Document Properties

Document Number:	D 4.3
Document Title:	**Final assessment on Connectivity Management for very dense scenarios**
Document Editor:	Antonio de la Oliva (UC3M)
Authors:	Arianna Morelli (INCS), Vincenzo Mancuso (IMDEA), Arash Asadi (IMDEA), Christian Vitale (IMDEA), Erick Bizouarn (ALBLF), Engin Zeydan (AVEA), Ahmet Serdar Tan (AVEA), Claudio Bottai (INCS)
Target Dissemination Level:	PU
Status of the Document:	Submitted to EC
Version:	1.0

Production Properties:

Reviewers:	Arianna Morelli (INCS) & Vincenzo Mancuso (IMDEA)

Document History:

Revision	Date	Issued by	Description
1.0	2015-04-30	UC3M	First EC submission

Disclaimer:

This document has been produced in the context of the CROWD Project. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under grant agreement n° 318115.

Abstract:

This document reports on the Final assesment on Connectivity Management for very dense scenarios and corresponds to the last deliverable of the WP4 of the CROWD project. In this deliverable we focus on presenting the final results of the research activities performed within WP4. The activities reported in this document regard to access selection algorithms, connection management, information aggregation and distribution and finally our SDN-based approach for mobility management.

Keywords:

connectivity management, traffic management, mobility, distributed mobility management

Contents

List of Figures

List of Tables

List of Project Partners

Name	Acronym	Country
Intecs S.p.A. (*coordinator*)	INCS	Italy
Alcatel-Lucent Bell Labs France	ALBLF	France
Avea Iletism Hizmetleri A.S.	AVEA	Turkey
Fundacion IMDEA Networks	IMDEA	Spain
National Instruments	SIG	Germany
Universidad Carlos III de Madrid	UC3M	Spain
Universitaet Paderborn	UPB	Germany

List of Acronyms

3GPP Third Generation Partnership Project

ANQP Access Network Query Protocol

AP Access Point

API Application Programming Interface

ARP Address Resolution Protocol

BS Base Station

CAP Combinatorial Auction Problem

CDH4 Cloudera Distribution Including Apache Hadoop

CLC CROWD Local Controller

CQI Channel Quality Indication

CRC CROWD Regional Controller

CROWD Connectivity management for eneRgy Optimised Wireless Dense networks

CSV Comma Separated Values

CTS Clear to Send

D2D Device-to-Device

DAD Duplicate Address Detection

DB Data base

DCF Distributed Coordination Function

DensNets Dense Networks

DHCP Dynamic Host Configuration Protocol

DMM Distributed Mobility Management

DNS Domain Name System

eNB Evolved NodeB

ESS Extended Service Set

ESSID Extended Service Set ID

FEM Flow Extraction Manager

GGSN Gateway GPRS Support Node

GNU GNU is not Unix

GO Group Owner

GTP GPRS Tunnel Protocol

GTP-U GPRS Tunnel Protocol User

GTP-C GPRS Tunnel Protocol Control

GW Gateway

HDFS Hadoop Distributed File System

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IEEE Institute of Electrical and Electronics Engineers

IGMP Inter Group Management Protocol

IPv4 Internet Protocol v4

IPv6 Internet Protocol v6

LA Location Area

LAC Location Area Code

LTE Long Term Evolution

MCS Modulation and Coding Scheme

MIH Media Independent Handover

MIIS Media Independent Information Service

MN Mobile Node

OFDM Orthogonal Frequency Division Multiplexing

OMNIRAN Open Mobile interface for omNI-RANge networks

OS Operating System

PCAP Packet Capture

PDN Packet Data Network

PDP Packet Data Protocol

PoA Point of Attachment

RB Resource Block

REST Representational State Transfer

RNC Radio Network Controller

RSS Received Signal Strength

RTS Ready to Send

SAC Service Area Code

SDN Software Defined Network

SGSN Serving GPRS Support Node

SINR Signal to Interference plus Noise Ratio

SLAAC Stateless auto-configuration

SNMP Simple Network Management Protocol

SNR Signal to Noise Ratio

SSID Service Set ID

TCP Transmission Control Protocol

TEID Tunnel Endpoint Identifier

UE User Equipment

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunication System

VFS Vertical First Scheduling

VHO Vertical HandOver

WiFi Wireless Fidelity

WLAN Wireless Local Area Network

WP Work Package

WP8 Windows Phone 8

WWAN Wireless Wide Area Network

XML Extensible Markup Language

Executive summary

The following document presents the consolidated view of the Connectivity management for eneRgy Optimised Wireless Dense networks (CROWD) Connectivity Management (WP4) functionality and its full specification. All the specification and architectural work performed in this WP was finalised and presented in D4.2. Therefore, in this deliverable we focus on presenting the final results of the different research activities performed in WP4. Since much of the research ideas presented in this document have been previously discussed in D4.1 [1] and D4.2 [2], we have opted for a compact presentation of the results, having first a brief overview of the research topics with further detailed explanation included in the document as appendix. This approach has been used for the finished work already submitted for publication and standardization activities, while more implementation related activities and on-going work is presented in the body of the deliverable. A brief explanation of each of the sections composing this document can be found in the following:

