i_n}(\mathbf{Q}, f_{i_n};\mathbf{Q}^{\nu}) + \lambda_f\left(\sum_{i_n\in I} f_{i_n} - f_T\right). \tag{199}$$

It is not difficult to see that $\mathcal{L}(\mathbf{Q},\mathbf{f},\boldsymbol{\lambda};\mathbf{Z}^{\nu})$ can be rewritten as

$$\mathcal{L}(\mathbf{Q},\mathbf{f},\boldsymbol{\lambda};\mathbf{Z}^{\nu}) = \sum_{n=1}^{N_c}\mathcal{L}_{\mathbf{Q}_n}(\mathbf{Q}_n,\boldsymbol{\lambda};\mathbf{Q}^{\nu}) + \sum_{n=1}^{N_c}\mathcal{L}_{\mathbf{f}_n}(\mathbf{f}_n,\boldsymbol{\lambda};\mathbf{f}_n^{\nu}) \tag{200}$$

Where

$$\mathcal{L}_{\mathbf{f}_n}(\mathbf{f}_n,\boldsymbol{\lambda};\mathbf{f}_n^{\nu}) = \sum_{i=1}^{K_n}\left\{\frac{c_f}{2}(f_{i_n} - f_{i_n}^{\nu})^2 + \frac{\lambda_{i_n}\cdot c_{i_n}\cdot f_{i_n}}{f_{i_n}\cdot\tilde{T}_{i_n} - \omega_{i_n}} + \lambda_f f_{i_n}\right\} \tag{201}$$

$$\mathcal{L}_{\mathbf{Q}_n}(\mathbf{Q}_n,\boldsymbol{\lambda};\mathbf{Q}^{\nu}) = \sum_{i=1}^{K_n}\tilde{E}_{i_n}(\mathbf{Q}_{i_n};\mathbf{Q}^{\nu}) + \sum_{i=1}^{K_n}\lambda_{i_n}\left(-r_{i_n}^{+}(\mathbf{Q}^{\nu}) - r_n^{-}(\mathbf{Q}_{-n}^{\nu}) + \frac{L_{\nabla_{i_n}}^{i_n}}{2}\left\|\mathbf{Q}_{i_n}-\mathbf{Q}_{i_n}^{\nu}\right\|^2 + \left\langle\boldsymbol{\Pi}_{i_n}^{(i_n)}(\mathbf{Q}^{\nu}),\mathbf{Q}_{i_n}-\mathbf{Q}_{i_n}^{\nu}\right\rangle\right)$$

$$+\sum_{i=1}^{K_n}\left(\sum_{n\neq l=1}^{N_c}\sum_{j=1}^{K_l}\frac{\lambda_{j_l}L_{\nabla_{i_n}}^{j_l}}{2}\left\|\mathbf{Q}_{i_n}-\mathbf{Q}_{i_n}^{\nu}\right\|^2 - \sum_{n\neq l=1}^{N_c}\sum_{j=1}^{K_l}\left\langle\lambda_{j_l}\nabla_{\mathbf{Q}_{i_n}^{*}}(r_l^{-}(\mathbf{Q}_{-l}^{\nu}) + r_{j_l}^{+}(\mathbf{Q}^{\nu})),\mathbf{Q}_{i_n}-\mathbf{Q}_{i_n}^{\nu}\right\rangle\right) \tag{202}$$

with $\mathbf{Q}_n = (\mathbf{Q}_{i_n})_{i=1}^{K_n}$, $\mathbf{f}_n = (f_{i_n})_{i=1}^{K_n}$.

It can be proved [SardBarb14] that the dual problem (198) admits an unique solution denoted by $\hat{\mathbf{Z}}(\boldsymbol{\lambda};\mathbf{Z}^{\nu}) = (\hat{\mathbf{Z}}_n(\boldsymbol{\lambda};\mathbf{Z}^{\nu}))_{n=1}^{N_c}$, with each $\hat{\mathbf{Z}}_n(\boldsymbol{\lambda};\mathbf{Z}^{\nu}) = (\hat{\mathbf{Q}}_n(\boldsymbol{\lambda};\mathbf{Q}^{\nu}),\hat{\mathbf{f}}_n(\boldsymbol{\lambda};\mathbf{Q}^{\nu}))$, which can be found in a distributed manner as

$$\hat{\mathbf{Z}}_n(\boldsymbol{\lambda};\mathbf{Z}^{\nu}) = \operatorname*{argmin}_{\mathbf{Q}_n\in\mathcal{D}_n,\mathbf{f}_n\in R_{+}^{K_n}}\left\{\mathcal{L}_{\mathbf{f}_n}(\mathbf{f}_n,\boldsymbol{\lambda},\mathbf{f}_n^{\nu}) + \mathcal{L}_{\mathbf{Q}_n}(\mathbf{Q}_n,\boldsymbol{\lambda};\mathbf{Q}^{\nu})\right\} \tag{203}$$

where $\mathcal{D}_n = \Pi_{i=1}^{K_n}\mathcal{Q}_{i_n}$. Given the multiplier vector $\boldsymbol{\lambda}$, (203) is separable in the pairs $(\mathbf{Q}_n,\mathbf{f}_n)$. The optimal solution can be then computed in parallel across the SCeNBs solving

$$\hat{\mathbf{Q}}_n(\boldsymbol{\lambda};\mathbf{Q}^{\nu}) = \operatorname*{argmin}_{\mathbf{Q}_n\in\mathcal{D}_n}\left\{\mathcal{L}_{\mathbf{Q}_n}(\mathbf{Q}_n,\boldsymbol{\lambda};\mathbf{Q}^{\nu})\right\} \tag{204}$$

$$\hat{\mathbf{f}}_n(\boldsymbol{\lambda};\mathbf{f}^{\nu}) = \operatorname*{argmin}_{\mathbf{f}_n\in R_{+}^{K_n}}\left\{\mathcal{L}_{\mathbf{f}_n}(\mathbf{f}_n,\boldsymbol{\lambda};\mathbf{f}_n^{\nu})\right\}. \tag{205}$$

Given $(\hat{\mathbf{Q}}_n,\hat{\mathbf{f}}_n)$, the optimal multipliers $\boldsymbol{\lambda}$ can be found by the master node (femtocloud) solving the dual problem (197), e.g., via gradient algorithms; an instance of such schemes is described in Algorithm III, whose convergence is stated in Theorem 7 (see [ScutFacchLamp], [SardBarb14] for further details).

Table 35. Dual-based Distributed Algorithm.

Algorithm III: Dual-based Distributed Implementation of Step 2 of Algorithm 1.
Data: $\boldsymbol{\lambda}^0 \geq \mathbf{0}$, $\mathbf{z}^\nu = (\mathbf{Q}^\nu, \mathbf{f}^\nu)$, $\{\beta^k\} > 0$. Set $k = 0$
(S.1): If $\boldsymbol{\lambda}^k$ satisfies a suitable termination criterion, STOP
(S.2): For each SCeNB n compute in parallel $\mathbf{Q}_n^{k+1}(\boldsymbol{\lambda}^k; \mathbf{z}^\nu)$ and $\mathbf{f}_n^{k+1}(\boldsymbol{\lambda}^k; \mathbf{z}^\nu)$ by solving (204) and (205);
(S.3): Update at the master node $\boldsymbol{\lambda}^{k+1}$ according to $\lambda_{i_n}^{k+1} = \left[\lambda_{i_n}^k + \beta^k \tilde{q}_{i_n}(\mathbf{Q}_n^{k+1}, \mathbf{f}_n^{k+1}; \mathbf{Q}^\nu, \mathbf{f}^\nu)\right]^+$, $\forall i_n \in \mathcal{I}$ $\lambda_f^{k+1} = \left[\lambda_f^k + \beta^k \left(\sum_{i_n \in I} f_{i_n}^{k+1} - f_T \right)\right]^+$
(S.4): $k \leftarrow k+1$ and go back to (S.1).

Theorem 7. Given the nonconvex problem (P), suppose that $\{\beta^k\}$ is chosen so that $\beta^k > 0$, $\beta^k \to 0$, $\sum_k \beta^k = +\infty$ and $\sum_k (\beta^k)^2 < \infty$. Then, the sequence $\{\boldsymbol{\lambda}^k\}$ generated by Algorithm 2 converges to a solution of (197). Therefore, the sequence $\left\{\hat{\mathbf{Z}}_k(\boldsymbol{\lambda}^k; \mathbf{Z}^\nu)\right\}$ converges to the unique solution of (196).

5.2.3.2.2.5.2 Cell-Distributed Primal Decomposition

In this section, we propose an alternative two layers decomposition (see [ScutFacch],[Palomar_07]) of problem based on a different decomposition. For any given set of CPU frequencies $\mathbf{f} = (f_{i_n})_{n=1}^{Nc}$, problem (196) can be decoupled across the cells. At the lower level, each SCeNB solves the following convex problem where $\mathbf{f}$ is fixed

$$\begin{aligned} \min_{\mathbf{Q}_n} \quad & \sum_{i=1}^{K_n} \tilde{E}_{i_n}(\mathbf{Q}_{i_n}; \mathbf{Q}^\nu) \\ & \tilde{q}_{j_m}(\mathbf{Q}_n; f_{j_m}, \mathbf{Q}^\nu) \overset{\mu_{j_m}^n(f_{j_m}; \mathbf{Q}^\nu)}{\leq} 0, \forall\ j_m \in \mathcal{I}, m \neq n \\ \text{s.t.} \quad & \tilde{q}_{i_n}(\mathbf{Q}_{i_n}; f_{i_n}, \mathbf{Q}^\nu) \overset{\mu_{i_n}^n(f_{i_n}; \mathbf{Q}^\nu)}{\leq} 0, i = 1, \ldots, K_n, \\ & \mathbf{Q}_{i_n} \in \mathcal{Q}_{i_n}, \quad i = 1, \ldots, K_n \end{aligned} \tag{206}$$

where $\mu_{j_m}^n, \mu_{i_n}^n$ are the optimal Lagrangian multipliers associated with the first constraints in (206). Let us denote by $\mathbf{Q}_n^\star(\mathbf{f}; \mathbf{Q}^\nu)$ the unique solution of (206) given $\mathbf{f}$. The optimal partition $\mathbf{f}^\star$ of the shared constraint $\sum_{i_n \in \mathcal{I}} f_{i_n} \leq f_T$ can be found by the femtocloud at the master level by solving the convex problem

$$\begin{aligned} \min_{\mathbf{f}} \quad & P(\mathbf{f},\mathbf{Q}^{\nu}) := \sum_{n=1}^{N_c}\sum_{i=1}^{K_n} \tilde{E}_{i_n}(\mathbf{Q}_{i_n}^{\star}(\mathbf{f},\mathbf{Q}^{\nu});\mathbf{Q}^{\nu}) \\ \text{s.t.} \quad & \mathbf{f} \in \mathcal{T} := \{\mathbf{f} : \mathbf{f} \geq \mathbf{0}, \sum_{i_n \in \mathcal{I}} f_{i_n} \leq f_T\}. \end{aligned} \tag{207}$$

Due to the non-uniqueness of the Lagrangian multipliers, the objective function in (207) is nondifferentiable; problem (207) can be solved using sub gradient methods. A sub gradient of $P(\mathbf{f},\mathbf{Q}^{\nu})$ is given by $\partial_{f_{i_n}} P(\mathbf{f},\mathbf{Q}^{\nu}) = \sum_{j=1}^{N_c} \mu_{i_n}^{j}(f_{i_n};\mathbf{Q}^{\nu}) \frac{\partial \tilde{q}_{i_n}}{\partial f_{i_n}}, \forall i_n \in \mathcal{I}$.

5.2.3.2.2.5.3 Numerical results

Let us consider a MIMO network with $N_c = 2$ cells, where all the transceivers are equipped with $n_T = n_R = 2$ antennas and in each cell there are $K_n = 4$ active users. The other system parameters are set as follows: $f_T = 10^8$, $\mathbf{R}_w = \sigma^2 \mathbf{I}$, with $\mathrm{snr} = P_T/\sigma^2 = 3$ dB, $P_{i_n} = P_T$, $\alpha = 1\mathrm{e}{-5}$; the termination accuracy δ is set to 10^{-2}.

As a numerical example, in Figure 153 we compare the evolution of the network energy $E(\mathbf{Q}^{\nu})$ versus the iteration index ν for the centralized, dual-based and primal-based algorithms. It can be observed that the proposed cell-distributed algorithms approach the same final result of the centralized method. Finally, in Figure 154 we plotted the optimal energy consumption averaged over 100 independent realizations of Rayleigh fading channels versus the ratio $\eta = w_{i_n}/b_{i_n}$ characterizing the application (high values of η indicate computationally intensive applications). We consider a scenario with $N_c = 3$ cells and $K_n = 4$ or 6 mobile users in each cell.

It can be observed that the energy consumption decreases as η increases, i.e. for more demanding applications requiring a higher number of CPU cycles to be executed with respect to the size of the program state to be transmitted. Furthermore, as expected, the global energy consumption increases with the number of active users while, as the number of receiving antennas is higher, the average energy consumption decreases as well, since our computation offloading scheme is able to take advantage of the channel spatial diversity.

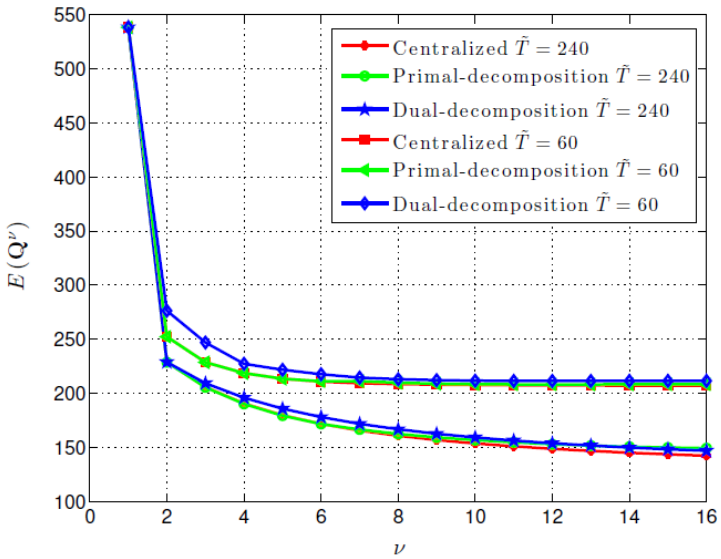


Figure 153. Evolution of the global energy for the centralized, dual-based and primal-based algorithms.

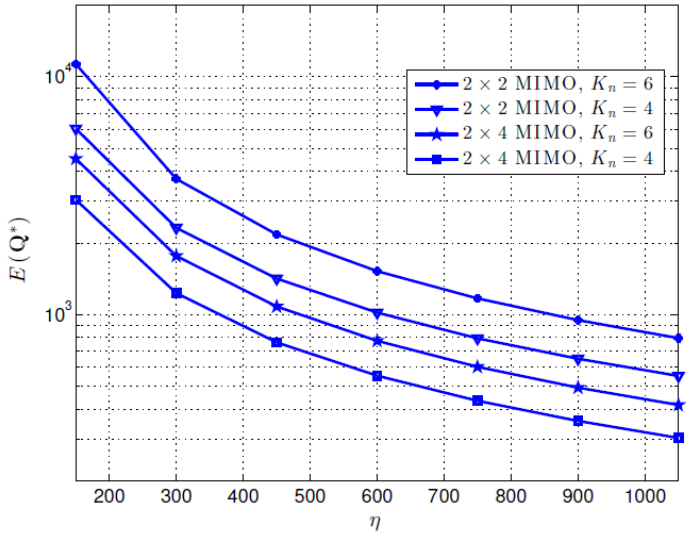


Figure 154. Global average energy vs. η.

5.2.3.2.2.6 *Conclusions*

In this section we have formulated the offloading problem in a multi-cell MIMO femto-cloud network as a joint optimization of the radio and computational resources: the transmit covariance matrices of the MUs as well as the computational resources assigned by the cloud to the MUs are jointly optimized, in order to minimize the overall energy consumption of the MUs, subject to latency constraints. Building on advanced SCA techniques, we developed an iterative algorithm with provable convergence to local optimal solutions of the proposed nonconvex optimization problem. The problem can be performed in a centralized or distributed manner by resorting to multi-levels decomposition methods. Numerical results show that the proposed offloading strategy yields energy savings with respect to a separate optimization of the radio and computational resources.

5.2.3.2.3 Multi-cell MIMO scenario with multiple SCM

5.2.3.2.3.1 *Optimal distributed allocation of communication/computation resources*

In this section we formulate the offloading problem in a much more general context composed of multi-cell with multiple SCM as proposed in [SardBarb2014]. Our goal is to find the optimal offloading strategy to assign each user to a base station and to a cloud, in order to minimize the overall energy consumption, under latency constraints.

Let us consider the network in Figure 155 composed of a set of N_c clouds, N_b small cell access points (SCeNBs in LTE terminology) and K mobile user equipments (MUEs). Each access point n is connected to the cloud provider m through a backhaul link assumed to be a high speed connection (e.g., fiber optics) with latency T_{Bnm}. The k-th MUE and SCeNB n are equipped with n_{T_k} transmit and n_{R_n} receive antennas, respectively. We denote by $\mathcal{I} = \{k: \ k = 1, \ldots, K\}$ the set of all the users. The mobile user decides whether to offload computations or not depending on which strategy entails a lower energy consumption for the mobile device.

In case of offloading, the overall latency experienced by the k-th MUE for accessing the network through the base station n and being served by the cloud m is given by

$$\Delta_{knm} = \Delta_{kn}^{\mathsf{t}} + \Delta_{mk}^{\mathsf{exe}} + \Delta_{knm}^{\mathsf{rx}} + T_{Bnm} \tag{208}$$

where: Δ_{kn}^{t} is the time necessary to send the program state and input (encoded with b_k bits) necessary to transfer the program execution from the k-th MUE to the n-th base station; $\Delta_{mk}^{exe} = \omega_k / f_{mk}$ is the time for the server to execute ω_k CPU cycles, having denoted by f_{mk} the number of CPU cycles/second of the virtual machine running the application of user k over the m-th cloud; Δ_{knm}^{rx} is the time necessary for the server to send the result (encoded in b_k^0 bits) back to the k-th MUE. Finally, the term T_{Bnm} is the delay over the backhaul used to transfer the program state from the n-th base station to the m-th cloud.

Denoting by $\mathbf{Q}_k$ the covariance matrix of the data transmitted by user k and by $\mathbf{Q} := (\mathbf{Q}_k)_{k\in\mathcal{I}}$ the set of all covariance matrices, the maximum achievable rate of MUE k accessing the network through the n-th SCeNB is $r_{kn}(\mathbf{Q}) = \log_2 \det\left(\mathbf{I} + \mathbf{H}_{kn}^H \mathbf{R}_{kn}(\mathbf{Q}_{-k})^{-1} \mathbf{Q}_k \mathbf{H}_{kn}\right)$ where $\mathbf{H}_{kn}$ is the channel matrix between the k th user and the n th base station; the covariance matrix

$$\mathbf{R}_{kn}(\mathbf{Q}_{-k}) = N_0 \mathbf{I} + \sum_{l\in\mathcal{I}, l\neq k} \mathbf{H}_{ln} \mathbf{Q}_l \mathbf{H}_{ln}^H \tag{209}$$

includes both noise and multi-user interference; $\mathbf{Q}_{-k} := (\mathbf{Q}_l)_{\forall l\neq k}$ denotes the tuple of covariance matrices of all users interfering with MUE k (i.e., all covariance matrices except $\mathbf{Q}_k$).

Given each $r_{kn}(\mathbf{Q})$ the time Δ_{kn}^{t} necessary for user k to transmit the b_k input bits of duration T_b to the n-th SCeNB is

$$\Delta_{kn}^{t}(\mathbf{Q}) = \frac{c_k}{r_{kn}(\mathbf{Q})} \tag{210}$$

where $c_k = b_k T_b$.

Our aim is to find the optimal strategy to assign each user to a base station and to a cloud by minimizing the overall energy consumption, under latency constraints.

The degrees of freedom are radio and computational parameters, namely: i) the precoding matrix $\mathbf{Q}_k \in \mathbb{C}^{n_{T_k} \times n_{T_k}}$ for each user; ii) the number f_{mk} of CPU cycles/second for running the application of user k over the m-th cloud; and iii) the rule for assigning each user to a base station and then to a cloud.

The assignment is performed by selecting the binary values $a_{knm} \in \{0,1\}$ for $k = 1,\ldots,K$, $n = 1,\ldots,N_b$, $m = 1,\ldots,N_c$, where the subscripts k, n, and m denote, respectively, user, base station, and cloud indexes.

In principle, a user could be served by multiple base stations, as in cooperative communications, and by multiple clouds. However, this scenario would make the overall computation management much more complicated. For the sake of simplicity, we assume that, at each time frame, each user is served by a single base station and a single cloud. As a consequence, for each k, $a_{knm} = 1$ if user k accesses the network through the n-th BS and it is served by the m-th cloud; all other values $a_{kn'm'}$, with $n' \neq n$ and $m \neq m'$, are set to zero.

Our objective is the minimization of a weighted sum of the energies spent by each mobile terminal:

$$E(\mathbf{Q},\mathbf{a})=\sum_{k=1}^{K}\beta_k E_k(\mathbf{Q},\mathbf{a}_k),\ \text{with}\ E_k(\mathbf{Q},\mathbf{a}_k)=\mathrm{tr}(\mathbf{Q}_k)\sum_{n=1}^{N_b}\sum_{m=1}^{N_c}a_{knm}\Delta_{kn}^{t}(\mathbf{Q}),$$

where $\mathbf{a}=(\mathbf{a}_k)_{k\in\mathcal{I}}$, $\mathbf{a}_k=(a_{knm})_{\forall n,m}$. The coefficients β_k are positive parameters that could be varied dynamically to enforce some sort of fairness among the users.

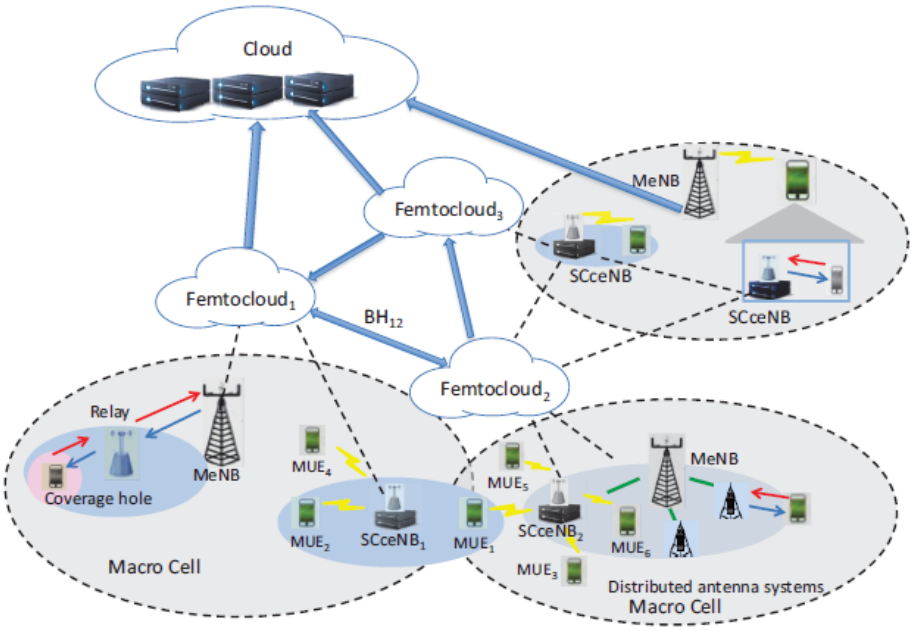


Figure 155. Distributed cloud scenario.