- Connectivity Management: this section is devoted to briefly explaining the Connectivity Management approach followed by the three most relevant smartphone operating systems. The presented version of this study is an extended version of D4.2 [2] in which we include iOS8 and Android 5.
- Software Defined Network (SDN) Distributed Mobility Management: this section is devoted to the experimental evaluation of the network- and host-based versions of the SDN-based Distributed Mobility Management (DMM) solution proposed in this project.
- Energy efficient access selection: this work continues and finalises the work introduced in D4.2 [2]. Through the results presented in this section a node is provided with algorithms to decide the best network to connect to considering energy savings.
- EmuLTE: emulation of dense Long Term Evolution (LTE) networks; this section presents the final results of EmuLTE, the platform used by CROWD to test mobility scenarios and the correctness of the controller code.
- User mobility model implementation: this section presents the novel mobility model implemented in EmuLTE. This is an extension of the model used by EmuLTE as reported in previous deliverables.
- Mobile traffic characterization: this chapter presents a first version of the on-going work on the analysis of real operator traces to develop a model for the optimization of their network using DMM.
- Information aggregation and distribution: this chapter finalises the contribution on Information Aggregation and Distribution, reporting about the implementation of an SDN-based reporting mechanisms which is then used to select the best access possible, based on Access Network Query Protocol (ANQP).

Key contributions

The main technical contributions of this deliverable can be summarised as follows:

- Finalisation of the work on the analysis of the Connectivity Management of Android, iOS and Windows Phone. Thanks to the insights gained while doing this analysis we have been able to propose modifications to the behavior of the Connectivity Management for Dense Networks (DensNets).
- Finalization of the work on the DMM SDN-based proposal. This deliverable presents the experimental analysis of the mobility solution. A paper including the infrastructure on demand part and the NS3 hook for the LTE interface is been prepared.
- Finalisation of the energy efficient access selection optimization. The deliverable also includes a novel access selection mechanism that improves the efficiency in the use of terminals batteries, if compared with state-of-the-art solutions.
- This document also includes the final description of how EmuLTE is able to emulate mobility mechanisms to validate controller behavior.
- One of the key aspects of the mobility management in CROWD corresponds to the information gathering and distribution. In this document we present how the CROWD network is able to gather information from the different Access Points and use it to influence the mobility of the user.
- We also report on this document the initial work on the processing of real life traces of the AVEA network. This traces will be used to analyse the benefits of applying a DMM approach in their network.
- Finally, this document also report on two important standardization activities performed during the last period on WP4. We have actively contributed to the development of the SDN Reference Model for IEEE 802 technologies, which is been defined in IEEE 802.21cf.

The key contributions to the research (contribution type P) and standardisation (contribution type S) communities resulting from the work performed within the context of this deliverable are shown in the table below.

Summary of Research (P) and Standardisation (S) contributions

Contributions	Type	Outline	Organisation
Tackling the increased density of 5G networks; the CROWD approach	P	Presents latest results of all CROWD WPs	Accepted in 5G Arch Workshop (VTC 2015)
How does my smartphone manage network connections?	P	Presents the analysis of the connectivity management as defined in this document	Submitted COMCOM
SDN DMM Approach	P	Paper under preparation	To be submitted
IEEE 802.1cf SDN Chapter and Reference model	S	UC3M form part of the task group and is leading the SDN work	IEEE 802.1cf
Technical edition of the draft standard with respect to group management	S	Group management aims to optimise the handover of multiple terminals at the same time	IEEE 802.21d

1 Introduction

This deliverable corresponds to the last deliverable of the WP4 of the CROWD project, the WP4 specification and architectural work of the Work Package (WP) was finalised in D4.2 [2], hence this deliverable is focused on the reporting of the last results of the resurch activities performed in the WP. This includes the final version of some works which have been previously reported and the first description of the works started in the last period of the project. Connectivity management is a very wide term which covers different operations such as access selection, initial attachment, access monitoring, access re-selection or handover management among others. All these topics constitute a very significant part of the research activities that we have reported in previous deliverable and the key focus of this document. In particular, this deliverable reports on (i) understanding and characterising connectivity management in commercial devices, identifying potential optimizations for the CROWD scenario; (ii) assist the process of network selection, handover and network attachment through information gathering and distribution systems enhanced for the dense environment; (iii) design and evaluate experimentally a mobility solution based on the SDN paradigm to bring into action the architecture envisioned in the context of the CROWD project; (iv) analyse real life traffic traces to assess the benefits of the application of DMM in a real operator network and finally (v) report the standardization work performed within Open Mobile interface for omNI-RANge networks (OMNIRAN) (IEEE 802.1cf) in which the CROWD architecture has been pushed to be included in the SDN Network Reference Model being discussed in this Task Group.

The document is structured in two differentiated parts, document body and appendix. While the document body includes brief explanations of the research work done, the appendix provides with the full detail on each of the research items. Note that implementation-based research and the explanation on the initial analysis of the traces is directly included in the body of the document for simplicity.

2 Research activities

This section presents summaries for the different research activities performed in the project (we present the whole contribution in the Annex 4). We have divided the different research activities in the three main blocks composing WP4: *i*) Mobility Characterization, *ii*) Mobility Management and *iii*) Access Selection.

2.1 Mobility Characterization

2.1.1 EmuLTE: emulation of dense LTE networks

In order to test the correct behaviour of the CROWD Local Controller (CLC) and the applications on top of it, we developed an LTE emulator, emuLTE,able to configure very dense LTE network scenario. The emuLTE is made by, in addition to scheduler, main and internal database, several modules that manage topology of the network, interference measurement and acquisition, mobility of the users and communication with external executable components. Regarding the mobility model we started studying a generic model where completely random values for direction angle, way and speed have been considered. Every 50 msec the emuLTE scheduler activated the mobility module which, based on the set parameters (speed, angle and way) computes the updated position for each users in the emulator. Once a new position is determinated, the interference value is calculated in the specific module and all information are stored in an internal database. In fact, as the User Equipments (UEs) move, the related Signal to Interference plus Noise Ratio (SINR) and consequently, the Channel Quality Indication (CQI) change depending on how far the Base Station (BS) is, and how many neighbour Evolved NodeBs (eNBs) the user detects and how many UEs are nearby. All the information detected are sent to the CLC by means specific interface using User Datagram Protocol (UDP) protocol. In order to implement a more realistic mobility model, the mobility module has been replaced by a new mobilty management module and a communication interface module able to communicate with an external application, PedSIM (see Section 2.1.2) a microscopic pedestrian crowd simulation system, suitable for crowd simulation where the interest is related to the high density pedestrian mobility. The tool allows to use pedestrian mobility in every external software. Fixed some characteristics, like starting coordinates, ending coordinates and respective tolerance, obstacles in the delimited area under test, PedSIM provides the UEs positions that change due to the mobility. The flow of the process in emuLTE is shown in Figure 2.1. The grey modules are the existing modules, while the blue indicate the updated version.

During initialization phase, emuLTE starts an UDP socket and reads from a configuration file the address and port through which it will be able to communicate with gCROWD. gCROWD is a customisation of PedSIM for CROWD tests. Currently address and port to be used are:

```
address=127.0.0.1
port=12666
```

The module mobility management replace the old mobility module and gCROWD communication is a new element that allows the communication between emuLTE and PedSIM. PedSIM starts a UDP server able to answer the request coming from emuLTE. The messages exchanged are made by a 4 bytes header (t_xfUEsH) and a payload (one or more t_xfUE) as follows:

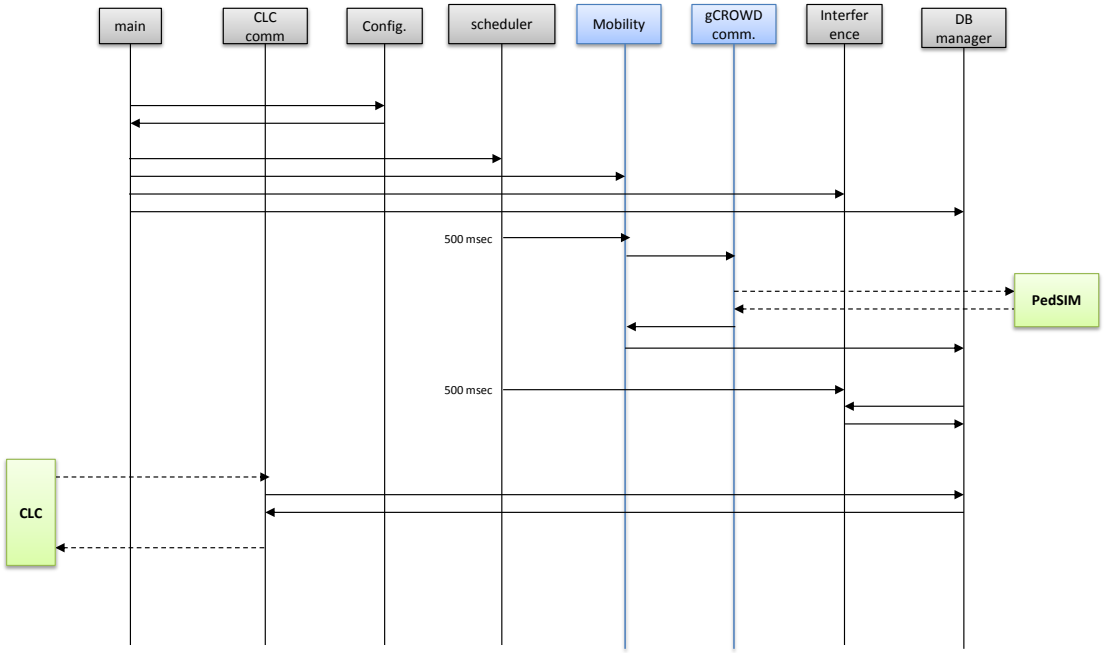

Figure 2.1: flow process emuLTE.

```
HEADER

typedef struct
{

  t_u8  u8Type; /* IF_UE = 34 */

  t_u8  u8Res;  /* not used   */

  t_u16 u16Cnt; /* number of UE struct that follow */

} t_xfUEsH;


typedef struct

{

  t_u16 u16Id; /* UE id */
  t_u16 u16PosX;
  t_u16 u16PosY;

} t_xfUE;
```

Periodically, every 500 msec, the emuLTE sends a request to the UDP server in PedSIM which answers as above.

The data received from PedSIM are managed by mobility management module, Data base (DB) manager module and interference module as in the previous version. Finally, the information related to the eNBs,UEs and CQI are sent to the CLC which runs specific algorithm or forward them to the higher layer applications for taking decision to be enforced in the emuLTE network.

2.1.2 User mobility model implementation

PedSIM is a microscopic pedestrian crowd simulation system particularly interesting for pedestrian density (refers to the PedSIM official website[1]). The tool is able to work with external software like emuLTE. We used PedSIM in CROWD context to obtain a more realistic mobility model and thus, extracting the UEs position we can test the behaviour of the CLC and components in CROWD architecture in realistic scenario where the UEs move with a velocity in a range 0.6-1.5 m/s.

We completely adopted the model used in PedSIM. For each UE, the related speed is made by

$$m_i \frac{dv_i}{dt} = m_i \frac{v_i^0 - v_i}{\tau_i} + \Sigma_{j \neq i} f_{ij} + \Sigma_W f_{iW}; \tag{2.1}$$

where:

m_i is the mass of the UE, v_i is UE velocity, v^0 is UE desired velocity. f_{ij} and f_{iw} are defined in the *social force model paper* [3].

PedSIM provides libraries that we modified and used according to the test we performed. When PedSIM starts, keeps information about the number of UEs, their characteristics and the obstacles in the area from an Extensible Markup Language (XML) file which charecterize the scenario. An example is provided below, where the *waypoints* are the coordinates of the destination, plus a tolerance, the *agent* means group of UEs, for each of them the starting point and several waypoint are provided, *obstacles* are elements that encumber the mobility.

PedSIM also provides a graphical user interface that we lightly customised for CROWD scope and renamed as gCROWD showed in Figure 2.2, where the green bullets are the UEs while the grey bullets, indicated by the red circle, are obtacles.

2.1.3 Mobile traffic characterization

2.1.3.1 Introduction

Identifying the structural patterns in the data traffic is of high importance for mobile operators in order to apply optimal mobility management techniques within their network. In this context, aiming to define the characteristics of mobile traffic in present networks and to contribute to the framework of traffic offloading in the CROWD scenario, a flow information extraction tool for real traffic traces is developed.

Mobile network traffic has a highly complex and massive structure making it though to analyse and reveal structural patterns. One minute of mobile traffic data may take up on average 1 TBs of space for a mobile operator consisting of 10 to 20 million subscribers, similar to AVEA. Big data anaylsis tools, such as Hadoop clusters, combined with several techniques have to be deployed in order to analyse and extract patterns from real mobile traffic data.