Formally, the optimization problem can be formulated as follows:

$$\min_{\mathbf{Q},\mathbf{f}_s,\mathbf{a}}\quad E(\mathbf{Q},\mathbf{a})=\sum_{k=1}^{K}\beta_k E_k(\mathbf{Q},\mathbf{a}_k),\ \text{subject to:}$$

$$\begin{aligned}
&\text{i)}\ g_{knm}(\mathbf{Q},f_{mk},a_{knm})\le L_k,\forall k,n,m\le L_k,\quad \forall k,n,m\\
&\text{ii)}\ \mathrm{tr}(\mathbf{Q}_k)\le P_T,\quad \mathbf{Q}\succeq\mathbf{0},\forall k=1,\ldots,K,\\
&\text{iii)}\ h_m(\mathbf{f},\mathbf{a})=\sum_{k=1}^{K}\sum_{n=1}^{N_b}a_{knm}f_{mk}\le F_m,\ \forall\, m,\mathbf{f}\ge\mathbf{0}\\
&\text{iv)}\ \sum_{n=1}^{N_b}\sum_{m=1}^{N_c}a_{knm}=1,\ a_{knm}\in\{0,1\},\quad \forall k,n,m
\end{aligned}\qquad (\mathcal{P}_{10})\qquad (211)$$

where $g_{knm}(\mathbf{Q},f_{mk},a_{knm})=a_{knm}\left(\frac{c_k}{r_{kn}(\mathbf{Q})}+\frac{w_k}{f_{mk}}+T_{Bnm}\right)$ and the constraints are:

i) the overall latency for each user k must be lower than the maximum tolerable value L_k;
ii) the total power spent by each user must be lower than its total power budget;
iii) the sum of the computational rates assigned by each server cannot exceed the server computational capability F_m;
iv) each mobile user should be served by one couple base station-cloud.

To drive the solution towards the situation where each user is served by a single base station and a single cloud, we enforce the constraint $\sum_{n=1}^{N_b}\sum_{m=1}^{N_c} a_{knm} = 1$, for each k . For simplicity we have incorporated the term Δ_{knm}^{rx} in the latency limit L_k.

Unfortunately, problem $\mathcal{P}_{10}$ is NP-hard in general, even if one focuses only on the optimization of the covariance matrices $\mathbf{Q}$ and CPU cycles $\mathbf{f}$. Things are even more complicated because of the integer nature of the coefficients a_{knm}. Motivated by the difficulties of handling an NP-hard problem, possibly in a scenario with a large number of users, we propose a suboptimal optimization strategy, based on a specifically tailored successive convex approximation, able to achieve high-quality solutions for problem $\mathcal{P}_{10}$. Let us start by using the popular relaxation of the integer constraints, enabling the binary variables a_{knm} to belong to the following convex set

$$\mathcal{A} = \left\{ (\mathbf{a}_k)_{k\in\mathcal{I}} : a_{knm} \in [0,1],\ \sum_{n=1}^{N_b}\sum_{m=1}^{N_c} a_{knm} = 1, \forall k,n,m \right\}.$$

Then, we can write the relaxed version of $\mathcal{P}_{10}$ as

$$\begin{aligned} \min_{\mathbf{Q},\mathbf{f},\mathbf{a}} \quad & E(\mathbf{Q},\mathbf{a}) \\ \text{s.t.} \quad & g_{knm}(\mathbf{Q}, f_{mk}, a_{knm}) \le L_k, \quad \forall k,n,m \\ & h_m(\mathbf{f},\mathbf{a}) \le F_m, \forall\, m,\ \mathbf{f} \ge \mathbf{0}, \\ & \mathbf{Q} \in \mathcal{Q}, \mathbf{a} \in \mathcal{A}. \end{aligned} \tag{212}$$

We denote by $\mathcal{X}$ the feasible set of the optimization problem in (212). Even by relaxing the binary variables $\mathbf{a}$, problem in (212) is still non-convex, since the objective function and the constraints g_{knm} and h_m are not convex. In what follows, we exploit the structure of (212) and building on some recent advances on Successive Convex Approximation (SCA) techniques, as proposed in [ScutariFacch14, ScutFacchLamp], we devise an efficient iterative approximation algorithm converging to a local optimal solution of (212).

5.2.3.2.3.1.1 Algorithmic design

To solve the non-convex problem (212) efficiently, we adopt an SCA-based algorithm where the original problem is replaced by a sequence of strongly convex problems. To do this, we need to find, at each iteration, a suitable convex approximation of the nonconvex objective function $E(\mathbf{Q},\mathbf{a})$ and the constraints $g_{knm}(\mathbf{Q}, f_{mk}, a_{knm})$, which are preliminarily discussed next.

Approximant of $E(\mathbf{Q},\mathbf{a})$

Let $\mathbf{Z} = (\mathbf{Q},\mathbf{f},\mathbf{a})$, $\mathbf{Z}^\nu = (\mathbf{Q}^\nu,\mathbf{f}^\nu,\mathbf{a}^\nu)$ and $\mathbf{Z}_k = (\mathbf{Q}_k,\mathbf{f}_k,\mathbf{a}_k)$ with $\mathbf{f}_k = (f_{mk})_{m=1}^{N_c}$. Following [ScutFacchLamp] the main idea is to approximate around the current (feasible) iterate $\mathbf{Z}^\nu \in \mathcal{X}$ the original nonconvex nonseparable objective function $E(\mathbf{Q},\mathbf{a})$ with a strongly convex function, say $\tilde{E}(\mathbf{Z};\mathbf{Z}^\nu)$, that is separable in the MUEs' variables and has the same first order behaviour of $E(\mathbf{Q},\mathbf{a})$ at $\mathbf{Z}^\nu$.

To motivate our choice observe preliminary that i) for any given $\mathbf{Q}_{-k}=\mathbf{Q}_{-k}^{\nu}$ each term $E_k(\mathbf{Q}_k,\mathbf{a}_k;\mathbf{Q}_{-k}^{\nu})=\sum_n\sum_m a_{knm}\mathrm{tr}(\mathbf{Q}_k)\Delta_{kn}^t(\mathbf{Q}_k;\mathbf{Q}_{-k}^{\nu})$ is biconvex in $\mathbf{Q}_k$ and convex in $\mathbf{a}_k$ given $\mathbf{Q}^{\nu}$; and ii) the other terms of the sum $\sum_{j\neq k}E_j(\mathbf{Q}_k;\mathbf{a}_j^{\nu},\mathbf{Q}_{-k}^{\nu})$ are not convex in $\mathbf{Q}_k$. Thanks to this structure a convex approximation of $E(\mathbf{Q},\mathbf{a})$ can be obtained for each MUE k by convexifying the term $E_k(\mathbf{Q}_k,\mathbf{a}_k;\mathbf{Q}_{-k}^{\nu})$ and by linearizing the non-convex part $\sum_{j\neq k}E_j(\mathbf{Q}_k;\mathbf{a}_j^{\nu},\mathbf{Q}_{-k}^{\nu})$.

More formally, for each user k we introduce the ``approximation" function $\tilde{E}_k(\mathbf{Z}_k;\mathbf{Z}^{\nu})$ defined as:

$$\tilde{E}_k(\mathbf{Z}_k;\mathbf{Z}^{\nu})=c_k\cdot\sum_{n=1}^{N_b}\sum_{m=1}^{N_c}\left\{\frac{\mathrm{tr}(\mathbf{Q}_k)a_{knm}^{\nu}}{r_{kn}(\mathbf{Q}_k^{\nu},\mathbf{Q}_{-k}^{\nu})}+\frac{a_{knm}\cdot\mathrm{tr}(\mathbf{Q}_k^{\nu})}{r_{kn}(\mathbf{Q}_k^{\nu},\mathbf{Q}_{-k}^{\nu})}+\frac{a_{knm}^{\nu}\cdot\mathrm{tr}(\mathbf{Q}_k^{\nu})}{r_{kn}(\mathbf{Q}_k,\mathbf{Q}_{-k}^{\nu})}\right\}+\sum_{j\in\mathcal{I},j\neq k}\left\langle\nabla_{\mathbf{Q}_k^*}E_j(\mathbf{Z}^{\nu}),\mathbf{Q}_k-\mathbf{Q}_k^{\nu}\right\rangle \quad (213)$$
$$+\tau_k\left\|\mathbf{Q}_k-\mathbf{Q}_k^{\nu}\right\|^2+\tau_f\left\|\mathbf{f}_k-\mathbf{f}_k^{\nu}\right\|^2+\tau_a\left\|\mathbf{a}_k-\mathbf{a}_k^{\nu}\right\|^2$$

where $\langle\mathbf{A},\mathbf{B}\rangle:=\mathrm{tr}(\mathbf{A}^H\mathbf{B})$ and $\nabla_{\mathbf{Q}_k^*}E_j(\mathbf{Z})$ denotes the conjugate gradient of $E_j(\mathbf{Z})$ with respect to $\mathbf{Q}_k$ evaluated at $\mathbf{Z}^{\nu}$. Based on each $\tilde{E}_k(\mathbf{Z}_k;\mathbf{Z}^{\nu})$, given $\mathbf{Z}^{\nu}\succeq\mathbf{0}$, we can now define the candidate sum-energy approximation $\tilde{E}(\mathbf{Z};\mathbf{Z}^{\nu})$ as

$$\tilde{E}(\mathbf{Z};\mathbf{Z}^{\nu})=\sum_{k=1}^{K}\beta_k\tilde{E}_k(\mathbf{Z}_k;\mathbf{Z}^{\nu}). \quad (214)$$

Note that this function enjoys some desirable properties such as i) it has the same first-order behavior of the original nonconvex function $E(\mathbf{Q},\mathbf{a})$; ii) it is separable in the users variables $\mathbf{Z}_k$ and iii) it is uniformly strongly convex on $\mathcal{X}$ (see [ScutariFacch14] for further details).

Inner convexification of the constraints

Let us introduce in this section an inner convex approximation of the constraints $g_{knm}(\mathbf{Q},f_{mk},a_{knm})$ around $\mathbf{Q}^{\nu}\in\mathcal{X}$, denoted by $\tilde{g}_{knm}(\mathbf{Q},f_{mk},a_{knm};\mathbf{Q}_{-k}^{\nu})$. To do so, let us exploit first the concave-convex structure of the rate functions $r_{kn}(\mathbf{Q})$

$$r_{kn}(\mathbf{Q})=r_{kn}^{+}(\mathbf{Q})+r_{kn}^{-}(\mathbf{Q}_{-k}), \quad (215)$$

with

$$\begin{aligned}r_{kn}^{+}(\mathbf{Q})&=\log_2\det\left(\mathbf{R}_{kn}(\mathbf{Q}_{-k})+\mathbf{H}_{kn}\mathbf{Q}_k\mathbf{H}_{kn}^H\right)\\ r_{kn}^{-}(\mathbf{Q}_{-k})&=-\log_2\det\left(\mathbf{R}_{kn}(\mathbf{Q}_{-k})\right)\end{aligned} \quad (216)$$

The constraints $g_{knm}(\mathbf{Q},f_{mk},a_{knm})$ in (212) can be rewritten as

$$g_{knm}(\mathbf{Q},f_{mk},a_{knm})=-r_{kn}^{+}(\mathbf{Q})-r_{kn}^{-}(\mathbf{Q}_{-k})+q_{knm}(f_{mk},a_{knm}) \quad (217)$$

where we introduce the convex function $q_{knm}(f_{mk}, a_{knm}) = \dfrac{c_k a_{knm} f_{mk}}{f_{mk}(L_k - T_{Bnm} a_{knm}) - \omega_k a_{knm}}$.

By retaining the convex part and linearizing the concave term $-r_{kn}^{-}(\mathbf{Q}_{-k})$ we obtain the inner convex approximation:

$$\tilde{g}_{knm}(\mathbf{Q}, f_{mk}, a_{knm}; \mathbf{Q}_{-k}^{\nu}) = -r_{kn}^{+}(\mathbf{Q}) - r_{kn}^{-}(\mathbf{Q}_{-k}^{\nu}) - \sum_{j \in \mathcal{I}, j \neq k} \left\langle \nabla_{\mathbf{Q}_j^*} r_{kn}^{0}(\mathbf{Q}_{-k}^{\nu}), \mathbf{Q}_j - \mathbf{Q}_j^{\nu} \right\rangle + q_{knm}(f_{mk}, a_{knm}) \quad (218)$$

with $\nabla_{\mathbf{Q}_j^*} r_{kn}^{0}(\mathbf{Q}_{-k}^{\nu}) = -\mathbf{H}_{jn}^{H} \mathbf{R}_{kn}(\mathbf{Q}_{-k}^{\nu})^{-1} \mathbf{H}_{jn}$.

Finally the non-convex constraint $h_m(\mathbf{f}, \mathbf{a})$ may be replaced by the following convex approximation

$$\tilde{h}_m(\mathbf{f}, \mathbf{a}; \mathbf{Z}^{\nu}) = \sum_{k=1}^{K} \sum_{n=1}^{N_b} \left(a_{knm}^{\nu} f_{mk} + a_{knm} f_{mk}^{\nu} \right).$$

Inner SCA algorithm

We can now introduce the proposed convex approximation of the nonconvex problem (212) which consists in replacing the nonconvex objective function $E(\mathbf{Q}, \mathbf{a})$ and the non-convex constraints g_{knm}, h_m with their convex approximations. More formally, given the feasible point $\mathbf{Z}^{\nu} \in \mathcal{X}$, we have

$$\hat{\mathbf{Z}}(\mathbf{Z}^{\nu}) \triangleq \operatorname{argmin}_{\mathbf{Q},\mathbf{f},\mathbf{a}} \quad \tilde{E}(\mathbf{Q}, \mathbf{f}, \mathbf{a}; \mathbf{Z}^{\nu})$$

$$\text{s.t.} \quad \begin{aligned} & \tilde{g}_{knm}(\mathbf{Q}, f_{mk}, a_{knm}; \mathbf{Q}_{-k}^{\nu}) \leq 0, \forall k, n, m \\ & \tilde{h}_m(\mathbf{f}, \mathbf{a}; \mathbf{Z}^{\nu}) \leq F_m, \forall m \\ & f_{mk} \geq 0, \forall k, m \\ & \mathbf{Q}_k \in \mathcal{Q}_k, \quad \forall k \in \mathcal{I}, \quad \mathbf{a} \in \mathcal{A} \end{aligned} \quad (219)$$

where we denoted by $\hat{\mathbf{Z}}(\mathbf{Z}^{\nu}) = (\hat{\mathbf{Q}}(\mathbf{Z}^{\nu}), \hat{\mathbf{f}}(\mathbf{Z}^{\nu}), \hat{\mathbf{a}}(\mathbf{Z}^{\nu}))$ the unique solution of the strongly convex optimization problem (219).

The proposed solution method consists in solving iteratively problem, starting from a feasible $\mathbf{Z}^0$. The formal description of the algorithm is given in Algorithm IV below. Note that in Step 2 of the algorithm we allow a memory in the update of the iterate $\mathbf{Z}^{\nu} = (\mathbf{Q}^{\nu}, \mathbf{f}^{\nu}, \mathbf{a}^{\nu})$. A practical termination criterion in Step 1 is to stop the iterates when $|E(\mathbf{Z}^{\nu+1}) - E(\mathbf{Z}^{\nu})| \leq \delta$, where $\delta > 0$ is the prescribed accuracy.

Inner SCA Algorithm 4 for (219)
Data: $\mathbf{Z}^0=(\mathbf{Q}^0,\mathbf{f}^0,\mathbf{a}^0)\in\mathcal{X}$; $\{\gamma^\nu\}_\nu\in(0,1]$; $\tau_k,\tau_f>0$; $\tau_a>0$. Set $\nu=0$.
(S.1): If $\mathbf{Z}^\nu$ satisfies a suitable termination criterion, STOP
(S.2): Compute $\hat{\mathbf{Z}}(\mathbf{Z}^\nu)=(\hat{\mathbf{Q}}(\mathbf{Z}^\nu),\hat{\mathbf{f}}(\mathbf{Z}^\nu),\hat{\mathbf{a}}(\mathbf{Z}^\nu))$ [cf. $\mathcal{P}^\nu$];
(S.3): Set $\mathbf{Z}^{\nu+1}=\mathbf{Z}^\nu+\gamma^\nu\left(\hat{\mathbf{Z}}(\mathbf{Z}^\nu)-\mathbf{Z}^\nu\right)$;
(S.4): $\nu\leftarrow\nu+1$ and go to (S.1).

5.2.4 Total offloading for continuous-execution applications in multi-user environments

In this section, we propose a dynamic packet-based scheduler that decides, at the beginning of each scheduling period, how to split the UL, computational, and DL resources among a set of users. This scheduler is described in detail in [Munoz14c]. It is assumed that a decision to offload completely the applications of such a set of users has been previously taken. We consider that these applications generate continuous flows of packets to be processed remotely. The goal of the scheduler is to minimize the experienced average latency of each user subject to a target energy saving with respect to the case of performing all the computations locally at the UE's. More specifically, the scheduling decisions are taken to minimize the average experienced latency of the worst case user.The energy saving constraint imposes an upper bound on the UL data rate that depends on the target energy saving of the UE and the energy consumption model of the terminal, as it was already described in section 5.2.1.1.2.

In order to accommodate the bursty nature of the packets to be processed, each stage (UL, P, and DL) implements one queue per user, which are served under TDMA. Figure 156 shows the UL, P, and DL queues for this system, where each queue is fed by the outputs coming out from the previous server.

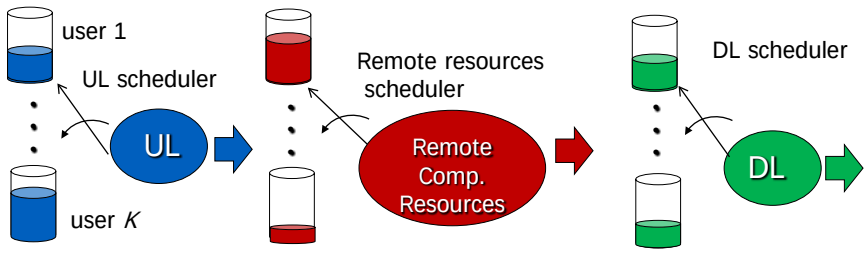


Figure 156. System model with UL, P, DL queues and associated servers.

We denote by $t_{UL,k}[n]$, $t_{P,k}[n]$, and $t_{DL,k}[n]$ the UL, P, and DL times allocated to the *k*-th user in the *n*-th scheduling period. Note that the *n*-th scheduling period of the three queues may be misaligned in time as a result of the different duration of the scheduling periods on each server, denoted by T_{UL}, T_P, and T_{DL} respectively (see Figure 157). As it is shown in Figure 157, the *n*-th scheduling period of the DL phase must follow the *n*-th scheduling period of the P stage that must, in turn, follow the *n*-th scheduling period of the UL phase.

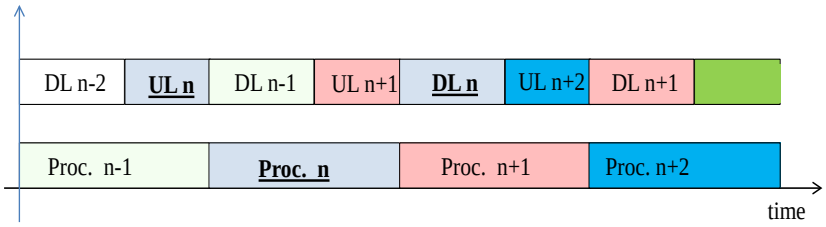


Figure 157. UL, P, DL scheduling periods.

$t_{i,k}[n]$	Times allocated to *k*-th user in the *n*-th scheduling period of the *i*-th stage
T_i	Duration of the scheduling period associated to the i-th stage
$q_{i,k}[n]$	Number of elements in the *k*-th user's *i*-th queue at the beginning of the *n*-th scheduling period
$\overline{q}_{i,k}$	Average number of elements waiting in the *k*-th user's *i*-th queue
$\alpha_{i,k}[n]$	Number of elements arriving at the *k*-th user's *i*-th queue during the *n*-th scheduling period
$\overline{\alpha}_{i,k}$	Average number of input bits, instructions, and output bits arriving at the *k*-th user's *i*-th queue per scheduling period
$s_{i,k}[n]$	Number of bits sent (in UL and DL) or instructions processed (in P) in the allocated time corresponding to the *k*-th user's *i*-th stage during the *n*-th scheduling period
$\overline{d}_{i,k}$	Averaged latency experienced by the *k*-th user at the *i*-th stage
$\hat{\overline{d}}_{i,k}[n]$	Exponential temporal average of the latency experienced by the *k*-th user associated to the *i*-th queue (UL, P, DL) up to the *n*-th scheduling period
$d_{i,k}[n]$	Instantaneous estimate of the latency experienced by the *k*-th user associated to the *i*-th queue (UL, P, DL) at the *n*-th scheduling period
$R_{i,k}[n]$	$R_{i,k}$ rate constraint in the i-th queue. • In the P stage, $R_{P,k}$ is the number of instructions per second that the remote processor can execute, and hence is independent of the user. • "As far as $R_{UL,k}$ is concerned, it depends on the target energy saving of the *k*-th MT as will be stated in (221) - (223). For higher target energy savings, $R_{UL,k}$ will be lower, and therefore the latency will increase." • $R_{DL,k}$ is the maximum DL data rate that it is only determined by the DL channel and the physical layer.

Table 36. Notation for the joint computational and radio scheduling problem.

Considering the notation provided in Table 36. Notation for the joint computational and radio scheduling problem., the problem to be solved at each queue during the *n*-th scheduling period has the same structure and can be formulated as (for $i \in$ {UL, P, DL}):

$$\min_{\{t_{i,k}[n],s_{i,k}[n]\}_{k=1}^{K}} \quad c\left(\left\{\hat{\overline{d}}_{i,k}[n+1]\right\}_{k=1}^{K}\right)$$

$$\text{s.t.} \quad \sum_{k=1}^{K} t_{i,k}[n] \le T_i, \qquad (220)$$

$$s_{i,k}[n] \le q_{i,k}[n],$$

$$s_{i,k}[n] \le t_{i,k}[n] \cdot R_{i,k}.$$

The complete algorithm to solve problem (220) is described in detail in Table 37.

1: **define** $\bar{K} = \{1, 2, \ldots, K\}$, T_i is the scheduling time

2: **calculate:** $d_{\min_{i,k}} = (1-\beta)\hat{\bar{d}}_{i,k}[n] + \beta$

3: **calculate:** $d_{\min_i} = \max_{k \in \bar{K}} \{ d_{\min_{i,k}} \}$

4: **calculate** $x = \sum_{k \in \bar{K}} \frac{1}{R_{i,k}} \left(q_{i,k}[n] - \frac{\bar{\alpha}_{i,k}}{\beta} \left(d_{\min_i} - d_{\min_{i,k}} \right) \right)^+$

5: **if** $x \geq T_i$

6: find d such that $\sum_{k \in \bar{K}} \frac{1}{R_{i,k}} \left(q_{i,k}[n] - \frac{\bar{\alpha}_{i,k}}{\beta} \left(d - d_{\min_{i,k}} \right) \right)^+ = T_i$

7: set $t^*_{i,k}[n] = \frac{1}{R_{i,k}} \left(q_{i,k}[n] - \frac{\bar{\alpha}_{i,k}}{\beta} \left(d - d_{\min_{i,k}} \right) \right)^+ \quad \forall k \in \bar{K}$

8: go to step 15

9: **else**

10: find $k_1 \in \bar{K}$ such that $d_{\min_{i,k_1}} = d_{\min_i}$

11: set $t^*_{i,k_1}[n] = q_{i,k_1}[n] / R_{i,k_1}$

12: set $\bar{K} \leftarrow \bar{K} - \{k_1\}$ and $T_i \leftarrow T_i - q_{i,k_1}[n] / R_{i,k_1}$

13: go to step 3

14: **end if**

15: **end algorithm**

Table 37. Joint computational and radio scheduler procedure.