[1] `http://pedsim.silmaril.org/`

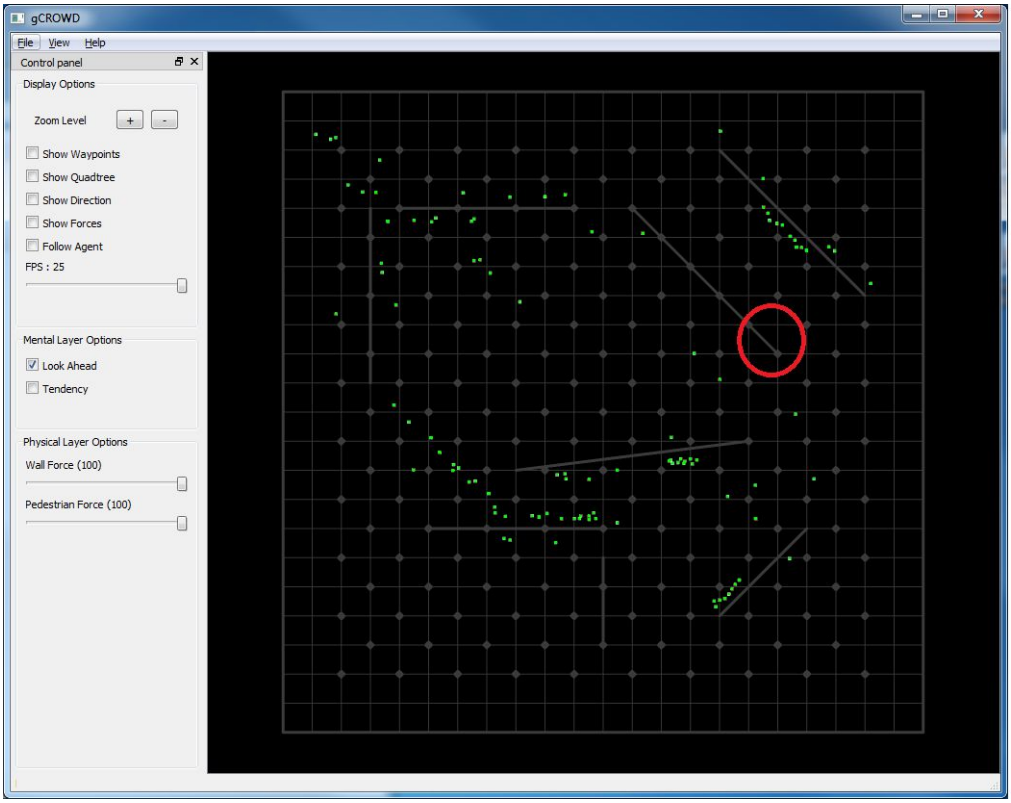

Figure 2.2: gCROWD: PedSIM for CROWD.

Mobility requirement analysis necessitates the extraction of handover related information which can be tracked in radio access and core network nodes. Accessing handover related information from the radio access nodes is difficult since the amount of probes that needs to places into the system can be large. Moreover, Radio Network Controller (RNC) log data is too hard to extract due to unavailability of appropriate tools for further mobility analysis.

In this work, handover related information observed in the core network nodes, mainly on the Gn interface, is extracted and analyzed. More specifically, the Update Packet Data Protocol (PDP) Context [4] control signal messages triggered by the Routing Area change of a mobile terminal is tracked. In current cellular networks, there exists also some challenges for extracting the exact cellular handover information from core network of 3G systems. For example, it is not possible to observe cellular handovers over Gn interface but only location area updates. Therefore, we have focused on extracting flows that have gone through location updates through users performing routing areas handovers. Routing area may include one to several base stations connected to same RNC.

Final outcome of this work provides a combined and detailed listing of the control packets and user plane flows indicating Routing Area changes of mobile terminals via the PDP Context Update messages.

2.1.3.2 System Description

A general view of the flow information extraction mechanism is provided in Figure 2.3. The Gn interface in the core network between Serving GPRS Support Node (SGSN) and Gateway GPRS Support Node (GGSN) is mirrored and collected in Flow Extraction Manager (FEM). FEM applies initial processing and transfers the data to be analyzed and filtered in the Hadoop Cluster. The extracted flow information is sent back to FEM from the Hadoop Cluster. The details of the

extraction process are given in Section 2.1.3.3.

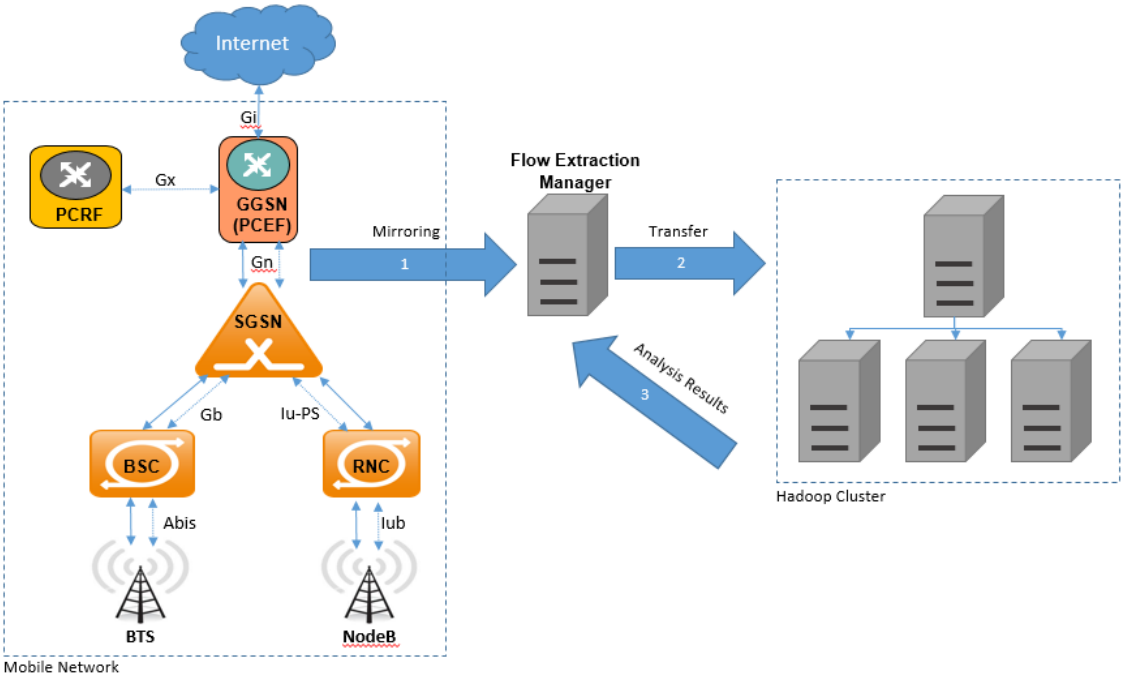

Figure 2.3: Flow Information Extraction Architecture.

GTP-U and GTP-C on Gn Interface

Gn Interface is the interface between SGSN and GGSN. Network packets sent from a UE to the Packet Data Network (PDN), e.g. internet, pass through SGSN and GGSN where GPRS Tunnel Protocol (GTP) constitutes the main protocol in network packets flowing through Gn interface.

The user plane packets on Gn interface flow on the GPRS Tunnel Protocol User (GTP-U) [4], which is in effect a relatively simple IP based tunneling protocol permitting many tunnels between each set of end points. When used in the Universal Mobile Telecommunication System (UMTS), each subscriber will have one or more tunnel, one for each PDP context that they have active, as well as possibly having separate tunnels for specific connections with different quality of service requirements. The separate tunnels are identified by a Tunnel Endpoint Identifier (TEID) in the GTP-U messages, which should be a dynamically allocated random number.

The control plane packets on Gn interface flow on the control section of the GTP, namely GTP-C [4]. When a subscriber requests a PDP context, the SGSN will send a create PDP context request GPRS Tunnel Protocol Control (GTP-C) message to the GGSN giving details of the subscriber's request. The GGSN will then respond with a create PDP context response GTP-C message which will either give details of the PDP context actually activated or will indicate a failure and give a reason for that failure.

Mirroring

AVEA network consists of several districts with more than 10 regional core areas throughout Turkey. The average total traffic over all regional areas consists of approximately over 15 billion packets in uplink direction and over 20 billion packets in the downlink direction daily. This corresponds to approximately over 80 TB of total data flowing in uplink and downlink daily in a mobile operator core network. The data usage behaviour results in exponential increase in data traffic of a mobile

operator. For example, in 2012, the approximate total data traffic was over 7TB in both uplink and downlink daily traffic.

We tested our methods on real-world Gn interface Internet traffic data. The streaming traces are obtained from one of the AVEA's core network region collected by a server on a high speed link of 200 Mbit/sec at peak hours.

Hadoop platform

Among the available big data platforms, Hadoop stands out as the most notable one as it is an open source solution [5]. It is made up of a storage module, namely Hadoop Distributed File System (HDFS) and a computation module, namely MapReduce. Whereas HDFS can have centralized or distributed implementations, MapReduce inherently has a distributed structure that enables it to execute jobs in parallel on multiple nodes.