In order to evaluate the performance of the propose scheduler, consider a detection application that compares an audio signal with size 262.144 kBytes captured by the terminal with *N*=20 patterns by performing cross-correlations. Such application requires around 337 cycles per byte [Johnson07]. We consider that the same number of bits is sent in the UL and the DL. The maximum transmission power for both the MT and the BS is 100 mW. The MT processor corresponds to a commercial model (Nokia N900) that performs 650 Mcycles per Joule when operating at 600 MHz [Miettinen10]. It is assumed that the remote processor is twice faster than the local one (i.e., 1.2 GHz). Finally, the scheduling times are $T_{UL} = T_{DL} = 1$ ms as in LTE [Sesia09], and $T_P = 20$ ms [Noon11]. We consider 5 MT's that offload this application to the same small cell BS. The request arrival to the UL queue follows a Poisson distribution with a mean rate of one audio signal per second. The channels gains are assumed to be $\gamma_{UL,1} = \gamma_{DL,1} = 17.5$ dB for the first user and 19 dB for the other users.

We compare the performance of the algorithm proposed in Table 37, i.e. scheduler (i), with two different algorithms:

- Scheduler (ii): all resources are allocated to the user with more bits/instructions waiting in the queue
- Scheduler (iii): all resources are allocated to the user with more incomplete packets waiting in the queue

We will consider that the offloading is worthy for the MT from an energy point of view if the average energy consumption per second for the *k*-th user, $\bar{e}_k\left(r_{UL,k}\right)$, is lower than a percentage of the average energy per second spent by the MT when performing all the processing locally i.e.

$$\bar{e}_k\left(r_{UL,k}\right) \leq (1-\rho)\bar{\alpha}_{P,k}\varepsilon_0 / T_P \,. \tag{221}$$

In (221), ε_0 is the energy required to process an instruction locally (i.e. the energy consumed per CPU cycle for computation at the MT), and ρ the energy saving that we want to achieve.

The average energy consumption per second for the *k*-th user can be written as follows, assuming that $r_{UL,k}$ remains constant over the averaging period:

$$\bar{e}_k\left(r_{UL,k}\right) = k_{tx,k}\frac{\bar{\alpha}_{UL,k}}{(T_{UL}+T_{DL})r_{UL,k}}\frac{2^{\frac{r_{UL,k}}{\Gamma W}}-1}{\gamma_{UL,k}} + k_{rx,k}\frac{\bar{\alpha}_{DL,k}}{(T_{UL}+T_{DL})}. \tag{222}$$

Note that in (222) we do not include the energy consumption for the processing stage since, when offloading is performed, the computation and execution of the instructions imply an energy spenditure outside the MT.

Combining (221) and (222) results in an upper bound for the UL data rate, equal to inverse function of (222), $\bar{e}_k^{-1}$, evaluated at $(1-\rho)\bar{\alpha}_{P,k}\varepsilon_0 / T_P$:

$$R_{UL,k} = \bar{e}_k^{-1}\left(\frac{(1-\rho)\bar{\alpha}_{P,k}\varepsilon_0}{T_P}\right). \tag{223}$$

Figure 158 shows the actual average latency experienced by the users to process completely a signal as a function of the energy saving factor ρ with respect to the local computation. We consider UEs without microsleep capabilities, i.e. the receiver and transmitter are not switched off during transmissions/receptions. It is important to remark that the latency is the sum of the latencies experienced through the three queues. As expected, we observe that in all cases the average latency to process a signal is higher as the target energy saving increases. For a particular energy saving, we can see that the average latency obtained with the proposed algorithm (i) is significantly lower than the obtained with the other ones, especially if we compare it with (ii), where the improvement is around 35%. Note also that for a given target latency (for instance 350 ms), the energy saving that it can achieved with the proposed scheduler, i.e. scheduler (i), is much higher than with the algorithms (ii) and (iii). Notice that the propose algorithm obtains good results even if we force a high reduction in the energy spending of the offloading process with respect to the case of performing all the computations locally.

Figure 159 shows the empirical cumulative density function (CDF) of the latency per signal obtained with the three algorithms for a specific energy saving (40%). It can be observed that the proposed algorithm shows a better performance not only in terms of average latency, but also in terms of jitter. Note also that, although user 1 has the worst channel, its experienced latency is closer to the other users' latencies with our algorithm than with the other two strategies.

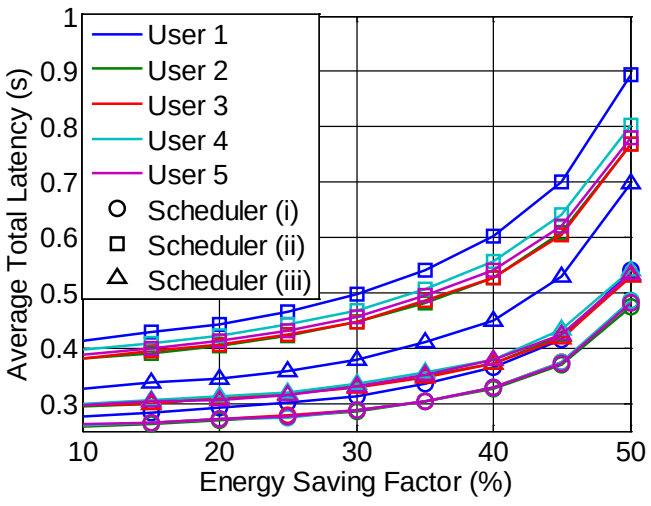


Figure 158. Actual average total latency per signal versus the target energy saving factor ρ.

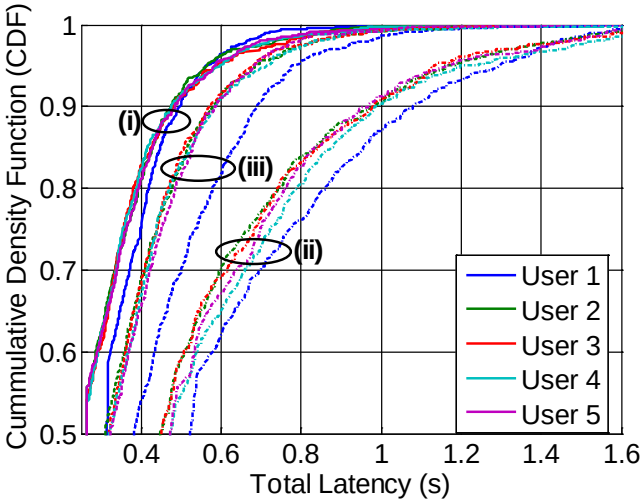


Figure 159. CDF of the actual average total latency per signal for an energy saving of 40%.

5.2.5 Offloading based on a multi-parameter sequential algorithm approach

In this section, we consider another alternative type of offloading decision process aiming to introduce a multitude of parameters in the decision process while keeping it simple to implement. To this end, we adopt a sequential approach where decisions are taken sequentially according to a decision tree. We propose indeed to approach multi-parameters optimization with a multi-fold task classification. We define successive and nested classifications for tasks at the mobile handset. We propose a multi-parameters classification. In this work, we adapt the offloading decision to the current state of the system. Requested tasks that are offloadable are classified and buffered depending on the criticality of their latency constraints. Time critical tasks are to be executed immediately, either locally or at the server. According to the application requirements and mobile available resources (computational capacity, memory space, battery life), the offloading decision is made. In order to keep high user QoE, time critical tasks that should be offloaded are always offloaded, regardless the offloading costs. Remaining offloadable tasks are allowed to be offloaded if the communication offloading cost is low. Using this approach, first, the offloading decision process complexity is reduced while dependency on a variety of parameters is considered through successive and nested tasks classifications. Second, mobile energy consumption is reduced by offloading tasks depending on time criticality, handset available resources, and channel conditions.

We consider a system with a user served by either a SCeNBce, within a distance d. We consider uplink connection, between MUE and the serving base station, with a bandwidth B. Instantaneous uplink bit rate is maximized based on adaptive modulation and coding scheme (AMC). A parameter Γ in the channel model indicates the SNR margin to guarantee the minimum error rate. We adopt a Rayleigh channel model with path loss exponent β, noise power N_0, and fading channel coefficient h_k. We assume perfect estimation of the channel coefficients hk. The channel is assumed constant for a whole transmission period, with a coherence time T_c. We consider a constant transmission power P_{Tx} through the defined channel. The maximum information rate that can be achieved through this channel is calculated using the following equation:

$$R_{Tx} = B \times log(1 + aP_{Tx}) \text{ where } a = \frac{|h_k|^2}{\Gamma d^{\beta} N_0} \quad (224)$$

Indeed, R_{Tx} determines the maximum number of tasks that can be transmitted through the wireless link in one time slot. Each MUE is characterized by a set of parameters summarized as follows:

- F : CPU computational capacity [instruction/sec]
- Tot_E : Total energy capacity of the mobile handset [Wh]

- EPI : Energy consumption per instruction [J/instruction]
- M_{av} : Current amount of available memory [MB]
- B_{lev} : Availablet battery level percentage

Applications are launched by the user at the mobile handset. The applications activity is modeled as a Poisson process with a rate λ, where λ represents the number of launched applications in a time window T_w. We consider that an application call generates a burst of tasks to be computed. Each generated task is a set of ω instructions that has to be executed with a required memory m, and a maximum latency L_{max}. A parameter ρ indicates if the task is offloadable (ρ = 1) or not (ρ = 0). The percentage of tasks that cannot be offloaded is defined by a parameter ano. In case the task is offloadable, a parameter N indicates the number of bits to send to the femtocell if the task is decided to be offloaded. In our study, we compute for each task its energy consumption. For the tasks that are computed locally, the mobile handset energy consumption is evaluated as the product of the number of instructions to be executed and the energy consumption per instruction:

$$E_{local}\ [\mathrm{J}] = \omega \times EPI\ [\mathrm{J/Inst}] \tag{225}$$

For the tasks that are offloaded, the energy consumption of the mobile handset physical components is evaluated based on the mobile user LTE power consumption model proposed by Jensen et al. in [Jensen12]. In this model, the consumed energy for transmission is evaluated as:

$$E_{offloading} = P_{Tx,C} + P_{Tx,BB} + P_{Tx,RF} + P_{con} \tag{226}$$

$P_{Tx,C}$: Power consumption of active transmission chain.
$P_{Tx,BB}$: Power consumption of the baseband (BB) components. It depends on the uplink data rate R_{Tx} by the following equation:

$$P_{Tx,BB}[\mathrm{mW}] = 34.5 + 0.87 R_{Tx}[\mathrm{Mbits/s}] \tag{227}$$

$P_{Tx,RF}$: Power consumption of radio RF components. It depends on the transmission power S_{Tx} by the following equation:

$$P_{Tx,RF}[\mathrm{mW}] = -943 + 117 S_{Tx}[\mathrm{dBm}] \tag{228}$$

P_{con} : Average power consumption in connected mode. It is equal to 1.35W, according to [Jensen12].

What we propose is to perform a series of classifications that embrace a multitude of parameters, without including them in a complex optimization problem. Each classification is based on a decision affecting parameter. At each time slot, the algorithm of mobile application offloading decision is run on the set of tasks generated by the launched applications. The proposed algorithm is summarized as follows:

Step 1: Not all tasks are offloadable. For example, some tasks are dependent of mobile handset hardware or peripherals, others may depend on input information, etc. For this reason, the first classification applied on the set of all generated tasks distinguishes offloadable tasks from not offloadable tasks. All tasks that have the possibility to be offloaded (ρ = 1) are assembled in a first set called "*Off*". Tasks that cannot be offloaded by characteristics definition (ρ = 0), are assembled in another buffer called "*NOff*".

Step 2: In this second step, tasks of both sets, *Off* and *NOff*, are classified as urgent and not urgent based on latency constraints. Latency is considered as the most important constraint to respect since it directly affects user QoE. For the Off set, a task is classified as urgent if $L < L_{max} L_{th}$. Where L_{th} is a

threshold on the latency deadline and L is the remaining task latency. This classification divides the *Off* set into two distinct parts: the urgent offloadable tasks set referred to as *"OffUrg"* and the not urgent offloadable set referred to as *"OffNUrg"*. Not offloadable tasks should all be computed locally at the mobile handset. However, the handset resources are limited in computational capacity as well as in memory. Therefore, priority should be given to the tasks with tighter latency constraints. The algorithm checks if all the tasks latency constraints can be met by assigning to each task in NOff an equal part of the total available local computation capacity (MHcap). Tasks that cannot meet latency constraints in these conditions are given local computation priority. The decision criteria can be formulated as:

$$L < \frac{MH_{cap}}{|NOff|} + \varepsilon \tag{229}$$

Tasks that verify this condition are set as locally urgent and are added to the not offloadable urgent set referred to as *NOffUrg*, and are computed locally. The remaining tasks are added to the set referred to as *NOffNUrg*.

Step 3: The third classification is applied on the offloadable urgent tasks represented by the *OffUrg* set. Basically, each of these tasks is not prevented from being computed locally ($\rho \neq 0$). However, local computation of some of these tasks may not be possible due to lack in available resources at the mobile handsets. In this case, these tasks have to be offloaded. This classification aims at selecting the tasks which offloading is necessary. This decision is based on a set of selected tests. If the mobile handset battery level is lower than a predefined threshold, $\mathrm{BLev}_{\mathrm{off}}$, all tasks are classified as "should be offloaded" referred to as "*SOffUrg*". Otherwise, a task is added to the "*SOffUrg*" set if its local computation memory requirements exceed the allowed usage of available handset memory percentage, $\mathrm{MEM}_{\mathrm{off}}$. Then, a second classification is applied, and tasks that require a computational capacity greater than a predefined percentage of the total locally available capacity, $\mathrm{CAP}_{\mathrm{off}}$, are added to "*SOffUrg*". A last classification is applied on the remaining tasks (tasks with acceptable memory and computational capacity requirements) to check their local computation battery consumption. If the task computation consumes more than a predefined percentage of the available battery level, $\mathrm{BAT}_{\mathrm{off}}$, it is added to "*SOffUrg*". The offloading thresholds $\mathrm{BLev}_{\mathrm{off}}$, $\mathrm{MEM}_{\mathrm{off}}$, $\mathrm{CAP}_{\mathrm{off}}$, and $\mathrm{BAT}_{\mathrm{off}}$ are parameters that could be defined by the user through its mobile equipment operating system. The remaining tasks form a set that we refer to as "*COffUrg*" in reference to offloadable urgent tasks that could be either offloaded or computed locally.

Step 4: In this step, we verify, for each of *COffUrg* tasks, if it could be computed locally applying the same test series in step 3. If the task is allowed to be computed locally, then we are facing two options: offloading or local computation. In this case, we compare Elocal , the consumed energy in case of local computation of this task using Equation (225) and $\mathrm{E}_{\mathrm{offloading}}$, the consumed energy at the mobile handset in case of offloading using Equation (226). If $\mathrm{E}_{\mathrm{local}} < \mathrm{E}_{\mathrm{offloading}}$, the task is computed locally, otherwise, the task is offloaded.

Step 5: The channel capacity at each instance can be calculated using (224). In the case where the channel capacity does not allow the transmission of all the tasks in "*SOffUrg*", tasks with lowest latencies are prioritized. The mobile will offload as many tasks as possible, and the remaining tasks will be differed to be sent at the next time slot. In the other case, where the number of bits to be sent is less than what the channel capacity offers, the channel conditions are examined: the channel coefficient is compared to the average channel coefficient calculated and updated over time. If the current channel realization is above this average, it is considered that the channel is in a relatively "good" state, and the maximum possible number of non-urgent tasks is offloaded. Priority is given to tasks in *OffNUrg* that have lower latencies. Offloading non urgent offloadable tasks can be seen not only as an opportunistic utilization of the radio link, but also as a tool to alleviate the system in future time slots. This would prevent the system from applying costly offloading when channel conditions are not favorable. Moreover, we know the tasks to be computed locally (set *OffUrg*). Therefore, we

know the remaining mobile resources. Applying the same conditions in step 3 to the set *NOffNUrg*, we assign more tasks to be computed locally. Remaining tasks are deferred to be served in the next time slots.

To investigate the offloading efficiency achieved with the proposed algorithm, we adapt, for numerical evaluation, the same parameters as the described system model, considering a single user served by a SCeNBce, within a distance $d = 5$m. The considered uplink bandwidth is of $B = 20$MHz, the path loss coefficient $\beta = 5$, the noise power as $N_0 = 10^{-3}$, and the channel coherence time $T_c = 10$ms. We consider a transmission power of $P_{Tx} = 0.2$W. The simulations are performed over approximately 2.10^5 channel instances per hour. The mobile handset EPI is estimated between 17nJ and 19nJ, its total energy capacity between 4Wh and 8Wh, its available memory of $M_{av} = 5$MB. We consider application calls that generate a burst of tasks (10 tasks). Each task is characterized by a required memory $1\text{KB} \leq m \leq 1\text{MB}$, a number of bits to be offloaded $1\text{KB} \leq N \leq 20\text{KB}$, a latency constraint $90\text{ms} \leq L \leq 300\text{ms}$ and a number of instructions to be executed that varies with each task and different scenario assumptions. The application arrival modeled as a Poisson distribution with a rate $\lambda = 2$ for $T_w = 10$ms, i.e. the user asks for an average of 2 applications each 10ms. The percentage of tasks that cannot be offloaded is $\alpha_{no} = 40\%$. The constraints thresholds are set as: $BLev_{off} = 20\%$, $BAT_{off} = 30\%$, $MEM_{off} = 70\%$ and $CAP_{off} = 50\%$.

According to [6], the decision must be to never offload when large amount of communication is needed with relatively small amount of computation, and must be to always offload when large amount of computations is needed with relatively small amount of communication. In neither case, the decision depends on the available bandwidth. To benchmark the proposed algorithm, we run simulations for different scenarios that represent the three cases in [6]. The adopted scenarios vary in channel conditions and amounts of computation per task. These scenarios are a combination of parameters $(\alpha_{min}, \alpha_{mix}, \alpha_{max})$ representing respectively a low, random and good channel coefficient average, and $(TC_{min}, TC_{mix}, TC_{max})$ representing respectively a low, random and high amounts of computation for each task. In this paper, we show simulation results for the following representative scenarios:

- **Scenario 1:** good channel conditions (α_{max}) and large amounts of computation per task (TC_{max}). For good channel conditions, we select the best 20% of the generated channel instances.
- **Scenario 2:** bad channel conditions (α_{min}) and small amounts of computation per task (TC_{min}). For bad channel conditions, we select the worst 20% of the generated channel instances.
- **Scenario 3:** random channel conditions (α_{mix}) and mixed amounts of computation per task (TC_{mix}). For random channel conditions, no selection over the generated channel instances is made.
- In order to evaluate the algorithm performance, we compare to the following reference algorithms:
- **No Offloading** Tasks are never offloaded.
- **Total Offloading** Offloadable tasks ($\rho = 1$) are always offloaded.
- **Reference Offloading** Task offloading is based on the offloading energy trade-off between local computation energy cost and offloading cost. This offloading approach is the traditional way to treat the problem.

$$\begin{aligned} E_{local} &\lesseqgtr E_{offloading} \\ E_{UE,loc} &\lesseqgtr E_{UE,off} + E_{femto} \end{aligned} \tag{230}$$

where $E_{UE,loc}$ is the energy spent at the mobile handset for locally computing the requested task. $E_{UE,off}$ is the energy spent at the mobile handset for sending the necessary information for the femtocell where the task will be offloaded. $E_{femto} = NC_f$ is the energy spent at the femtocell for computing the requested tasks. N is the number of instructions to be computed at the femtocell, and Cf

is the computation capacity accorded to the computation of these instructions. This entity depends on various components such as system implementation, femtocell clusterization, CAC (Call Admission Control) at femtocells, etc. In this scenario, two approaches are possible. The first one compares the energy of the whole system in order to decide on offloading. In this case, $E_{femto} \neq 0$. Another possible approach, which is adopted in this paper, is the user centric approach. In this case, the mobile handset searches only for reducing its own energy consumption, and the decision does take into account the whole system energy efficiency. It consists on comparing the energy spent only by the mobile handset for the cases of both offloading and local computation. In this case, $E_{femto} = 0$ and the trade-off equation will be reduced to:

$$E_{UE,loc} \lessgtr E_{UE,off} \tag{231}$$

In Figure 160, Figure 161, and Figure 162, curves with plus marks represent battery level in the case where the proposed algorithm is applied. Curves with point marks represent the case of reference offloading. Circle marked and diamond marked lines represent, respectively, the cases of total offloading and no offloading. Figure 160 shows the mobile handset battery discharge for Scenario 1. In this case, offloading is beneficial. The figure shows that keeping all tasks for local computation costs the most in terms of handset energy. Reference offloading and total offloading share the same results because offloading is less battery consuming than local computing in this scenario and thus the algorithm based on the energy trade-off will always decide to offload the requested tasks. The figure also shows that the proposed algorithm outperforms all other algorithms in terms of handset battery life. In fact, taking advantage of good channel conditions to deal with non urgent tasks prevents the system from having a large amount of urgent tasks to deal with in the near future. Applying the proposed offloading algorithm, the battery extension factor is of 2.29 compared to total offloading, and approximately 1.55 compared to total local computing. We can also see on Figure 163 that the proposed algorithm prevents the system from having CPU memory or capacity outage while respecting latency constraints.

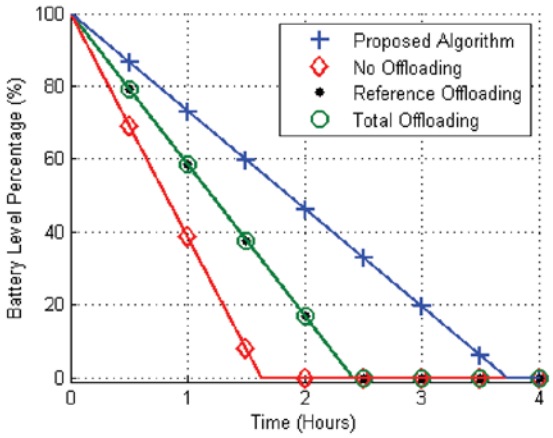


Figure 160. Scenario 1: Mobile battery discharge due to tasks computation/offloading for all considered algorithms.

For Scenario 2, offloading tasks is not beneficial. Figure 161 shows the battery discharge in such conditions. It is clearly shown that the algorithm that does not allow offloading outperforms the total offloading algorithm. In this case, the reference offloading algorithm that is based on the energy trade-off gives results that area close to the no offloading algorithm decisions, which are less energy consuming. The proposed algorithm is more energy consuming than the reference offloading and the no offloading algorithms. This is due to the fact that the proposed algorithm sends offloadable tasks that are assigned as urgent regardless the channel conditions. This affects the energy consumption of the mobile handset, but will guarantee a good user QoE. As it is shown in Figure 163, latency constraints are violated only for the case of total offloading up to 3%, due to bad channel conditions.