The extraction and analysis of flows performing routing area updates was tested in AVEA's network. Towards this purpose, a prototype was implemented along the lines outlined in the previous section. A data processing platform was implemented through using Cloudera Distribution Including Apache Hadoop (CDH4) [6] version on four nodes including one cluster name node, with computations powers corresponding to each node with INTEL Xeon CPU E5-2670 running @2.6 GHZ, 32 CORE CPU, 132 GB RAM, 20 TB hard disk.

For the analysis, after mirroring stage of Gn interface, network traffic is transferred into a server (called Crowd Server). The size of data flowing through this interface is observed to be around 200 Mbit/sec at peak hours during a day.

2.1.3.3 Flow Information Extraction

The process extracts and matches the tunnel identification (TEID_DATA) and TEID in the GTP-C (Update PDP Context) and GTP-U packets respectively, in order to obtain the Location Area Code (LAC) and Service Area Code (SAC) of the user plane packets that does not exits inherently.

The Service Area identifies an area of one or more cells of the same Location Area (LA). A Service Area is identified with a SAC and represents Cell-ID in AVEAs 3G networks. A location area is defined to be a set of base stations that are grouped together to optimise signalling overhead. In a typical mobile operator, to each location area, a unique number called a LAC is assigned. Typically, tens or even hundreds of base stations are present in a given location area. A TEID uniquely identifies a tunnel endpoint on the receiving end of the GTP tunnel [4]. A local TEID value is assigned at the receiving end of a GTP tunnel in order to send messages through the tunnel.

The GTP-C packets contain Cellular identification (LAC and SAC) and TEID_DATA. The GTP-U packets contain TEID plus several information regarding the data such as size of each packet. The flow information can be extracted from data packets by taking into account 5 tuples namely, layer 4 protocol information (e.g. Transmission Control Protocol (TCP), UDP, Internet Control Message Protocol (ICMP), etc), source and destination IP address and source and destination port numbers. Taking advantage of the unique TEID, these information are joined to obtain a merged table showing the flow and cellular identification of the user packets. Note that in a given PDP session with a specific TEID, there can be multiple flows. However, each TEID belongs to specific user. Each user can also have multiple TEIDs and create multiple flows. Note also that one flow can have multiple TEIDs and and belong to only one user.

Flow information extraction mechanism is composed of several steps as provided in Section 2.1.3.3. The mechanism is applicable to all mobile operators using the Third Generation Partnership Project (3GPP) standard Gn interface.

Step-by-step Extraction

Step 1: Capture packets

Use tshark to capture and save packets into files.

Input: NONE

Command:

```
tshark -a <DURATION> -b <FILE-SIZE> i <INTERFACE> -w <FILE-LOCATION>
```

Output: Ordered Packet Capture (PCAP) files for time duration of DURATION, each having packet information in binary format with size FILE-SIZE stored in folder FILE-LOCATION.

Step 2: Merge captured packets

Tools that are used can only handle one PCAP file. Therefore, all PCAP files have to be merged into one big PCAP file.

Input: Output of Step 1; time ordered PCAP files

Command:

```
mergecap merge.pcap <FILE_REGEX>
```

Output: All packets in a single PCAP file (merge.pcap; for the example)

Step 3: Extract required fields

The unnecessary header and data information contained in the packets are eliminated and required header fields are converted to text format with a script code.

Input: Output of Step 2, Merged PCAP file (merged.pcap for this example)

Command:

```
script_sharing.sh
```

Output: PCAPs in text format with only extracted fields; <filename>.out (merge.pcap.out for this example)

Step 4: Extract the control packets

Control packets, which keep header information of LAC and SAC, are extracted.

Input: Output of Step 3, a file containing merged PCAPs with extracted fields (merged.pcap.out for this example)

Command:

```
python find_control_packets.py merge.pcap.out
```

Output: a Comma Separated Values (CSV) file with all control packets (example: control_packets.csv)

Step 5: Extract the user packets and generate the flows

User packets associated with same flow are extracted using a separate code written in C (fsd_server).

Input: Output of Step 2, merged PCAP file (merged.pcap for this example)

Command:

```
fsd_server -i <filename> -o <outputdirectory> -pt -n
```

Output: CSV files containing user packets with their associated flows

Step 6: Merge user packet files

Packets are merged into a single file for processing in the Hadoop cluster.

Input: Output of Step 5, CSV files containing user packets

Command:

```
for i in $folder
do
cat $i >> user_packets.csv
done
```

Output: User packets with assigned flows in one big file, (user_packets.csv)

Step 7: Insert user packets into hive table

Merged user packet files are placed in a hive table.