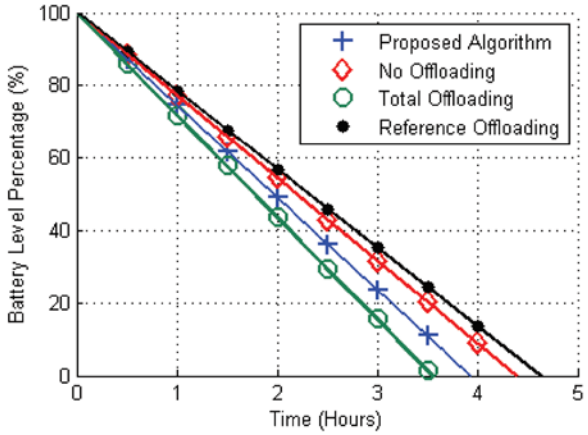


Figure 161. Scenario 2: Mobile battery discharge due to tasks computation/offloading for all considered algorithms.

In Figure 162, results are shown for scenario 3. The figure shows that even in this random case scenario, the proposed algorithm outperforms the reference offloading algorithm. The latter is clearly seen as less energy consuming than the no offloading and the total offloading algorithms. The proposed algorithm prolonged the battery life 1.45 times in these random conditions scenario.

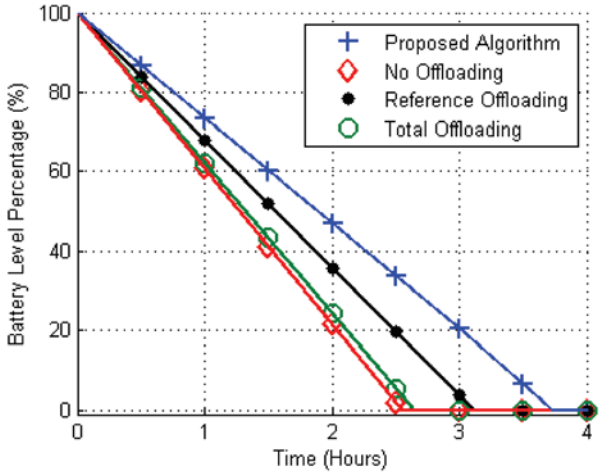


Figure 162. Scenario 3: Mobile battery discharge due to tasks computation/offloading for all considered algorithms.

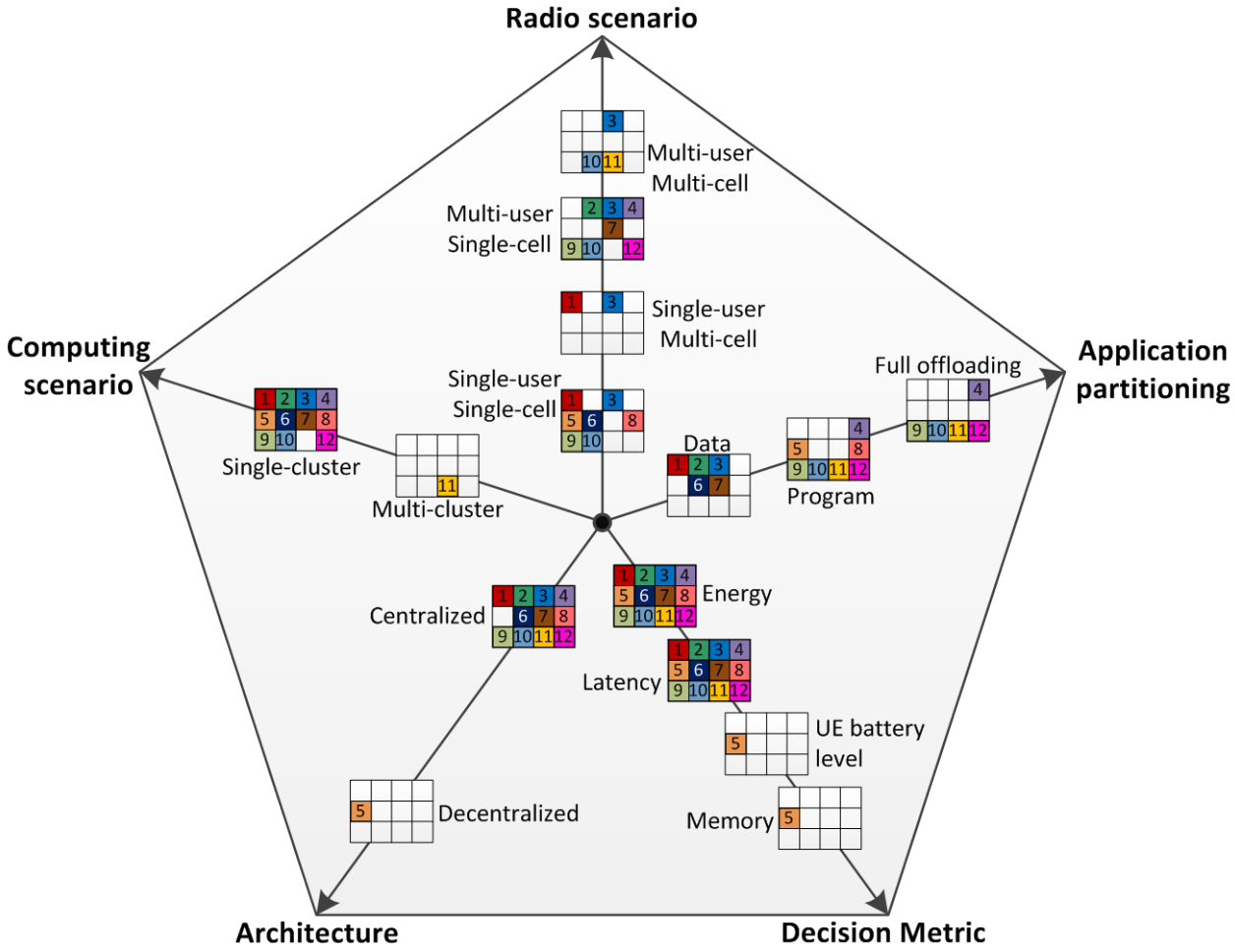
In addition of energy savings at the mobile handset, the proposed algorithm prevents the mobile from suffering of CPU memory overflow and computational capacity outage and allows the user to always have good QoE by respecting each task latency. Those benefits are also validated by simulations (see Figure 163).

Scenario	Algorithm	Battery Life [mn]	CPU Memory Overflow (%)	CPU Capacity Outage (%)	Latency Violation
Scenario 1	A	**223.9**	**0**	**0**	**0**
	B	97.86	2.61	5.02	0
	C	144.77	0.17	4.31	0
	D	144.77	0.17	4.31	0
Scenario 2	A	**235.64**	**0**	**0**	**0**
	B	263.19	1.29	0	0
	C	277.8	0	0	0
	D	231.02	0	0	0
Scenario 3	A	**223.96**	**0**	**0**	**0**
	B	153.09	0.3	0	0
	C	187.09	0.02	0	0
	D	158.87	<0.01	0	3

Figure 163. Simulations results for scenarios 1, 2, and 3. Algorithms: Proposed algorithm (A), No Offloading (B), Reference Offloading (C), Total offloading (D).

5.2.6 Summary of offloading and resource allocation algorithms

All above-mentioned algorithms of offloading decision differ among each other in suitability for different scenarios, type of applications, or architecture, and usage of different metrics for decision. Classification and suitability of individual algorithms for different situtions is depicted in Figure 164.



Legend:

1	Data-oriented offloading: optimization of the tradeoff between the energy consumption at the UE and the latency in the single-user case (Section **5.2.1.1.1**)
2	Data-oriented offloading: optimization of the tradeoff between the energy consumption at the UEs and the latency – extension to multi-user case (Section **5.2.1.1.2**)
3	Data-oriented offloading: optimization of the tradeoff between the energy consumption at the UEs and the latency – extension to multiple VMs (Section **5.2.1.1.3**)
4	Total offloading for continuous-execution applications in multi-user environments (Section **5.2.4**)
5	Multi-parameter sequential algorithm (Section **5.2.6**)
6	Data-oriented offloading: joint optimization of radio resource allocation and computation offloading: minimize energy consumption under delay constraints in the Single-user Single-cell case (Sections **5.2.1.2.2, 5.2.1.2.3, 5.2.1.2.4**)
7	Data-oriented offloading: joint optimization of radio resource allocation and computation offloading: minimize energy consumption under delay constraints in the Multiple-user Single cell case (Sections **5.2.1.2.5**)
8	Program-oriented offloading: joint optimization of energy consumption and code partitioning under latency and power constraints in the single user single-cell case (Section **5.2.2**)
9	Task-oriented offloading: joint optimization of energy consumption and computational rate under latency, power and computational stability constraints in the single and multi-user, single-cell cases (Section **5.2.3.1**)
10	Task-oriented offloading: joint optimization of energy consumption and computational rate under power and latency constraints in MIMO multi-user multi cell, single cluster case (Section **5.2.3.2.2**)
11	Task-oriented offloading: joint optimization of energy consumption and computational rate under power and latency constraints in MIMO multi-user multi-cell, multicluster case (Section **5.2.3.2.3**)
12	Task-oriented offloading: joint optimization of energy consumption and computational rate under power and latency constraints in SISO multi-user single cell case (Section **5.2.3.2.1**)

Figure 164. Classification and overview on suitability of offloading decision algorithms for different scenarios and situations.

All offloading decision algorithms require to collect specific metrics and information about status of networks and its individual elements. List of all parameters for offloading decision together with the list of algorithms exploiting each parameter is presented in Table 38. It can be seen that the radio

channel related parameters are required by all algorithms together with knowledge of energy required for computation and status of computing cells.

Monitored parameter	Exploitation in algorithms
Channel status	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Optimum UL data rate	1, 2, 3, 4, 5
Actual user throughput in UL and DL	1, 2, 3, 4
Bandwidth allocated in UL and DL	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Maximum application latency	1, 2, 3, 4, 5, 8, 10, 11, 12
Application average latency and overflow constraints	6, 7
Time required to do the local processing of the i-th partition	1, 3, 8
Total time required for each configuration	1, 3
Transmission delay	1, 2, 3, 4, 6, 7
Maximum transmission delay	9
UL transmission delay to SCeNBce	1, 2, 3, 4, 6, 7
DL transmission delay from SCeNBce	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12
Processing delay at SCeNBce	1, 2, 3, 4, 6, 7
Latency of VM to each SC	3
UL and DL times and rates allocated	1, 2, 3, 4
Decision time slot duration	6, 7
User buffer size	6, 7
UL packets length in bits (for offloading)	4, 6, 7
DL packets length in bits (for offloading)	4, 6, 7
Number of transmitted packets/bits	1, 2, 3, 4, 6, 7
Number of instructions to be executed	4
Number of new bits generated by the application at each scheduling period	4
Application arrival packets distribution	6, 7
Amount of bits to be sent per offloaded block in both UL and DL	1, 2, 3, 4, 8, 9, 10, 11, 12
Energy consumption of the terminal	1, 2, 3, 4, 5, 8
Energy consumption for each data partition	1, 3
UL energy consumption	1, 2, 3, 4, 6, 7
DL energy consumption	1, 2, 3, 4, 6, 7
Power consumed to process one packet for local processing	6, 7
Target energy saving	2, 4
MUE maximum power budget	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12
UL transmission RF power	1, 2, 3, 4, 5, 6, 7
DL power transmitted at SCeNBce	6, 7
Reception power	1, 2, 3, 4, 6, 7
Waiting power	2, 5, 6, 7
Maximum supported UL rate to achieve a minimum energy saving	2, 4
Number of CPU cycles to be executed	1, 2, 3, 4, 10, 11, 12
Maximum server rate (CPU cycle/sec)	1, 2, 3, 4, 9, 10, 11, 12
Application computational rate (CPU cycle/sec)	9
Frequency of SCeNBce CPU	1, 2, 3, 4, 6, 7
Frequency of UE CPU	1, 3, 6, 7
Optimum partition	1, 3
Effective processing rate in the remote VM	1, 2, 3, 4
Available resources at all VMs	1, 2, 3, 4
MUE battery level	5
MUE available memory	5
MUE computational capacity (instructions/sec)	5

Table 38. List of parameters required for each algorithm to make proper offloading decision.

In Section 5 we have provided several strategies for the offloading process. The number of proposed strategies is high since the scenarios, kind of applications, the user demands, the target quality measure, etc. to be faced may be different. For each specific combination, a focused offloading

strategy is needed. All strategies aim on minimizing the mobile transmit energy spent for computation offloading while meeting the latency constraint imposed by the running application. In spite of the diversity of strategies proposed in the project, we may summarize the results stating that thanks to the offloading process, the system is able to increase the QoS perceived by the user in terms of experienced delay and/or energy spending. Based on performance analysis provided in whole section and considering complexity of individual approaches, data-oriented and task-oriented strategies presented in Section 5.2.1.1.3 (Algorithm 3) and Section 5.2.3.2.1 (Algorithm 12), respectively, are selected as the most suitable approaches recommended for deployment in network and also for deep evaluation and comparison in wide range of scenarios in WP6.

6 MOBILITY ASPECTS

Mobile users can change their serving eNB during movement. In order to guarantee QoS even for SCC services, solutions for mobile users are proposed in this section.

Mobility of the UEs can be classified into low and medium/high (see Figure 165). The low mobility is understood as user's movement inside buildings such as a flat or a house. It means the users is moving with low speed (less than 1m/s) within area covered by one or few SCeNBces. This corresponds to the residential scenario (see [TROPIC_D21]). Some specific spots of the indoor area can be out of coverage of the indoor SCeNBce. These spots are covered by outdoor MeNB or by another SCeNBces. If the UE leaves the indoor area, it is assumed to be in medium or high mobility states (pedestrians or vehicular users). The UEs in large indoor areas (e.g., malls, industrial buildings, stadiums, etc) covered by multiple SCeNBces are assumed to be in the state of medium mobility.

For indoor low mobility UEs, the intention is to keep the UE connected still to the same SCeNBce to avoid redundant hadnovers and complicated selection of path for delivery of computed results back to the UE. This problem is adressed by a control of transmission power of SCeNBces in the UEs' neighborhood. The objective of the power control is to avoid redundant handovers if the UE is moving indoor and keep the UE attached to the same serving SCeNB until all computation results are delivered back to the UE (for more details and performance evaluation, see D322, section 5.3.1). If the UE moves outdoor or if the transmitting power of SCeNBs in UE's neighbornood cannot be change enogh to keep the UE at the same serving SCeNB, handover to another cell is performed. If the handover is completed before the offloading is initiated, common offloading decision procedure (defined in Section 5) is followed. If data is offloaded to the SCC and handover to a new cell is performed during data processing in SCC, we propose two options how to guarantee results delivery to the UE: VM migration and path selection. The first option is based on assumption that the serving SCeNB is also the cell, which performs computation for the UE. In this case, the computation is migrated together with the handover. The path selection does not change place of computation (the same SCeNBces perform computation disregarding handover) but new path is found after computation. Thus, the results are delivered to the UE through path with the shortest delay. Both options are described and analyzed in Sections 6.1 and 6.2.

Handover can be initiated also during offloading. In this case, offloading decision is already done using parameters of radio and backhaul belonging to the former serving cell. In addition, computing cells are already selected by the SCM and computation can be already in progress. In this case, the conditions for offloading decision are not valid and user's QoS can be negatively impaired. However, it is always possible to complete the offloading and computation process considering VM migration and/or path selection. Of course, the same level of QoS as during original offlaoding decision cannot be quaranteed; however, the computation is completed and results are delivered to the UE.

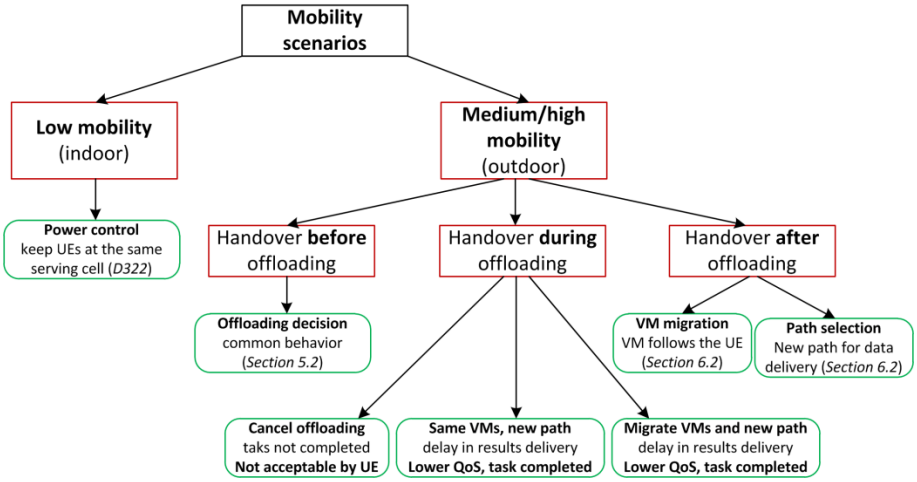


Figure 165. Overview of mobility scenarios and solutions.

6.1 *Path selection for mobile users*

To send data for computation from the UE to the SCeNBce, optimal path has to be selected. For this purpose a new routing protocol needs to be designed. With specifics of the proposed small cell cloud (SCC) concept, new challenges arise as the network is heterogeneous with different communications medium at different part of route as shown in Figure 166, where $\boldsymbol{d_{proc}}$ denotes processing delay of each SCeNBce and $\boldsymbol{c}$ represents capacity of the link. Radio capacity is represented by $\boldsymbol{c^R}$ and backhaul capacity by $\boldsymbol{c^B}$.

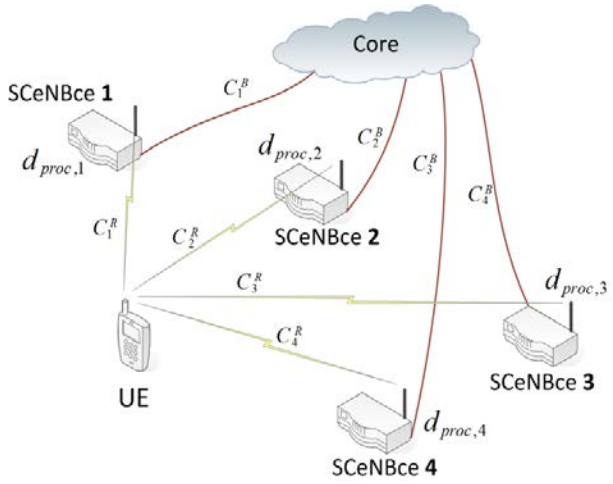


Figure 166. Network topology and definition of parameters required for path selection.

The problem of selection of the most appropriate path for data delivery to the computing cells can be seen as an analogical problem to routing in Wireless Sensor Networks (WSN). Thus, the WSN routing protocols may provide an inspiration how to treat the path selection in the SCC. Of course, mobile network topology does not enable such freedom as conventional WSN but it rather follows hierarchical network structure in the WSN where some nodes are selected as gateways (cluster heads), which relay data to a target destination [Al-Karaki04].

In the SCC, the SCeNBces can be seen as gateway nodes. Each gateway has a fix number of options how to distribute offloaded computation data to computing cells through a fix infrastructure of the mobile network. This infrastructure is represented typically by a wired backhaul and core network of the operator. Therefore, the problem consists in selection of a proper gateway (serving cell) for individual parts of offloaded data. The selected gateway must minimize data transmission delay and energy consumed by the UEs for the transmission. The same problem can be defined also for delivery of computation results back to the UE if the original path is far from optimal, e.g., due to user's movement.

In the WSN, plenty of algorithms have been defined. Basic routing algorithms for the WSN do not consider energy consumption of data delivery or dynamic path update [Al-Karaki04]. In the SCC, the energy is limiting only for radio communication between the UE and the SCeNBces. Also, dynamics of the system is inherent feature of mobile networks. Therefore, energy as well as dynamics must be taken into account. Dynamics of scenario for the WSN is addressed, for example, by the Ad-hoc On-demand Multipath Distance Vector with Dynamic Path Update (AOMDV-DPU) [Marina06]. Additionally to hop count metric, this algorithm selects paths based on the Received Strength Signal Indicator (RSSI). However, this algorithm does not consider transmission energy, which is essential in our case. Similar weakness prevents implementation of Adaptive Multi-metric Ad-Hoc On-Demand Multipath Distance Vector Routing algorithm [Medley13] to the SCC since it routes data based only on RSSI, latency and node occupancy. Moreover, backhaul from the serving cell to the mobile operator's core network is typically wired. In addition, if serving cell selection is based on RSSI, the

same path to the core network would be selected all the time disregarding selection of the SCeNBces for computation. It means that WSN-like approaches cannot be simply applied to our problem.

Designed path selection algorithm should take into account UE's limited energy resources, radio and backhaul conditions, and UE requirements on maximal possible delay using selected path to guarantee Quality of Service (QoS). In the existing approaches exploited for delivery of offloaded data from the UE to the computing SCeNBces in the SCC, the data is always delivered to the computing cells through the serving cell [Lobillo2014], [Valerio14]. It means the UE is attached still to the same cell during data delivery. Then, the serving cell distributes data through operator's core networks to the computing cells. This approach can be efficient if both radio channel between the UE and its serving cell and backhaul connection of the serving as well as all computing cells are of a sufficient throughput. Otherwise, limitation at any part of the communication chain leads to degradation in the overall delay of computation offloading.

6.1.1 **Proposed path selection algorithm**

In common approach, only the path from the UE to the computing cell through the serving cell is used. To overcome potential delay due to the limited throughput of a backhaul, an opportunity to transfer data also via neighboring cells can be exploited. In this case, individual parts of the data for computation are delivered to specific computing cells through the neighbors, which offer the lowest delay of the transmission over both radio and backhaul. Note that for each computing cell, data can be delivered through different serving cell. This implies a need for performing handover during communication. Therefore, the proposed algorithm is labeled as Path Selection with Handover (PSwH) while the existing scheme using only serving is denoted as Serving Only (SO).

The path selection algorithm suitable for the problem combines cost of data transmission over wired and wireless links with energy consumed by the UE for transmission over the radio in order to satisfy delay constraint *Treq*. Therefore, any delay higher than the required one is considered as unsuitable. If at least one available path fulfills *Treq*, all paths with delay exceeding *Treq* are dropped and are not considered in the path selection. If no path is able to provide delay lower than *Treq*, path with the lowest delay is selected.

The path selection is based on weighting of path delay (*D*) and energy (*E*) consumed by UE's transmission over the radio part of the path. In order to weight both metrics, normalization is used as follows:

$$D_i^N = \frac{D_i}{\max\{D_1, D_2, ..., D_p\}} \tag{232}$$

$$E_i^N = \frac{E_i}{\max\{E_1, E_2, ..., E_p\}} \tag{233}$$

where D_i (E_i) is the delay (energy) of the *i*-th path and *p* is the number of possible paths from the UE to the computing cell. The path selection is then defined as the Markov Decision Process (MDP), which calculates reward (penalty) of transition from the current state *s* to one of possible future states *s'* [Bolsch98]:

$$V_\pi^k = Est\left[\sum_k R^t \middle| \pi, s\right] = R(s)\sum_{'k} T(s, \pi(s,k), s) V_\pi^k(s) \tag{234}$$

$$\pi : s \rightarrow a \tag{235}$$

The current state *s* represents currently selected path (using the serving cell) and the future state *s'* represents another possible path including all combinations of radio and backhaul connections. Hence, the estimate (*Est*) represents possible outcome of reward by performing handover to a different cell. The *Est* is computed over *k* steps, representing duration of the data transmission. The parameter π

stands for the policy, which defines what action (*a*) should be taken in step *s* to maximize total reward as (234) denotes. Mapping the action (*a*) to the state *s* is a stationary policy if mapping is the same regardless of the time. Total reward for transition from the state *s* to the *s'* consists of two parts. The first one, *R*(*s*), denotes immediate reward for transition from the state *s*. The second part, summation, represents expected future payoff as a sum over *k* steps. In this document, π is obtained at the end of the algorithm providing desired policy maximizing the reward. As the delay and the energy are used as metrics, optimal policies can be calculated in order to minimize delay, energy or a trade-off between both metrics. The reward depends on the delay due to handover if the handover is performed (T_H), delay by the transmission over radio (T_R), delay on backhaul (T_B) and delay using X2 interface for the Over The Air (OTA) communication (T_{OTA}). Thus, the reward for transition from the state *s* to the *s'* is written as:

$$V_\pi^k(s,s') = \gamma\left[k\left[E\left[T_R(s')\right] - E\left[T_R(s)\right]\right] + E\left[T_H(s,s')\right]\right] + (1-\gamma)\left[T_H(s,s') + k[T_R(s') - T_R(s)] + k[T_B(s') - T_B(s)] + k[T_{OTA}(s') - T_{OTA}(s)]\right] \quad (236)$$

where γ is the weighting factor showing preference for low delay ($\gamma \to 0$) or for high energy efficiency ($\gamma \to 1$), $E[T_R]$ denotes the energy consumed by the UE's radio communication through the serving cell (in the state *s*) or another neighboring cell (in the state *s'*), $E[T_H(s; s')]$ stands for energy consumed by handover from the serving to the neighboring cell (from state *s* to state *s'*). The transmission delays T_R ,T_B and T_{OTA} are computed knowing amount of data to be transferred over radio (n_{bits}^R) and backhaul (n_{bits}^B) and knowing capacity of radio link (c_i^R), capacity of OTA link (c_i^{OTA}), capacity of backhaul of the serving (c_i^B), and the computing (c_x^B) cells:

$$T_R = \frac{n_{bits}^R}{c_i^R}; \quad (237)$$

$$T_B = \frac{n_{bits}^B}{c_i^B} + \frac{n_{bits}^B}{c_x^B} \quad (238)$$

$$T_{OTA} = \frac{n_{bits}^B}{c_i^{OTA}} \quad (239)$$

Each SCeNBce can be switched off at any time since the SCeNBces can be deployed also by users. Thus, a secondary path should be defined for cases if the primary path is no longer available due to its failure. To keep routing overhead low in case of the link failure, data to be send over this link will be rerouted through the serving SCeNBce if possible. In case of the serving SCeNBce failure, the UE will reinitiate path selection as there is a major change in state of links. Note that this problem requires also selection of a new serving cell. However, this is common problem, for which even existing mobile network must be able to find a solution. Therefore, we do not address this problem.

6.1.2 **Derivation of path selection parameters**

To enable path selection, several metrics and information on several individual paths must be known. This subsection describes how the individual parameters can be derived in the existing networks. The set of computing cells is labeled as *Y* and the set of cells with radio access is denoted as *I*. Each computing task is split into *n* subtasks. Each subtask is of the size L_{UE} computed as:

$$L_{UE} = \sum_{n \in Y} \lambda_n L_{UE} \quad (240)$$

$$\sum_{n \in Y} \lambda_n = 1 \quad (241)$$

where λ specifies distribution of load among computing cells (i.e., how big part of the whole task will be computed at a given cell).

While considering the SCeNBces and especially the HeNBs, the capacity of the radio link of the UE can be higher than the one of backhaul connection (especially for DSL). In the proposed path selection algorithm, the computation of metrics and routes is done at the SCM to lower energy consumption of the UE. In order to lower both algorithm overhead and its complexity, Neighbor Cell List (NCL) [3GPP TS 36.331] is exploited. The NCL contains all potential neighbors of the serving cell. Thus, this list corresponds to the list of cell, which can be considered for data transmission. To describe the algorithm following notation of the sets of SCeNBces is introduced:

- T is the set of the SCeNBces included in the NCL with SINR to UE above a selected threshold T_1:

$$T \subseteq I : SINR(T) \geq SINR_{T_1} \tag{242}$$

- O is the set of the SCeNBces from the set T with SINR, which is lower than the SINR between the UE and its serving SCeNBce only by a selected threshold T_2:

$$O \subseteq T : SINR(O) \geq \left(SINR_{SCeNB_{serving}} - SINR_{T2}\right) \tag{243}$$

both sets are defined to lower the number of the SCeNBces considered for calculation of the optimal path for data delivery. In T we remove all SCeNBces from which the UE observes SINR below the threshold value ($T1$). This threshold is defined in order to ensure enough radio resources for data transmission. Since the UE uses the serving SCeNBce there is no reason to make a handover to different SCeNBce with SINR deep below SINR of its serving SCeNBce. This way the O is created to cut off unusable SCeNBces with very low channel quality. To that end, the second threshold T_2 is defined.

These sets will be updated if there is a new SCeNBce detected. If the UE demands path to a single or multiple SCeNBces for its data, the following four steps are followed:

1. The UE sends list T to its serving SCeNBce and to all SCeNBs in O
2. The serving SCeNBce sends, along with cells in O, a new message to the cells in T to obtain delay and available bandwidth.
3. Using information obtained in the previous step, the serving SCeNBce and the SCeNBces in O derive their own routing tables with information on delay to the neighbors (T or Y)
4. The UE obtains the cost of individual routes to all computing SCeNBces and selects the path with a minimal delay.

For the path selection algorithm, two new management messages are proposed to obtain information on delay and available bandwidth between each neighbor cell from the NCL with SINR above selected threshold. Format of both messages is shown in Figure 167.

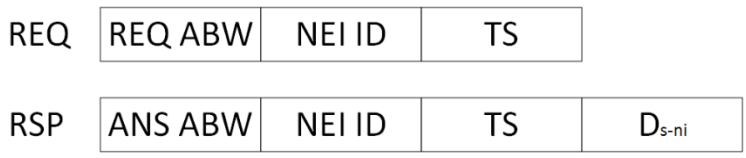


Figure 167. Messages for obtaining information on network delay.

The messages are exchanged in the following way:

i. Source SCeNBce (i.e., the SCeNBce requesting the information) sends ***REQ*** message to destination SCeNBce (one of the neighbors)**.** The ***REQ*** contains Request of Available Bandwidth (***REQ ABW***), Neighbor ID (***NEI ID***) and Time Stamp (***TS***).

ii. Neighbor ***ni*** responds with ***RSP*** containing Response with Available Bandwidth, its ***NEI ID***, Time Stamp (***TS***) and delay observed for direction from the source cell to the neighbor $\boldsymbol{D_{S\text{-}ni}}$.

$$D_{S\text{-}ni} = TS - received\,time \tag{244}$$

iii. The source SCeNBce calculates backward delay from the neighbor to the source cell $\boldsymbol{D_{ni\text{-}s}}$ in the same way as the neighbor.

In 3GPP [3GPP TS 36.314], methods for obtaining packet delay measurement and Scheduled IP Throughput for the radio part are described. Packet delay is intended for use of Voice over IP (VoIP) and thus downlink (DL) and uplink (UL) are measured independently. Measurement uses the same principle of sending time stamp and calculating delay from subtracting two values. Difference is in calculation of delay for DL and UL separately. IP throughput is measured for LTE-Uu interface. Therefore, approach proposed in [3GPP TS 36.314] can be used for delay measurement at the radio link (LTE-A), while the proposed messages are implemented in order to calculate backhaul delay and to obtain available bandwidth.

6.1.3 **Algorithm complexity**

Complexity of the path selection algorithm is proportional to the number of computing SCeNBces (n) and the number of the SCeNBces in proximity of the UE (m). The SCeNBs in proximity of the UE are selected according to SINR as described in the previous section. The number of possible paths can be computed as partial permutation thus complexity of the proposed path selection algorithm is $O(m^n)$. When the OTA interface is added, SCeNBces may utilize OTA instead of backhaul connection. Possibility of OTA doubles the complexity, as we can utilize OTA instead of backhaul connection if OTA provides lower delay, making it $2m^n$, however complexity is still $O(m^n)$. Number of computing SCeNBces n is to be in order of units and number of SCeNBces in proximity m is to be in order of units also but is being even further reduced by selecting only SCeNBces with SINR above specified threshold, to lower algorithm complexity.

6.1.4 **System model and simulation methodology**

In this section, models and scenarios for performance evaluation are defined. The evaluation is carried out by means of simulations in the MATLAB.

6.1.4.1 ***System model***

We assume the system composed of S SCeNBces and U UEs. For each UE, a serving cell is selected as the SCeNBce with the highest RSSI in the first step. As the UE moves, the serving cell is updated if the RSSI from the target SCeNBce ($RSSI_T$) is higher than the RSSI of the serving cell ($RSSI_S$) plus handover hysteresis (Δ_{HM}), i.e., if $RSSI_T > RSSI_S + \Delta_{HM}$.

For each computation offloading request, the maximal delay of data delivery from the UE to the computing cells, T_{req}, is specified. This delay can be derived as a difference between maximum delay required by the UE for delivery of the computation results back to the UE (T_{max}) and the time required for computation of the offloaded task (T_{comp}); $T_{req} = T_{max} - T_{comp}$. Parameters T_{max} and T_{comp} are related to application and available computing capacity of cloud-enhanced SCeNBces, respectively. For our purposes, specific way of T_{max} and T_{comp} derivation is not relevant; we just need to know the time constraint remaining for data transmission.

The set of SCeNBces selected for computation is denoted as a Y. Each cell is expected to compute a part $\lambda_n \epsilon (0,1]$ of the whole offloaded task of the overall size of L_{UE}. The individual part L^n computed by the $SCeNBce_n$ is then expressed as $L_n = \lambda_n \cdot L_{UE}$ with $\sum \lambda_n = 1$. In this document, we assume equal distribution of the task among all computing cells i.e. $\lambda_1 = \lambda_2 = ... = \lambda_n$.

As shown in Figure 166, data from the UE to the *SCeNBce*$_i$ is transferred over radio link with capacity c_i^R. Further, the *SCeNBce*$_i$ is connected to the operator's core with a backhaul of capacity c_i^B. Data is then processed by the *SCeNBce*$_i$ or forwarded to another computing *SCeNBce*$_x$ through backhaul of the *SCeNBce*$_i$ (with capacity c_i^B in UL) and backhaul of the computing cell *SCeNBce*$_x$ (with capacity c_x^B in DL). Note that index *x* stands for any SCeNBce out of *S* except *SCeNBce*$_i$. After cells perform data computation, the results are delivered back to the UE. New path for backward data delivery (from SCeNBce$_x$ to the UE) can be derived if radio and backhaul links are not symmetric in UL and DL, if the UE moves during computation or if the channel/link load or quality change. Otherwise, the same path can be reused.

6.1.4.2 *Transmission of offloaded data to the computing cells (UL)*

In UL connection, Single Carrier FDMA (SC-FDMA) is used in LTE. As a way to transmit to different SCeNBce, handover needs to occur. While in WCDMA, networks possibility of soft handover existed, LTE(-A) supports only hard handover. If UE transmits data to multiple computing SCeNBces, handover (on a radio interface) from first computing SCeNBce Y_1, to other computing SCeNBce Y_2 should occur only if the delay to deliver required data utilizing radio connection to Y_2 ($D_{Y2radio}$) and delay caused by a handover ($D_{handover\ to\ Y2}$) is lower than the delay utilizing radio to Y_1 ($D_{Y1radio}$) and backbone connection between Y_1 and Y_2 ($D_{Y2backbone}$) as is shown in equation (245).

$$D_{Y_1 radio} + D_{Y_{2backbone}} > D_{Y_2 radio} + D_{handover\ to\ Y_2} \tag{245}$$

To illustrate how much data can be transmitted on a radio interface of LTE during the handover procedure for different number of Resource Blocks (RB) is shown below in a Figure 168.

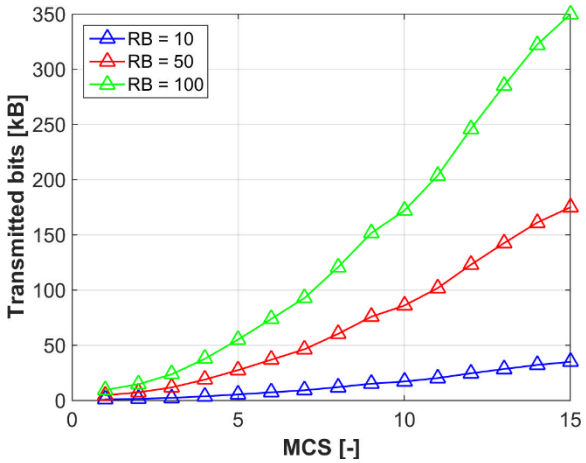


Figure 168. Transmitted kB during handover (30 ms).

6.1.4.3 **OTA interface**

In order to provide UE alternative for backhaul connection, OTA in [TROPIC_D321] is described. It enables SCeNBces to communicate with each other utilizing radio resources over existing X2 interface using radio connection as shown in Figure 169, where OTA interface provides alternative to overcome backhaul connection of low quality. The X2 interface can be also created using backhaul connection. This however does not affect PSwH algorithm as if X2 interface would be utilizing backhaul connection it would look to PSwH as backhaul connection being more utilized.

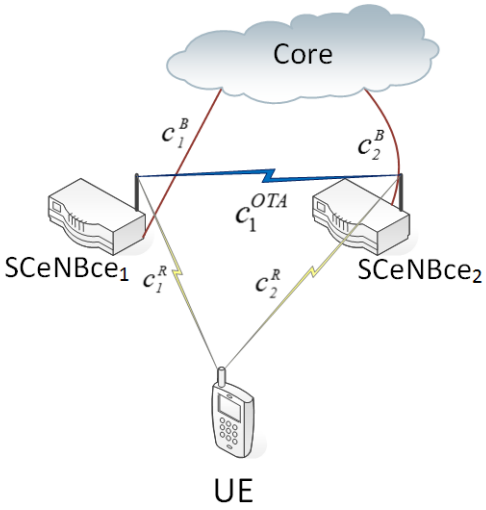


Figure 169. OTA interface.

In [TROPIC_D322] is provided description of OTA interface using UL or DL resources. As there is an asymmetry between available UL and DL resources, labeled R_{UL} and R_{DL}, respectively, we can utilize either of them, based on the resources utilization thus using direction with more available resources, while respecting available resources at the source SCeNB *S* and the target SCeNB *T* as expressed in following formula:

$$R_{OTA} = \max\{\min[R^S{}_{UL}(t), R^T{}_{UL}(t)], \min[R^S{}_{DL}(t), R^T{}_{DL}(t)]\} \quad (246)$$

The capacity of OTA link c_i^{OTA} is then calculated as a function of R_{OTA} and $SINR_i$.

For the simulation purposes, we utilize OTA connection only, if at the source and destination SCeNBs, number of connected UE is less than ε and if gain by use of OTA is higher than ρ that could be expressed as:

$$\frac{D(backbone)}{D(OTA)} > \rho \quad (247)$$

As resources for OTA and UE are shared, OTA should be utilized only if there are enough available resources for UE or if a gain due to use of OTA is significant..

6.1.4.4 *Simulation scenario and models*

Major parameters of the simulation, presented in Table 39, are in line with the recommendations for networks with small cells as defined by 3GPP in [3GPP TR 36.814]. We also follow parameters of the physical layer and frame structure for the LTE-A mobile networks as defined in the same document.

Requests for data offloading are generated by UEs continuously, just after end of the previous one, to model behavior of heavily loaded system. This case is the most challenging due to limited capacity of backhaul and radio. Each request corresponds to generated traffic of a size of 300kB and 30MB. The offloaded application is computed at 1, 2, 3 or 4 SCeNBces, with equal probability of each option. One of the computing SCeNBces is always the serving one (as suggested in [Valerio14]). All UEs are moving within an area composed of two-stripes of buildings [3GPP TR 36.814] as shown in Figure 170. Size of each block of buildings is 20x100m and blocks are separated by streets with width of 10m. The overall area is composed of 4x4 blocks (i.e., size of the whole simulated area is 560x130m). Fifty outdoor UEs are randomly deployed at the beginning of the simulation and then they move along the streets according to Manhattan Mobility model [Meghanathan10], with movement speed of 1 m/s. One eNB is placed outside the area with the two stripes buildings at coordinates of [200m, 200m].

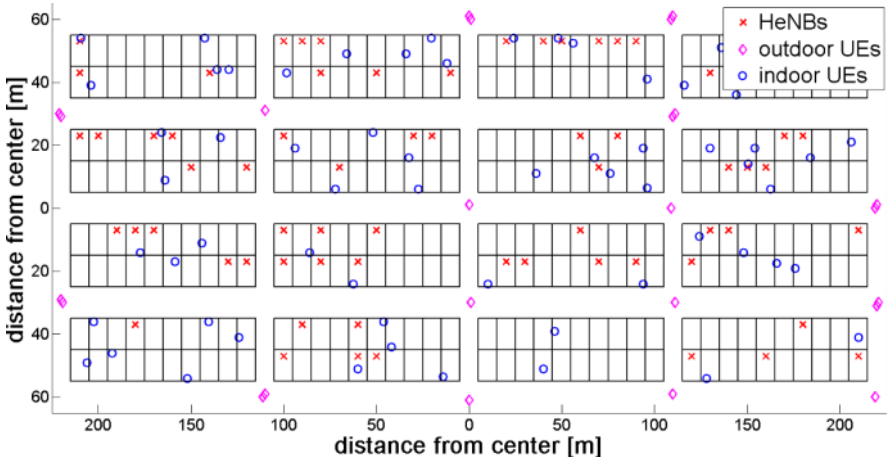


Figure 170. Simulation scenario with example of deployment of buildings, users, HeNBs and eNB for simulations.

Inside buildings, the SCeNBces are randomly dropped to the apartments with equal probability in a way that 20% of apartments are equipped with a SCeNBce. Therefore, 64 UEs and 64 SCeNBces are deployed indoor. Movement of the indoor UEs is modeled according to [TROPIC_D21], the UEs move within an apartment at discrete positions with a specific time distributions as defined in [TROPIC_D21].

Parameter	**Value**
Simulation area	560m x 130m
Carrier frequency	2 000 MHz
Tx power of eNB/SCeNBce [dBm]	43 / 23
Attenuation of external/internal/separating walls	20/3/7 dB
SCeNBce deployment ratio	0.2
Shadowing factor	6 dB
Handover interruption duration	30 ms
Number of Indoor/Outdoor UEs	64/50
Speed of outdoor users	1 m/s
Traffic generated by one request	300/3000 kB
Simulation time	20 000 s
OTA UE limit ε	4
OTA delay ratio limit ρ	1.1
Radio - available RBs in UL/DL	40/80

Table 39. Simulation parameters.

In order to show performance of the proposed algorithm, two types of backhauls are used: ADSL and optic fiber. The first type of backhaul corresponds to the Residential scenario while the second one is used for corporate scenario [TROPIC_D21]. In the residential scenario, the SCeNBces are deployed in private flats. The corporate scenario assumes fiber optic backhaul and Local Area Network (LAN) consisting of Ethernet connection with the throughput of 100 Mbit/s among SCeNBces within the same building. If two or more SCeNBces from the same block communicate with each other, we assume only LAN connection and no data is distributed over fiber optic. The basic characteristics of both backhaul models is summarized in Table 40.

Corporate scenario (optical fiber)	
μ (backhaul UL/DL)	100/100 Mbit/s
σ(backhaul UL/DL)	11.5/11.5
Shared block backhaul UL/DL	100 Mbit/s
MC UL/DL	1000/1000 Mbit/s

Residential scenario (ADSL)

μ (backhaul UL/DL)	1/5.5 Mbit/s
σ(backhaul UL/DL)	0.52/2.6
MC UL/DL	1000/1000 Mbit/s

Table 40. Backhaul models.

6.1.4.5 *Energy consumption model*

Energy consumption at the UE is due to radio transmission only. From the standards [3GPP TS 36.213] and models from [Ahmadi13], [Lauridsen11] and [Lauridsen14] we calculate power requirements on the UE, based on received signal, required throughput, etc. The energy *E* is computed as the power consumed by the UE for data transmission over a transmission time. Since both UL and DL power models are different, we describe them separately.

6.1.4.5.1 Uplink

To avoid dramatic increase in energy consumption at the UE, we model energy *E* as the power consumed by the UE for data transmission over a transmission time. The energy consumption depends on Modulation and Coding Scheme (MCS) and available bandwidth represented by RBs in LTE-A system. The MCS is a function of Signal to Interference plus Noise Ratio (SINR) observed at the receiver. The SINR at the receiver is proportional to transmission power P_{Tx} at the transmitter, path loss and interference from other cells. In LTE-A, the P_{Tx} required for selected MCS for a given number of allocated RBs is defined, according to 3GPP [3GPP TS 36.213] and [Ahmadi14] as follows:

$$P_{Tx} = \min(P_{MAX}, P_0 + \alpha \cdot PL + 10\log_{10}(M) + \Delta_{TF} + f) \quad (248)$$

where P_{MAX} is the maximum available transmission power (23 dBm or the UE class 3 [3GPP TS 36.101]); $\alpha \epsilon \{0, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ corresponds to the path loss compensation factor, *PL* is path loss estimate in DL, *M* stands for the number of assigned RBs, Δ_{TF} represents a closed loop UE specific parameter, which is based on the applied MCS, and *f* is a correction value, also referred to as a TPC command (for more details on Δ_{TF} and *f*, see [Ahmadi14]); and parameter P_0 represents the power offset computed as: $P_0 = \alpha \cdot (SINR_o + P_N) + (1-\alpha) \cdot (P_{MAX} - 10\log_{10}(M_0))$; where P_N is the noise power per RB, and M_0 defines the number of RBs for which the SINR target is set with full power.

Parameters Δ_{TF} and *f* are used for dynamic adjustment of the transmission power to keep required SINR at the receiver. As we use open loop power control, we can omit these parameters as indicated in [Tejaswi13]. The parameter α is set to 1 so the UE fully compensates path loss. Under these assumptions, we can simplify power offset to $P_0 = \alpha \cdot (SINR_o + P_N)$ and then, (248) can be rewritten as:

$$P_{Tx} = \min(P_{MAX}, \alpha \cdot (SINR + P_N + PL) + 10\log_{10}(M)) \quad (249)$$

The energy consumed by transmission of data over a radio channel is derived as:

$$E = P_{Tx} \cdot T_R \quad (250)$$

6.1.4.5.2 Downlink

For the energy consumed by the UE in DL direction, different model is used as the UE just receives data. The power required to receive data depends on P_{rx} and data rate as well. Main power consuming part of the UE is the power amplifier and the turbo decoder. In this document, power consumption model defined in [Lauridsen14] is used to match the power consumption to data transmission:

$$P_{Rx} = P_{RxBB} + P_{RxRF}\,[mW] \quad (251)$$

$$P_{RxRF} = \begin{cases} -0.04 \cdot S_{Rx} + 24.8\,[mW], where\ S_{Rx} \le -52.5dBm \\ -0.11 \cdot S_{Rx} + 7.86\,[mW], where\ S_{Rx} > -52.5dBm \end{cases} \quad (252)$$

$$P_{RxBB} = 0.97 \cdot R_{Rx} + 8.16\,[mW] \quad (253)$$

where P_{RxRF} denotes radio component depending on the received power S_{Rx}, P_{RxBB} denotes radio component depending on bit rate of received transmission. Energy consumption is calculated in the same way as for the UL.

6.1.5 Simulation results

The evlaution is done by MATLAB simulator with models described in previous section. Preliminary results has been presented in [Becvar14]. Traffic generated by a UE request for the SCC is affecting network load, and, thus it lowers available resources for further requests. In this regard, for each new generated request is dedicated exactly half of the available resources (both radio and backhaul), while the rest of resources is reserved for new potential requests to come in near future. Time between two arrived UE requests is selected to T_{UEreq}= 64 s and T_{UEreq}= 512 s for request size of 300 kB and 30 MB, respectively. Time between requests is selected in order to generate enough traffic so one request transfer affects selection of path for another.

Comparison of delay achieved by both the SO and proposed PSwH schemes is shown in Figure 171, where request load is 300 kB and 30 MB. If the PSwH is utilized for DSL backhaul and 300 kB request load is assumed, delay *D* is reduced by up to 26.9%. The reduction in *D* by up to 52.8% is observed if using the PSwH in combination with OTA. For the optical fiber backhaul, delay is reduced by up to 7%. However, if OTA is utilized, delay is increased by 34.4%, as shown in Figure 171a for optical fiber backhaul. Still, since the optic fiber is of high throughput, the increase in delay corresponds only to 0.26 s. For request load of 30 MB, the PSwH shortens the delay by 21.5% compared to the SO in case of DSL backhaul and by up to 48.8% if DSL backhaul is utilized together with OTA. For the optical fiber backhaul, delay is shortened by up to 53.7% and by up to 62.6% if PSwH and PSwH+OTA are utilized, respectively, as shown in in Figure 171b. As the request size is increased, delay using PSwH decreases as the reward of handover to other SCeNBces increases. Use of OTA when the request size is low (300 kB) for backhaul of high quality provides an increase in mean delay. The reason is that OTA consumes resources for other UEs. For other combinations of backhaul and request size, OTA reduces delay as it either overcomes backhaul of low quality or the backhaul is highly utilized.

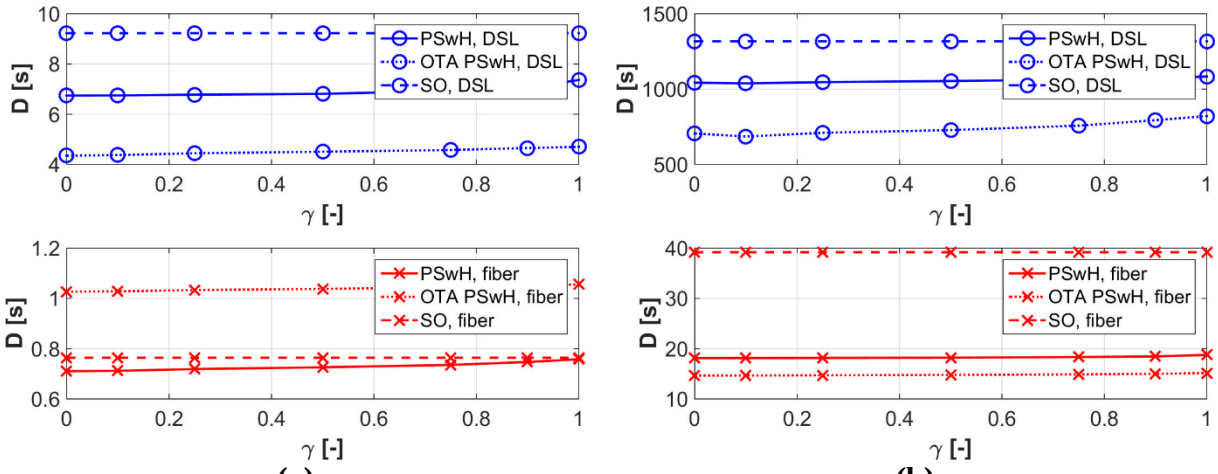
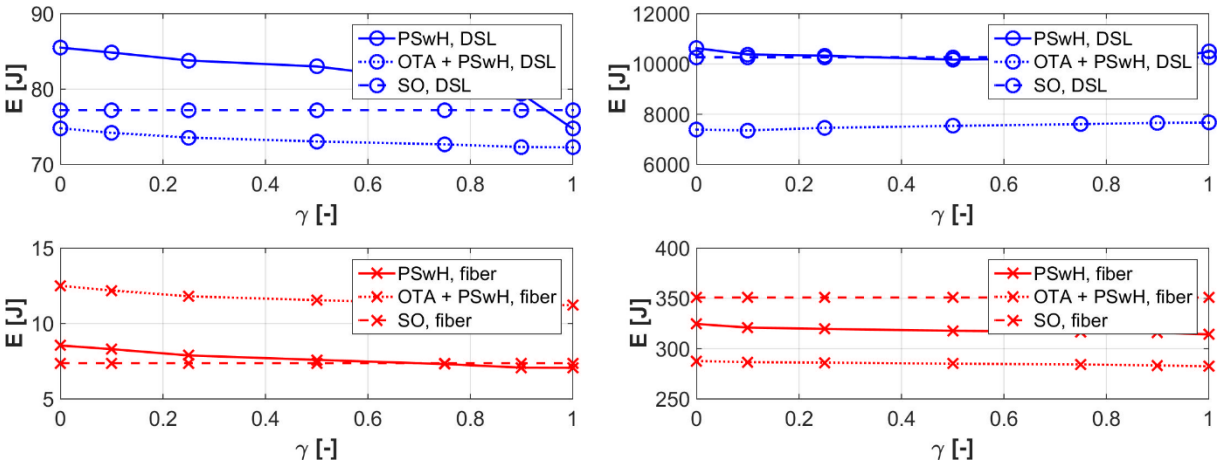


Figure 171. Average delay D required for transmission of offloaded computing task to computing cells for request size of 300kB (a) and 30 MB (b).

From the energy point of view, the PSwH decreases energy consumption by more than 3% if DSL backhaul and request load of 300 kB is taken into account. Enabling also OTA for DSL adn 300 kB requests, energy consumption is further lowered by up to 6.5%. For optical fiber backhaul, the PSwH decreases energy consumption by more than 4%, whereas PSwH in combination with OTA increases energy consumption by up to 52% as shown in Figure 172a. This increase is again caused by

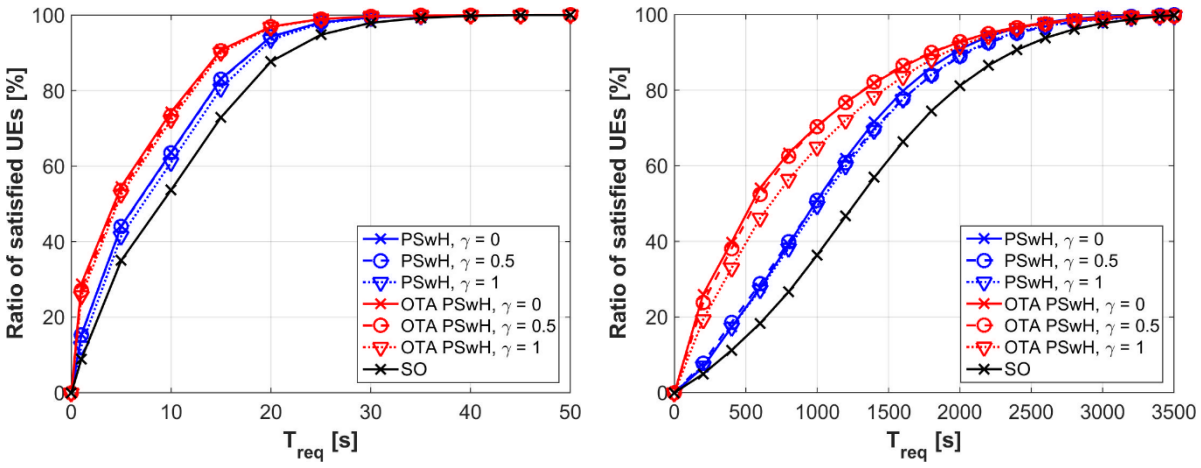
consumption of radio resources commonly allocated for users by OTA. When request size is increased to 30 MB carried over DSL backhaul, energy consumption is decreased by up to 4.1% and by up to 28.8% if PSwH and PSwH+OTA are used, respectively. For the optical fiber, energy consumption is decreased by up to 10.5% and 19.5% when PSwH and PSwH+OTA are utilized, respectively, as shown in Figure 172b. As delay is increased for the optical fiber and request size of 300 kB when OTA is utilized, energy increases as radio transmission between UE and SCeNBces requires more time.



(a) **(b)**

Figure 172. Average energy E required for transmission of offloaded computing task to computing cells for request size of 300kB (a) and 30 MB (b).

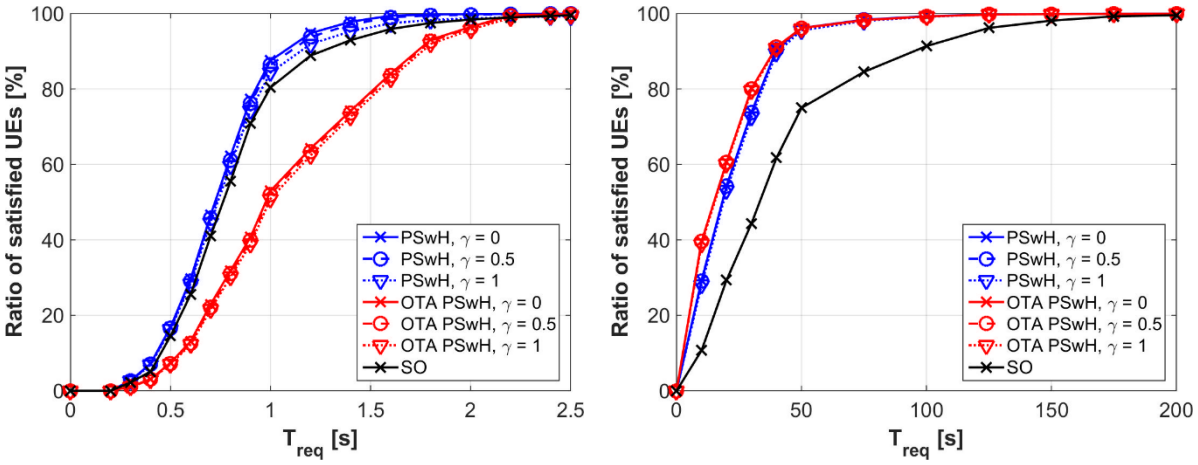
Satisfaction of UEs for DSL backhaul and request size of 300 kB (see Figure 173a), is increased by up to 10% using the PSwH and up to 20% if OTA is utilized. For the request size of 30 MB, satisfaction of UEs is increased by up to 15% for PSwH and up to 36% if also OTA is utilized as shown in Figure 173b. As the request load increases, utilization of handover to other computing SCeNBces enables to decrease delay even more as the DSL backhaul is a limiting factor in this scenario.



(a) **(b)**

Figure 173. Satisfaction of users with experienced delay for DSL backhaul, for request size of 300kB (a) and 30 MB (b).

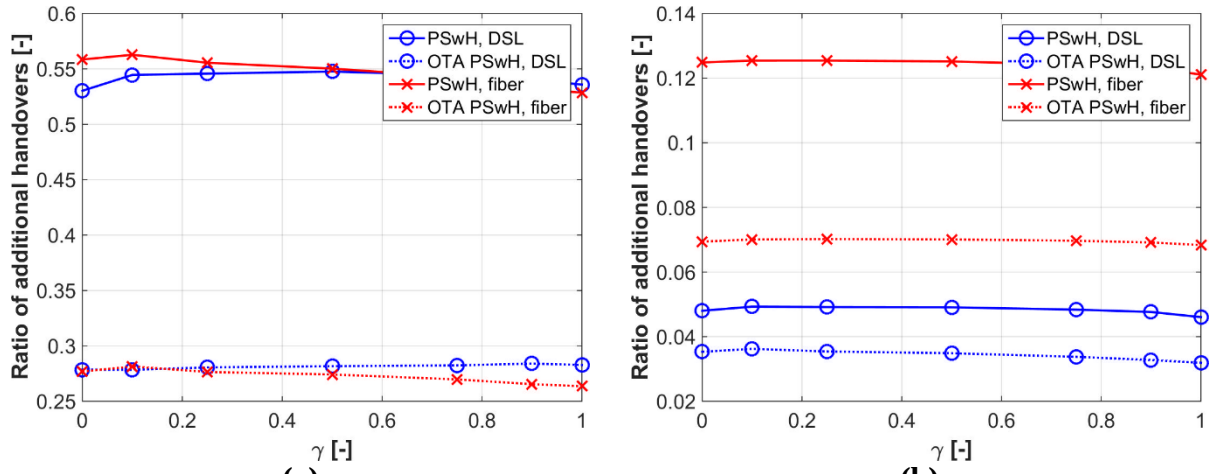
For the optical fiber backhaul adn 300 kB size of requests, the PSwH provides up to 7% increase in UE satisfaction, while up to 30% decrease in satisfaction is observed if also OTA is considered as shown in Figure 174a. The degradation of satisfaction by OTA is caused by sharing radio resources of users with OTA comunication. However, for the request size is of 30 MB, the PSwH increases UE satisfaction up to 28% and OTA further improves satisfaction up to 36% as shown in Figure 174b.



(a) **(b)**

Figure 174. Satisfaction of users with experienced delay for optical fiber backhaul, for request size of 300kB (a) and 30 MB (b).

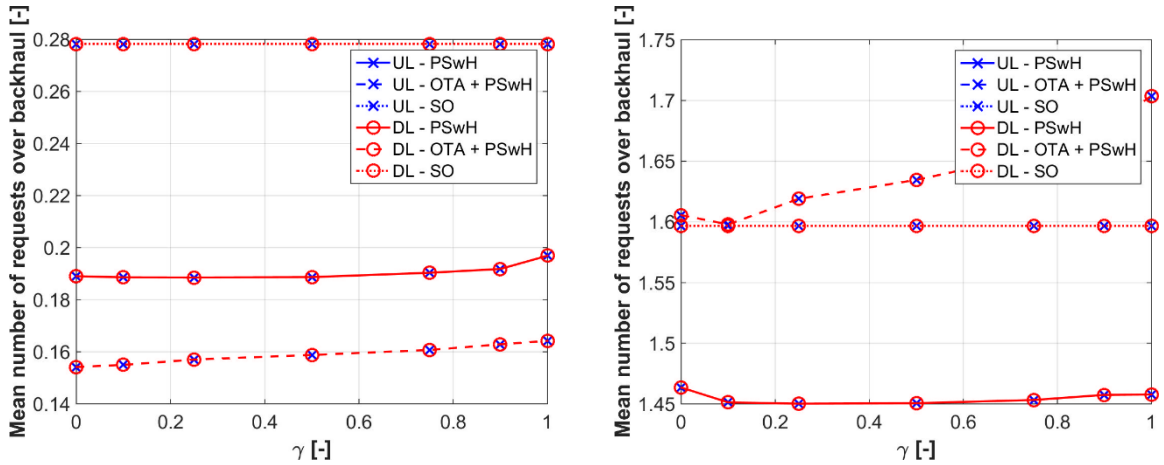
Like in the previous scenario, we also investigate impact of the proposed algorithm on the number of handovers. Number of additional handovers for request load of 300 kB is shown in Figure 175a, whereas for load of 30 MB is depicted in Figure 175b. If the PSwH is used for request size of 300 kB, the number of handovers increases by 55% for both backhauls. If request load size is 30 MB and DSL backhaul is exploted, the number of handovers increases by 5% while for the optical fiber backhaul increases by 12.5%. Use of the OTA reduces number of additional handovers up to 50% for both request sizes as it provides an option to overcome backhaul with a low throughput avoiding unnecessary handovers. As the request size increases, number of handovers for DSL decreases. Contrary, for the OTA and optical fiber backhaul, slightly more handovers is performed. A decrease in number of handovers for the DSL is caused by utilization of other SCeNBces than serving one by requests from other UEs. This make handover disadvantageous as the DSL backhaul provide less throughput than optical fiber or OTA. Higher throughput of optical fiber backhaul and OTA, on the other hand, makes handover advantageous for higher request size. Generated signaling overhead by handover is in order of kb per handover event so this increase in the number of performed handovers introduce only negligible overhead. Delay of handover (handover interuption) is included in PSwH algorithm, so there is no additional negative impact on users quality os service.



(a) **(b)**

Figure 175. Ratio of additional handovers due to proposed algorithm with respect to usage of serving cell only (SO), for request size of 300kB (a) and 30 MB (b).

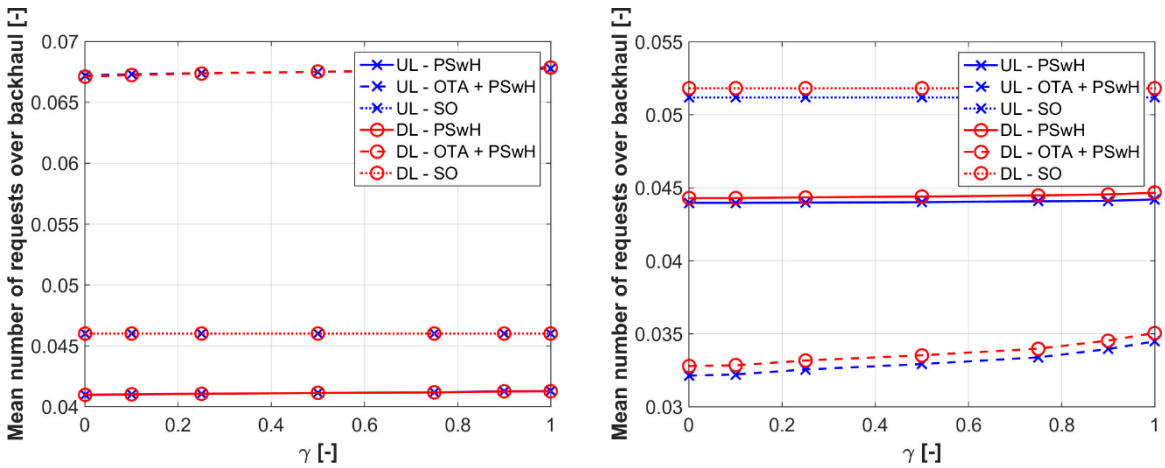
The proposed algorithm takes advantage of handover to speed up data delivery and also to offload backhaul of small cells. The load of backhaul is represented by mean number of requests transmitted per backhaul link and time (see Figure 176 for DSL and Figure 177 for optical fiber). Comparing to the SO, the PSwH reduces backhaul load by up to 32% and if OTA is utilized by up to 44.6% for both directions (UL and DL) as shown in Figure 176a. For the load of 30 MB, the PSwH reduces backhaul load by up to 9.2%. If OTA is used and γ is equal to 0.2, the backhaul load is the same as in case of the SO. Nonetheless, backhaul load increases up to 6.7% as the gamma rises. Increase is caused by radio resource utilization by OTA, which then forces UEs to utilize backhaul. However, as the radio is being over-utilized, delay in transmission increases, which leads to a higher load of backhaul. The decrease in load of backhaul, apart from the mentioned case, by the PSwH is due to exploitation of the knowledge of link parameters. With this knowledge, algorithm chooses the path, which may require handover, but which also provides a higher throughput.



(a) **(b)**

Figure 176. Load of DSL backhaul, for request size of 300kB (a) and 30 MB (b).

In case of the optical fiber backhaul, the PSwH lowers the backhaul load by up to 11% while, if OTA is used, the load is increased by up to 45.5% for the request size of 300 kB as shown in Figure 177a. For the request size of 30 MB, PSwH reduces backhaul load by up to 15.5% for UL as well as DL. If OTA is used, the load is decreased by up to 37% again for both directions, as shown in Figure 177b. An increase in backhaul load for request size of 300 kB while using OTA is caused by consumption of radio resources for UEs by OTA. This causes other UEs to have less resources and thus it increases time of transmission. Backhaul utilization rises with request size. In this case, OTA can help to alleviate backhaul load.



(a) **(b)**

Figure 177. Load of optical fiber backhaul, for request size of 300kB (a) and 30 MB (b).

6.1.6 Summary of performance

In this section optimal values of γ for the proposed algorithm are discussed. Optimum value of γ is selected according to combined impact on energy and delay.There is a trade-off between dealy and energy, so we present two values of γ representing different level of gain prefered by users in energy and delay. Optimal value of γ are shown in Table 41. Note that green cells represents inprovement in respective scenario and metric while red color means worse performance comparing to the SO.
For the optimums elected γ, the PSwH provides improvement in both delay and energy except optical fiber backhaul in combination with OTA and small size ofoffloaded requests. The OTA decreases delay in comparison to the PSwH, for DSL backhaul by up to 30% and energy up to 27%. For the optical fiber, delay is reduced up to 9% and energy by 10%. For the optical fiber backhaul and load of 300kB, both delay and energy are increased as explained below respective figures.

backhaul	load	optimal γ value	reduction of D [%]	reduction of E [%]
DSL	300 kB	1	20.2	3.2
	30 MB	0.75	19.2	1
DSL + OTA	300 kB	0	52.8	3.2
	30 MB	0.9	46.5	28
optical fiber	300 kB	0.75	3.8	0.8
	30 MB	0	54.3	7.5
optical fiber + OTA	300 kB	1	-38.2	-52.2
	30 MB	0	63.2	18

Table 41. Optimal γ and impact on delay and energy consumption of the proposed PSwH comapring to the SO.

6.1.7 Conclusion

The proposed algorithm PSwH enables the UE to select preferences on delivery of computed results by weighting delay due to transmission/reception of data and energy consumned by the UE for data transmission/reception. The PSwH exploit handover to gain from communication over radio between UE and multiple SCeNBces to speed up the data transmission and/or energy saving. From the simulation results can be observed that the PSwH can significantly lower the transmission delay. In addition, depending on weighting of delay and energy, also a lower energy consumption can be achieved as well. Improvement is seen also in the satisfaction of the users with experienced delay as the UEs utilize the paths with lower delay. From the network point of view, the PSwH requires additional handovers. Nevertheless, number of additional handovers is, in average, lower than 0.9 handover per offloaded request. This introduced only negligible overhead in order of few kB per offlaoded tasks. The delay caused by handover itself is already considered in the proposed algorithms so it further influences neither delay nor users' satisfaction. In addition to this, utilization of handovers for data delivery can reduce load of backhaul network. The results can be found also in [Becvar14] and [Plachy14].

Extension of PSwH by enabling OTA can further improves performance of th eproposed algorithm. However, OTA increases delay and energy for the optical fiber and small size of offloaded requests (300 kB) as the radio link is limiting factor in this case and OTA consumes a part of radio resources normaly dedicated for users. should be used only when improvement by usage of OTA is significant.

6.2 *Migration of VMs for mobility support*

Handover can occur due to multiple reasons, for example changing radio conditions caused by UE movement or load balancing or just because a better signal is received from a newly-installed eNB. Another reason for handover may be that the available VM and memory resources are in a different eNBs in the radio neighborhood.

The application shall be designed such that it can be exited or stopped upon request and the execution context can be saved. It is preferable that, if possible, during the UE handover the application will enter a dormant state until the handover has been completed.

The execution context means the values of all relevant execution-related parameters, as well as the memory content. This includes program address, value of loop counters, stack addresses, memory content within the relevant memory range, etc.
At the virtual machine level, the execution context may have different names, however the meaning is similar.

The application handover request may be the result of a SCM-BS decision or of a SCM-UE request. The SCM-BS request should be transmitted to the radio controller of the serving eNB, part of the eNB Computing Unit, to perform a handover from the source eNB (serving eNB) to the target eNB indicated by the message from SCM-BS.
In another case, the SCM-UE may ask the SCM-BS to check the possibility of a handover. It may optionally indicate a preferred target eNB.
This request will go over the Z-C interface to the serving SCM-BS, which will control the context (including the relevant memory content) transfer to the new VM.

An alternative approach is that the serving SCM will initiate the request over Z-C interface and the SCM-UE will cause the UE to execute the handover.

If the target SCM-BSs sends back a response indicating plain refusal or some degree of refusal, the serving SCM-BS may try to contact another collaborating SCM-BS.

The collaborating SCM-BSs may share the status of availability of their VMs each time a change is made.

In situations in which an UE has multiple applications running on multiple VMs and the VMs are not located in the same eNB, it may be necessary to send the application data from the serving eNB to the cooperating eNB and retrieve the results provided by the application from this eNB. In such a case the data in both directions will be sent in a container over the Z-interface. This mode of operation is similar to LTE X2-interface operation in the initial handover period, but in the case of Z interface will be active for the duration of the application offload.

It is clear that there is an interaction between the radio handover and application handover. For avoiding execution errors or losing data, it is recommended to follow the process described below and in Figure 178.

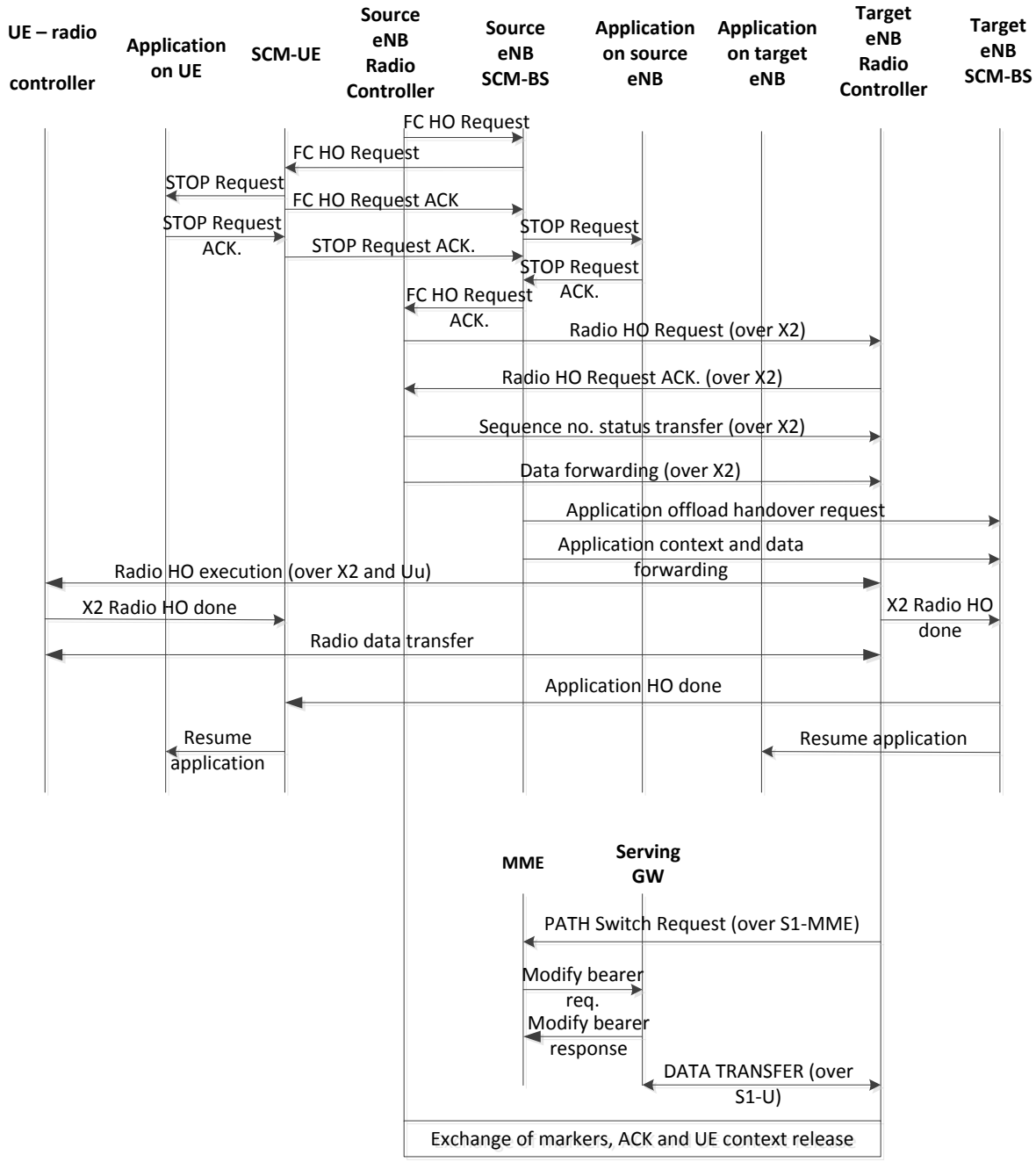


Figure 178. Flowchart for virtual machine handover.

The high-level approach is:

1. SCM-BS makes sure that the handover of the VMs is possible;
2. The application offload is stopped;
3. The UE radio handover is executed and the external data arrives at the target eNB; in parallel, the handover of the context of the VM to the target eNB is executed.
4. The application offload is resumed using a VM on the target eNB.

The combined application and radio handover should follow the following steps, when the handover is initiated by UE due to radio conditions:

1. The source (serving) eNB radio controller, before initiating the radio handover (HO) process, informs the SCM-BS about its handover intention for the UEx.
2. SCM-BS makes sure that there are suitable VM and memory resources on the target eNB; if not, it looks for an alternative solution.
3. The SCM-BS communicates to SCM-UE the base station handover intention for a specific UE and the found platform (eNB) for the continuation of the offload.
5. Both SCM-BS and SCM-UE request the applications running on the eNB and UE platforms to stop.
6. Both SCM-BS and SCM-UE wait until the applications acknowledge that they stopped; SCM-UE updates SCM-BS about the stop of all offloaded UE applications.
7. SCM-BS informs the source eNB, when the execution of the VM also stopped, that the radio handover can start.
8. The source eNB requests the target eNB to start the radio handover procedures.
9. The radio handover takes place over the X2 interface
10. In parallel, the source SCM-BS announces to the target SCM-BS that the VM handover can start.
11. The application offload handover (VM context) takes place over the Z interface between the source SCM-BS and the target SCM-BS.
12. When the X2 handover is ended and the UE successfully communicates upon termination with the target eNB, the radio controller of the target eNB announces the termination to the SCM-BS in the target eNB.
13. The SCM-BS in the target eNB waits for the successful termination of the application VM context and data handover.
14. The SCM-BS in the target eNB announces to the SCM-UE and the SCM-BS in the source eNB that the handover of the offloaded application has ended.
15. The SCM-BS in the target eNB and the SCM-UE announce to the application to resume the execution.

6.3 *Optimal distributed offloading strategy for mobility management*

As a particular case of the optimization strategy proposed in Section 5.2.3.2.3 to assign each user to a base station and to a cloud, we can handle user mobility as follows. In case of computation offloading, handover comes to depend on both radio and computing resources (where virtual machines are running). Consider, with reference to Figure 155, a single user case $K = 1$, where a user (MUE 1) moves from a position near SCceNB 1 to a position close to SCceNB 2.

While moving, a conventional cellular system would perform a base station handover in order to associate that user to the best station, i.e. the station providing the higher SNR. However, if we consider the allocation of radio and computing resources jointly, we need to consider also a cloud handover. This means that if we switch the radio access from SCceNB 1 to SCceNB 2, but we keep the virtual machine running over SCceNB1, to avoid virtual machine migration, we need to take into account the delay for sending data over the backhaul from SCceNB 2 to SCceNB 1. Alternatively, we might consider switching both radio access point and serving cloud, but in such a case, we need to migrate the virtual machine.

The solution of this problem can be achieved as a particular case of problem $\mathcal{P}_{10}$, with $K = 1.$ Our experiments in this section are run under the following setup. We consider a MIMO network with a number of base stations equal to the number of clouds, i.e. $N_b = N_c = 4$. All transceivers are equipped with $n_T = n_R = 2$ antennas. The other system parameters are set as follows: $F_m = 2 \cdot 10^7$, $P_k = 10^3$ and $N_0 = 1$. In Figure 179 we show an example of final base stations assignment resulting as a solution of our algorithm, for $K = 6$ and $N_b = N_c = 4$. Note that the final base station selection tends to assign each user to its nearest base station, while distributing the computation among the clouds.

Indeed, even if we relaxed the binary values a_{knm} to be real values belonging to the interval [0, 1], in our numerical results we found that the indexes assigning a user to a base station tend to be very close to either 0 or 1, whereas the indexes assigning the user to a cloud tend to assume intermediate values. This means that the relaxed algorithm tends to a solution where each user accesses the network through a single base station, but it is served by, possibly, multiple clouds. This is indeed a quite interesting result.

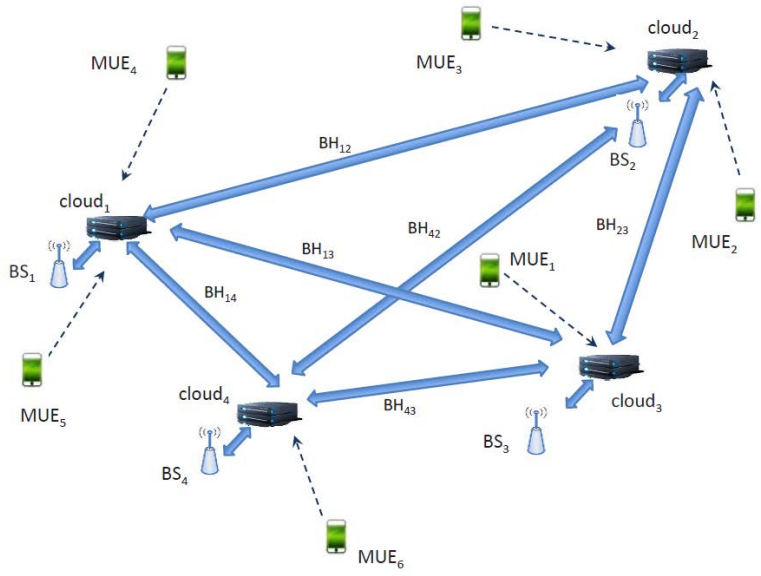


Figure 179. Example of user-base stations assignment.

As mentioned above, our optimization strategy provides also, as a by-product, a mechanism to handle hand-over, in case of low user mobility. As an example, with reference to Figure 179, we considered the case in which no users are moving, except user 1, who is moving from nearby the third base station towards the first base station. In Figure 180, we have plotted the energy consumption E_1 of user 1, while it moves, averaged over independent channel realizations, versus the distance d_1 between mobile user 1 and base station 3. In the same figure, we report the result of the optimal method solving a mixed integer programming problem, considering all possible combinations of the integer values a_{knm}, and the results of our suboptimal approach, based on successive convex approximations.

Different curves refer to different total numbers K of mobile users in the system. The two left and right end-points refer to the situation where the user is close to base station 1 or 3. We may observe that, as expected, the energy consumption increases as the user moves away from each base station and achieves its maximum when MUE is on the edge of two cells (middle point). But the most striking result, shown in Figure 180 is that the suboptimal approach provides results very close to the benchmark algorithm, which has exponential complexity. Finally, the slight increase of energy consumption with K is the result of an increased interference level resulting from an increasing number of users. The slight increase testifies that the joint optimization is taking care of the interference level properly.

Clearly, the association of a mobile user to a cloud depends on the radio load of each base station as well as on the computational load of each cloud. Selecting clouds non co-located with a base station may be a viable solution if the backhaul link between a base station and the cloud is sufficiently fast. To test this situation, we considered again the scenario of Figure 180, where user 1 is moving and the backhaul link between base stations 1 and 3 is either congested or relatively free. In Figure 181 we have plotted the average global energy consumption obtained by solving problem (219) in the two extreme cases where the backhaul link among the two clouds is congested or not. In the second case, the latency along the backhaul is sufficiently low to essentially enable each mobile to offload

computation to any cloud in the network. From Figure 181, we may observe how the backhaul availability brings, as expected, a network energy saving, as the energy consumption increases with the number *K* of active users.

In Figure 182 we report the average energy consumption in the presence of *K*=4 mobile users and a varying number of base stations N_b. We compare again, the proposed suboptimal method with the optimal benchmark. We can see that, as the number of base stations increase (denser deployment), the increased number of degrees of freedom yields a considerable energy saving. Furthermore, also in this case we can observe how the proposed algorithm provides results very close to the optimal ones.

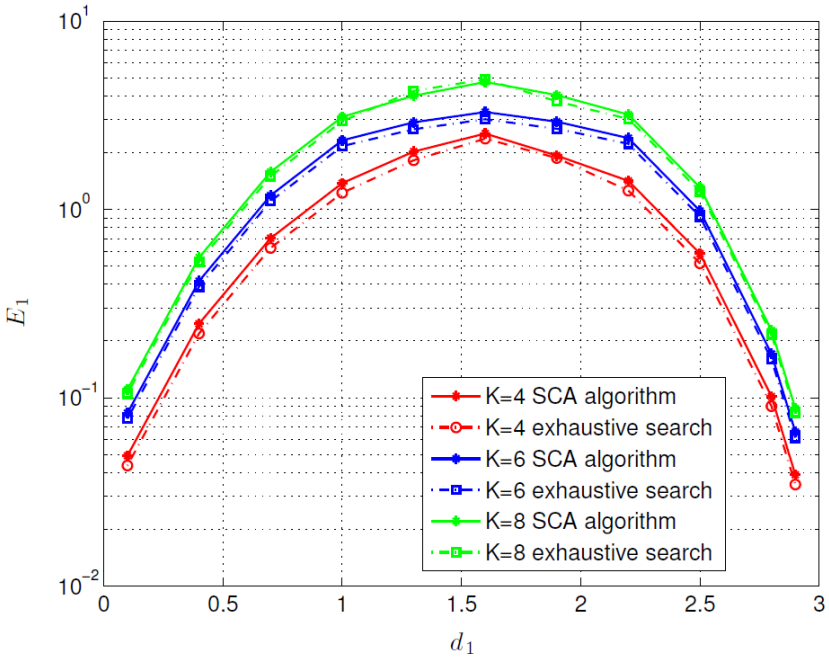


Figure 180. User energy consumption vs. d_1.

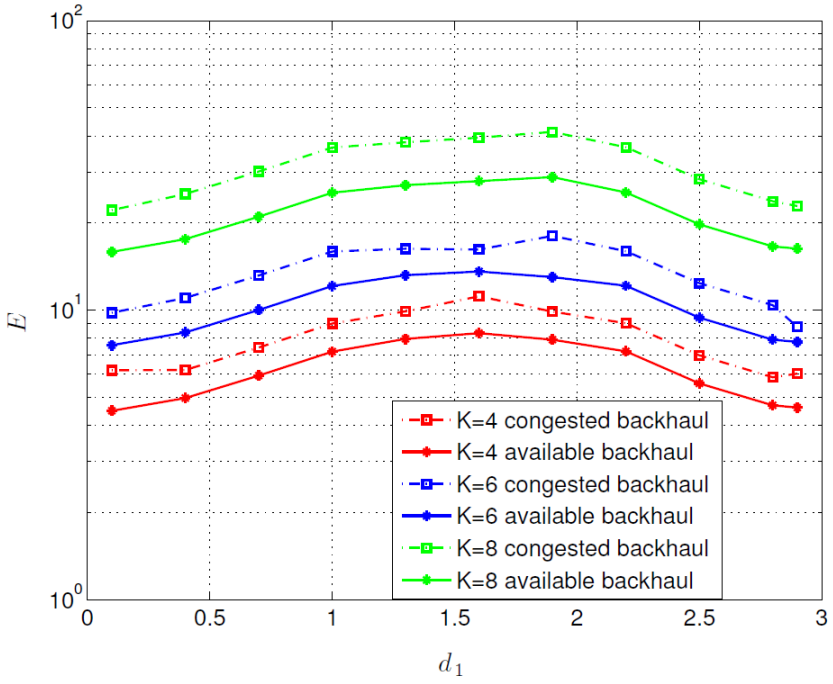


Figure 181. Global energy consumption vs. d_1.

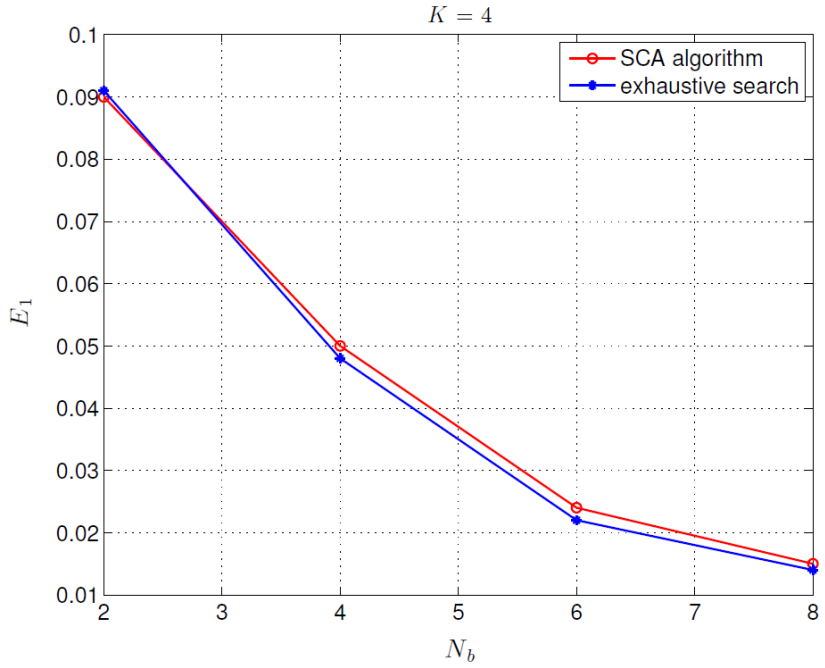


Figure 182. Energy consumption of user 1 vs. the number of base stations.

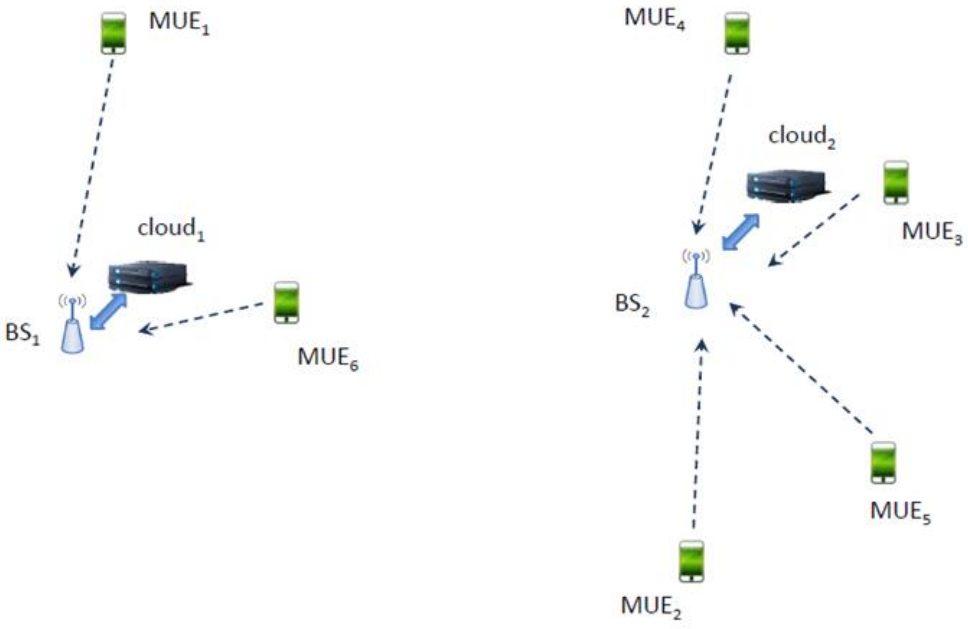


Figure 183. Example of network scenario.

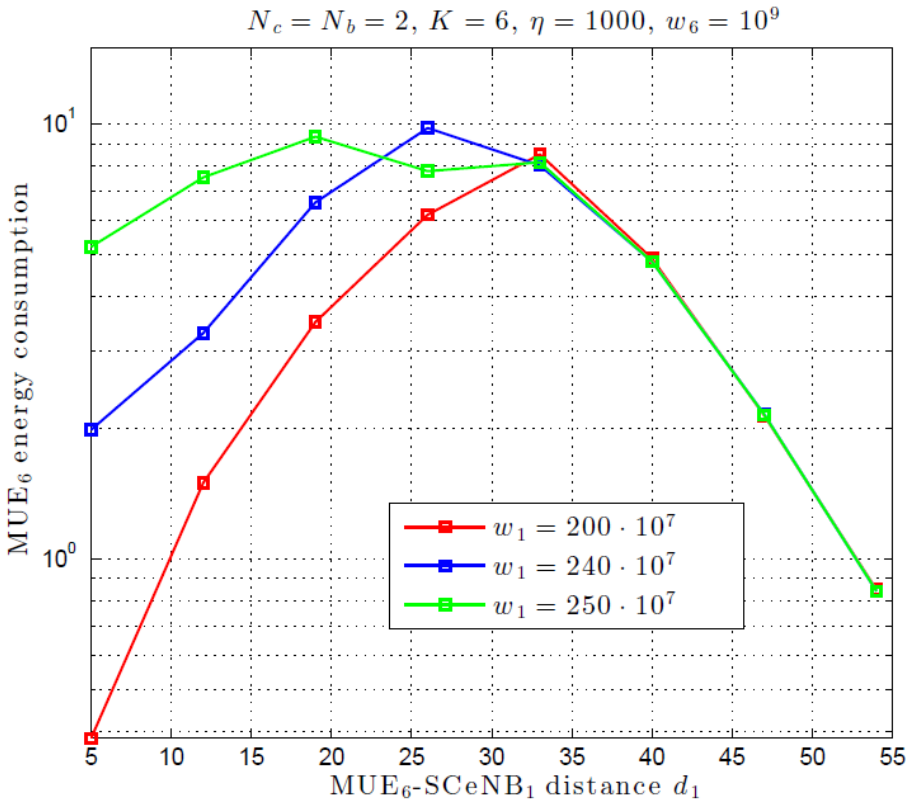


Figure 184. Mobile user 6 energy consumption.

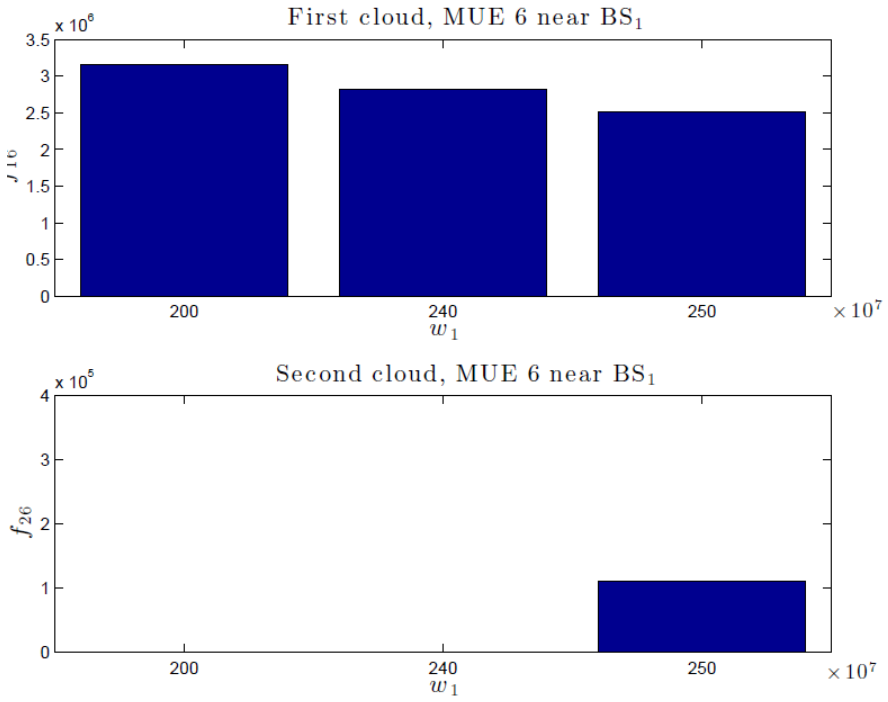


Figure 185. Computational rate assignement of MUE 6 when placed near BS 1: First cloud (top); Second cloud (bottom).

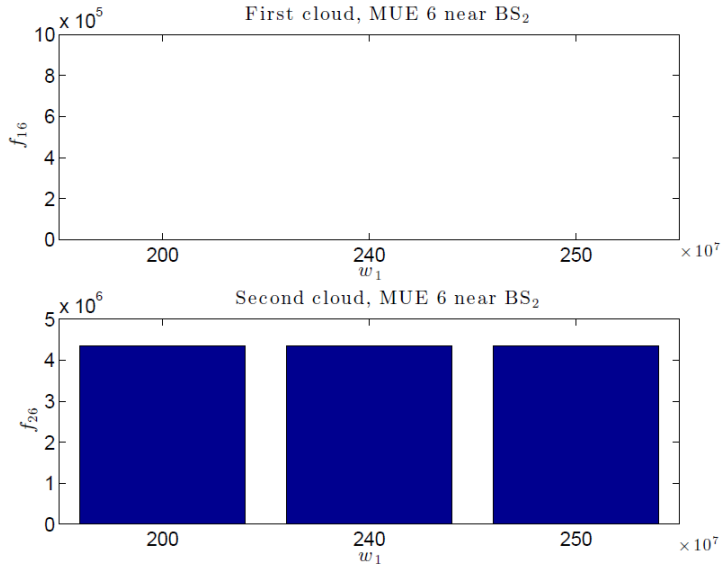


Figure 186. Computational rate assignement of MUE 6 when placed near BS 2: First cloud (top); Second cloud (bottom).

Furthermore to evaluate the impact of the cloud computational load on the optimal user-BS-cloud assignment we focus on the scenario depicted in Figure 183 where all mobiles are fixed and MUE 6 moves from BS 1 towards the second one. More specifically for all the MUEs, the number of transmitted bits needed to transfer execution is fixed and the number of CPU cycles to be executed is set as $\omega_2 = \omega_3 = \omega_4 = \omega_5 = 10^7$, $\omega_6 = 100 \cdot 10^7$ while the computational load of MUE 1 is increased from $\omega_1 = 200 \cdot 10^7$ to $\omega_1 = 250 \cdot 10^7$. This setting leads to a scenario where the first cloud tends to be heavily loaded with respect to the second one. From Figure 184 it can be noted that the energy consumption of MUE 6 tends to increase as the computational load of the first mobile increases since

the first cloud tends to allocate more computational resource to the most demanding mobile. On the other hand from Figure 185 it can be observed that when $\omega_1 = 250 \cdot 10^7$ the computation of MUE 6 is split between the two clouds being the first cloud unable to serve both users at the same time. Additionally in Figure 186 we report the computational rate of MUE 6 when placed near BS 2. As expected being the mobile user served by the unloaded second cloud the computational rate of MUE 6 tends to be independent of any variation of the first cloud load.

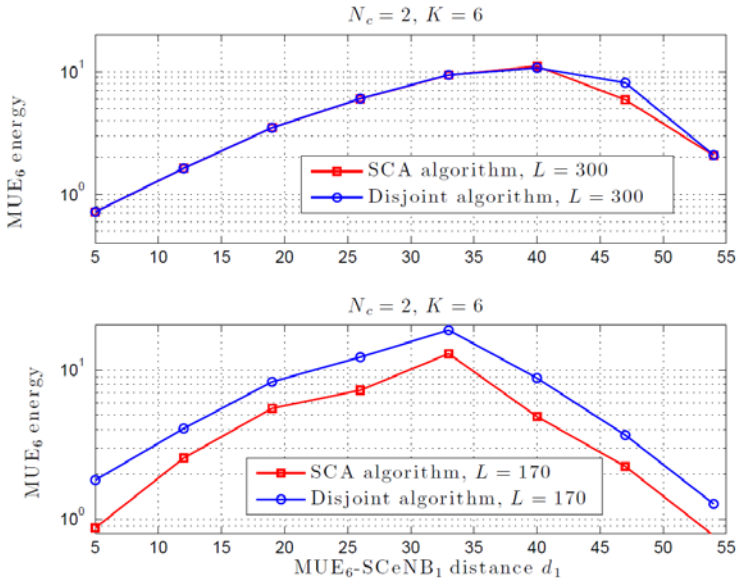


Figure 187. Comparison between joint and disjoint optimization strategy.

Finally in Figure 187 we show the performance gain of the proposed SCA joint optimization strategy with respect to a strategy where the optimization of the communication and computational resources is performed separately by fixing the computational rate proportionally to the computational load ω_k of each user. The analysis of the figure shows that the proposed SCA algorithm yields a performance gain as the delay constraint becomes more stringent.

6.4 *Summary of mobility aspects*

The mobility of users can lead to problem with delivery of computing results to the user. This requires to change handover paradigm from the purely radio motivated handover towards handover with awareness of cloud aspects. We propose three options how to handle mobility and guarantee results delivery to the mobile UE: VM migration, path selection and joint cell association and radio (precoding matrices) /computational resources allocation.

The first option is based on assumption that the serving SCeNB is also the cell, which performs computation for the UE. In this case, the computation is migrated together with the handover. A drawback of this solution consists in heavy load implied by the VM migration.

Contrary, the path selection algorithm does not change place of computation (the same SCeNBces perform computation disregarding handover) but new path is found after computation. Thus, the results are delivered to the UE through path with the shortest delay. In this case, the handover decision considers not only common radio channel quality but also time required for delivery of results back to the user and energy consumed data transmission/reception constrained with maximum delay tolerable for offloaded service.

In addition to before mentioned aspects, an offloading strategy for joint optimization of mobile user – base station-cloud assignment, and radio (precoding matrices) /computational resources has been described in Section 6.3. The optimal association of mobile users to base stations and clouds provides a mechanism for optimal instantiation of virtual machines and represents a new way to handle handover depending on channel, application parameters and backhaul state. The goal is to assign in a multi-cell interfering scenario with multiple clouds each user to a single base station-cloud couple, in order to minimize the energy consumption at the mobile side under power budget, latency and cloud computational capability constraints. The moving UE can perform offloading by accessing the most convenient base station and the most appropriate cloud for computation. With respect to the path selection algorithm (Section 6.1), the optimal offloading strategy in Section 6.3 is motivated by the fact that to properly handle handover according to the channel conditions and the server computational loads, a general framework has to be designed able to jointly optimize the radio and computational resources allocation, together with the MUE/BS/cloud assignment. On the other hand, this approach assumes to deliver for each user all offloaded data through the most convenient base station while path selection enables to split data according to the assignment of individual parts of offloaded code to the computing cells. Therefore, those two approaches complement each other.

7 RECOMMENDATIONS TOWARDS WP6

In the demonstration platform to be developed within WP6, a practical scenario has to be implemented and tested under realistic conditions as a proof of concept. Such implementation will require a number of simplifications in the techniques and strategies developed in other WPs from a more theoretical and research-oriented point of view. As far as WP5 is concerned, and more concretely task 5A1, the following recommendations and simplifications are suggested to be taken into account in the execution of WP6.

7.1 *Recommendations towards the simulation platform*

The simulation platform will take the abstraction models designed in this deliverable as the basis for the simulation implementation. It will get results from the integration of the radio part and the needed cloud aspects. To that end, the recommendations towards the simulation platform are the following:

- **Applications models:** applications can be very diverse. There can be applications that are clearly defined before they are executed (i.e., the number of computational cycles to be executed and the number of bits to be processed are known a-priori) and other ones where the information to be processed is generated in real-time as the user interacts with the UE. In that sense, it is recommended that a limited number of concrete applications are selected representing very clear models of applications in WP6. These models should be simple and representative enough to validate the proof of concept.

- **Wireless communication abstraction model:** the offloading optimization strategy has to take information from the channel state (i.e., from the PHY layer) to decide which is the best offloading strategy to be followed. Describing completely the wireless communication channel from upper-layers is quite complicated and the solution for this is not unique. In that sense, it is important to work under a simple abstraction model of the PHY layer that reduces significantly the number of parameters that represent the channel state but that, at the same time, captures and describes the channel state in a really representative way. A possible approach to do this relies on the fact that the standard identifies a set of finite possible values of the communication rate according to the pre-defined MCSs. Thanks to this, a simple abstraction model of the PHY layer could be based on just a table with a list that indicates for each possible MCS which would be the energy spent by the UE in the communication and the time needed for such communication. This will reduce the information to be exchanged among the entities involved in the offloading decision process. At the same time, a PHY model with a higher accuracy should be used for modelling the radio communications among UE and SCeNBce in order to provide effective results, especially considering that this aspect cannot be evaluated with the proof-of-concept of 6A2.

- **Optimization criteria:** the optimum offloading strategy will depend, of course, on the optimization criterion that is taken and according to this, different strategies are possible. Thus, it is recommended that in WP6 a limited set of very clearly defined and simple offloading criteria are selected to be representative enough and to be able to compare the results according to such criteria.

- **Optimization solution:** finding an optimum solution to the general optimization problem associated to the offloading process may be quite complicated because of the high number of optimization variables involved and possible solutions. Thus, it is recommended that only a limited finite number of representative offloading strategies are considered as possible solutions of the optimization problem. This will allow implementing the optimization problem just as a procedure in which each of these limited number of possible offloading strategies are tested and then, the best one according to the pre-defined optimization criterion is selected. Considering the inevitable constraints imposed by the simulation platform and the

implementation limitations arising from the algorithms themselves, like the usage of mathematical tools not available in the simulator, the offloading strategies suggested for the evaluation in 6A1 are the "data-oriented" with the optimization of the energy-latency tradeoff (Section 5.2.1) and the "Task-oriented" offloading with joint optimization of energy consumption and computational rate under power and latency constraints (or vice-versa) in SISO multi-user single cell case (Section 5.2.3.2.1).

- **Implementation of offloading decision**: As far as the implementation of the offloading decision is concerned in reference to data-partitioned applications (see section 5.2.1), in the theoretical approach described in this deliverable it is assumed that any partition and any UL rate is possible. In practice, only some partitions of the data to be processed are possible. In addition, only some discrete values of the UL rate (as defined by the standard) can be applied. That means that, from a practical point of view, the optimization of the problem described in this deliverable can be performed just as the evaluation of the energy or latency consumption associated to each possible combination of a partition and an UL rate, which constituted a discrete and finite set of possibilities, simplifying the implementation.

- **Mobility support**: in order to evaluate the solutions proposed by the project in a realistic scenario, it is recommended that mobility is taken into account in the simulator. The mobility of the UEs and the possible handover will allow the evaluation of the power control algorithm, the path selection algorithm and the cost of migrating the Virtual Machine towards the new serving SCeNBce.

- **SCC architecture**: from the simulator point of view, both ways of architecture implementation are equivalent but the one with protocol translation (presented in Section 4.4.1) is recommended due its compatibility with device addressing in existing networks.

7.2 *Recommendations towards real-world prototype*

In this section we summarize major recommendations for simulator and prototype of the SCC system. The proof of concept will develop a real-world prototype of the SC-cloud system. It will focus in the cloud aspects, as the implementation of the radio channel is carried out via WiFi, and therefore the demonstration of the radio aspects is not feasible (radio aspects are demonstrated in the simulation platform). The recommendations towards the proof of concept are the following:

- **Application models**: As for the application, the complexity of implementing real applications running in real mobile devices leads to the consideration of just a few application types. We recommend to developed following apps to demonstrate various aspects of the SCC system:
    - Virus scanning
    - Augmented Reality
    - Video processing

- **Offloading mechanism**: In the same way, the recommendation towards the prototype is to implement "data-oriented" offloading mechanism with the optimization of the energy-latency tradeoff (section 5.2.1). This offloading mechanism will have the following characteristics:

    - The offloading decision will be taken in the UE side.
    - The necessary information will flow from the SCM to the SC and vice-versa.
    - The offloading modules are distributed in the UE and the SCM as required for the optimal offloading decision.
    - We recommend to consider the possibility to include also a comparison with a system where the offloading is being taken in the SCM side. In this case, the offloading modules will be distributed on the UE, SCM, and SCC side.

- **Optimization solutions**: The optimization solutions taken in the prototype will be based mainly in the outcomes of [TROPIC-D52].

- **SCC architecture**: For real-world hardware prototype, the architecture with centralized SCM and with protocol translation is recommended for implementation as it does not require to change way of addressing of devices.

8 CONCLUSIONS

This document presents integration of work carried out in WP2, WP3 and WP4 into a single framework combining cloud computing and mobile communications. The deliverable defines integration and interconnection of both formerly independent areas in order to enable joint optimization of radio and cloud. We design functional scheme of the SCC system and define all modules and interfaces for proper cooperation among all entities. Based on this, architecture originally proposed in D22 is updated to the final architecture suitable for efficient management of the SCC. The architecture enhances common LTE by the SCM managing all computation with respect to the status of the network and requests of the users. In addition, also SCeNB is enhanced by the SCC-GW, which interacts with the SCM and mediates execution of commands from the SCM. The SCC-GW also separates SCC traffic from conventional LTE traffic to minimize delay imposed by data transmission as SCC traffic does not need to be delivered to P-GW. Besides hardware modification of network entities, also software of UEs and SCeNBs must be updated to enable communication with the SCM and for application offloading.

In addition to detailed analysis of centralized architecture, we show potential future gain by implementation of advanced architectures with partly or fully distributed SCM. This enhancement can bring reduction in signaling overhead and delay by roughly 40% and 60%, respectively. The project final architecture is designed in the way that extension towards advanced architectures can be done easily by moving a part of the centralized functionalities into related blocks already designed within base stations.

The computation is offloaded from the UE to the SCC if it brings advantage to the user in terms of energy consumption or latency. The offloading can be done either partially (i.e., a part of computation is done locally at the UE and part remotely at the SCC) or fully (whole computation in the SCC). To decide whether to offload complete computation, a part of the computation or not offload at all, several algorithms are designed. Those algorithms differ among other by means of metrics considered for decision, by radio and computing scenarios, type of application partitioning and by suitable architectures. All algorithms prove energy saving and latency reduction for users. With respect to the final architecture and complexity of algorithms, following two algorithms are recommended for mutual comparison by system level simulator in WP6: the “data-oriented” with the optimization of the energy-latency tradeoff (Section 5.2.1) and the “Task-oriented” offloading with joint optimization of energy consumption and computational rate under power and latency constraints in SISO multi-user single cell case (Section 5.2.3.2.1). For evaluation purposes and for system level simulator in WP6, this document also provides summary of recommended models of the SCC elements from literature and backhaul model derived from the real network.

To enable application offloading also for fully mobile users, we investigate two different approaches: migration of VMs according to users’ movement and update of the users association to the cells to choose the most efficient cell through which the data between the UE and computing cells is exchanged. The VM migration suffers from limited backhaul as, typically, whole VM that needs to be migrated is of larger size than the computation result itself due to migration of input data, memory status, intermediate computation results, etc. On the other hand, update of serving cell might lead to redundant handovers on the radio link.

ANNEX I – SYSTEM REQUIREMENTS

Area	Area / Functional Domain	Requirement ID	Requirement statement	Requirement description	Priority
Cloud	SCM	TROPIC_001	Clustering	A set of small cells must be organized in a cluster that will be managed by one SCM.	High
Cloud	SCM	TROPIC_002	Cluster definition awareness	The SCM must be aware of the small cells belonging to its cluster and their status at any time.	High
Cloud	SCM	TROPIC_003	VIM	The Virtual Infrastructure Manager (VIM) must communicate with the hypervisors in order to deploy an application on the small cells cloud and to manage distributed applications at runtime.	High
Cloud	SCM	TROPIC_004	SCM access to VM	The SCM, through the VIM, must communicate with the hypervisor running on a base station. The hypervisor must inform the VIM of VM states.	High
Cloud	SCM	TROPIC_005	Monitoring	It must be possible to monitor VMs at runtime on a small cell.	High
Cloud	SCM	TROPIC_006	Capability scheduling	The SCM must be able to schedule capabilities across SCceNBs on policies; For example, if a femtocell detects that one VM is collapsing due to a peak of demand, it must be possible to elastically launch an additional VM.	High
Cloud	SCM	TROPIC_007	Cluster management	The SCM must be able to manage users and VM across multiple small cells.	High
Cloud	SCM	TROPIC_008	VM deployment in SC	The SCM, on the basis of the network information must be able to deploy a VM in a specific small cell.	High

Cloud	SCM	TROPIC_009	Optimization of communication mechanisms	Communication between VMs installed in SCceNBs or between VMs and the SCM can be delivered via wireless interface or by combination of SCceNBs backhaul and radio links. To that end, control mechanisms for efficient selection of appropriate access SCceNB via radio channel and fast distribution of data among SCceNBs involved in computation must be found. The algorithm must take into account the state of backbone and radio links of individual SCceNBs	High
Cloud	SCM	TROPIC_010	Self-environment	A cluster must be an autonomous system: it must include self-management, self-healing, self-reporting, self-provisioning capabilities.	High
Cloud	SCM	TROPIC_011	Elasticity	When the SCceNBs in a cluster are not able to address a spike of demand then these systems must be able to burst to a higher capacity cluster (Elastic capabilities – Bursting)	Medium
Cloud	SCM	TROPIC_012	Scalability	Given the voluntary and temporary nature of resources' contribution to the system, it must be able to dynamically adapt to available resources (on the fly), which means that it must be scalable.	High
Cloud	SCM	TROPIC_013	Fault tolerance	In case of lack of power supply the system should be able to implement recovery procedures (backup copies, for example).	High
Cloud	SCM	TROPIC_014	Dynamicity	As resources are not dedicated, the system must be able to manage unexpected loss of contributed resources. Dynamic membership implies the need to create replicas on multiple servers when data is first stored and	High

				the need to compensate for lost replicas when femtos fail or leave.	
Cloud	SCM	TROPIC_015	Recovery	In case a user decides to turn off a femtocell, the system must be able to recover ongoing processes to continue the operation.	High
Cloud	SCM	TROPIC_016	Execution monitoring	The SCM should keep the status of the execution in order to react as quickly as possible to failures in the infrastructure.	High
Cloud	SCM	TROPIC_017	External cloud	The small cells cloud could keep a connection open to the back-end service in order to handover the connection if a major failure happens.	Medium
Cloud	SCM	TROPIC_018	Deployment policies	The SCM must be able to support different deployment polices (based on network context, based on host capacity,…)	High
Cloud	SCM	TROPIC_019	Heterogenity	SCceNBs might have different technical features but all of them must have the same functional and technical interface.	High
Cloud	SCM	TROPIC_020	Contextual information	Information taken into account from deployment to execution and during execution must include cloud aspects (availability, capacity, etc.) but also backhaul and radio information (current serving femtocell,etc.)	High
Cloud	SCM	TROPIC_021	Joint optimisation	There are two QoS types to consider that have to be aggregated: the radio QoS and the cloud performance for the running application. The optimisation must be joint.	High
Cloud	SCM	TROPIC_022	Transparency	All normal operational activities at runtime must be transparent for the end user.	High

Cloud	SCM	TROPIC_023	Reason of failure notification	When anomalies occur at runtime that cannot be solved, the user must be informed by a message on his device indicating a clear reason of failure.	High
Cloud	SCM	TROPIC_024	Multi application	One user may request running several applications at a time.	High
Cloud	SCM	TROPIC_025	Multi instance	One user may request several instances of the same application running at a time.	High
Cloud	SCM	TROPIC_026	Service monitoring	A service must provide the means to get monitored.	High
Cloud	SCM	TROPIC_027	Resource liberation	When a SCceNB has finished a task, it should free up resources for other applications/users.	High
Cloud	SCM	TROPIC_028	Authentication	Users must be authenticated on the system for these to accept the provision of a service (ideally, the user identifier should be enough for the whole cluster).	High
Cloud	SCM	TROPIC_029	Authorised users	When a femtocell owner has authorized a guest user, this user will be able to use the cluster in the same conditions than the owner.	High
Cloud	SCM	TROPIC_030	SCM invocation	When a user requests to execute an application over the small cells cloud, the serving cell must invoke the SCM so that the application is automatically deployed in an optimal way (the service QoS is compliance with the user requirements).	High
Cloud	SCM	TROPIC_031	OVF	A service deployed must have an OVF (Open Virtualisation Format) in order to be able to apply elasticity if needed in the cluster.	High
Cloud	SCM	TROPIC_032	In-bursting	The small cells cloud system must offer VM transfer mechanisms from the backend service datacenter to the small cells cloud. A Service Datacenter must be able to provide VMs to the local	Medium

				small cells clouds managed by the SCMs	
Cloud	SCM	TROPIC_033	Cluster interaction	Small cells clouds must provide mechanism to federate two clusters.	Medium
Cloud	SCeNBe	TROPIC_034	Communication	Communication between HeNBce and SCM must be initiated from HeNBce only. HeNBce can be in general hidden behind NAT/PAT, thats why secure communication can't be established from SCM side. After creating an encrypted tunnel between HeNBce and SCM, communication can be initiated from any side.	High
Cloud	SCeNBe	TROPIC_035	Communication	SCM must have a public IP address.	High
Radio	SCeNBce	TROPIC_036	Global SCUE monitoring	SCeNBces periodically report to SCM the total number of UEs connected to it and the medium radio link quality of the cell.	Medium
Radio	SCeNBce	TROPIC_037	Radio link latency	Each small cell is able to measure/estimate its own latency in radio communication exploiting knowledge of scheduling policy (also related to frame structure) and internal resources	high
Radio	User equipment	TROPIC_038	Radio link quality measurement	The user equipment ranges its power and gets feedback from the serving small cell about the quality of the uplink channel in terms of supported transmission rate.	High
Backhaul	SCeNBce	TROPIC_039	Backhaul monitoring	SCeNBces test and report to SCM its link quality status, in terms of throughput and latency	High

Table 42. Requirements on the SCC system.