Input: Output of Step 6, user_packets.csv

Command:

```
hive
create table crw_user_packets(
flow_id STRING,
source_ip STRING,
source_port STRING,
destination_ip STRING,
destination_port STRING,
protocol STRING,
packet_size STRING,
arrival_time STRING,
teid STRING
) row format delimited
fields terminated by ','
stored as textfile;

load data local inpath "<Path_of_input>" overwrite into table svnc_data_usage
```

Output: User packets table in hive

Step 8: Insert control packets into hive table

Merged control packet files are placed in a hive table.

Input: Output of Step 4, control_packets.csv

Command:

```
hive
create table crw_control_packets(
gsma_cell_ci STRING,
gtp_ext_sac STRING,
gtp_lac STRING,
gsma_cell_lac STRING,
ip_protocol STRING,
frame_time STRING,
frame_len STRING,
```

```
gtp_teid STRING,
gtp_teid_data STRING,
gtp_message STRING
) row format delimited
fields terminated by ','
stored as textfile;

load data local inpath "<Path_of_input>" overwrite into table svnc_data_usage
```

Output: Control packets table in hive

Step 9: Merge hive tables

Merge two hive tables using UNION command to order packets as their original order in Step 1.

Input: Tables in Step 7 and 8

Command:

```
hive
Create table crw_all_packets as
select * from
(
SELECT
"U" as type,
"" as sac,
  "" as lac,
  "" as cell_lac,
  p.flow_id as flow_id,
  p.arrival_time as time
  p.teid
FROM
  crw_packet_flow_information p

Union all

SELECT
"C" as type,
  c.gtp_ext_sac as sac,
  c.gtp_lac as lac,
  c.gsma_cell_lac as cell_lac,
  "" as flow_id,
  c.frame_time as time
FROM
  crw_control_packet_information c
  )
  asd
   order by time
```

Output: Merged table

Step 10: Export the merged table

Export the table as a CSV file.

Input: Merged table in Step 10

Command:

```
hdfs dfs -getmerge <Hadoop-Path-of-Table> <Output-file-path>
```

Output: Exported table as CSV; all_data.csv

Step 11: Generate LAC and SAC

Missing LAC SAC fields of the user packets in the merged table of Step 10 are filled accordingly.

Input: Output of step 10; all_data.csv

Command:

```
python lacsacs.py <Input-File>
```

Output: A file containing all packets <Input-File>.out

Step 12: Hadoop analysis (optional)

The final table with LAC and SAC fields is further analyzed in Hadoop if required.

Input: Output of 11, all_data.csv.out

Command:

```
load data local inpath "<Path_of_input>" overwrite into table crw_all_packets
```

Output: Final table after Hadoop analysis.

Sample Output

A sample output of the information extraction process is provided in Table 2.1. Using this table mobility pattern analysis can be carried out. The most important fields to investigate in the constructed table are Flow ID and Cell ID. A change in Cell ID for the same Flow ID indicates a routing area change. Note that red coloured numbers represent the cell_id information obtained from previous control packets after Step 11. The fields in Table 2.1 are explained below.

type: (C) for control, (U) for user plane packets

flow_id: Flow identity for user packets, empty for control packets

protocol: Type of the protocol of user packet

time: Time stamp for packet arrival time (in msec)

size: size of the packet (in bytes)

cell_id: cellular identification number, unique number obtained from LAC and SAC

2.1.3.4 Conclusions

In this work, handover related information observed in the core network nodes on the Gn interface is extracted for further analysis. The extraction steps, which can be applied to all mobile networks using 3GPP standard, are provided in detail. The extracted information reveals the missing LAC and SAC information in user plane packets that will be used for mobility requirement analysis.

Table 2.1: Ordered Control (C) and Data (U) plane packet information

type	flow_id	protocol	time	size	cell_id
C			1408087486.2140		0x2bc7
C			1408087509.5465		0x2bc5
U	6527581	TCP	1408087605.3124	120	0x2bc5
U	6527581	TCP	1408087606.4562	113	0x2bc5
C			1408087611.1240		0x2bc7
C			1408087646.7231		0x2bc7
U	6527581	TCP	1408087648.6929	1498	0x2bc7
U	6527581	TCP	1408087648.6929	110	0x2bc7
U	6527581	TCP	1408087649.1796	1440	0x2bc7
U	6527581	TCP	1408087650.1539	110	0x2bc7
U	6527581	TCP	1408087652.0997	143	0x2bc7
U	6527581	TCP	1408087655.9953	113	0x2bc7
C			1408087662.1045		0x0088
C			1408087679.8554		0x42d4
C			1408087780.6871		0x42d4
...					...

2.2 Mobility Management

2.2.1 How does my smartphone manage network connections?

We are immerse in a world that becomes more and more mobile every day, with ubiquitous connectivity and increasing demand for mobile services. Current mobile terminals support several access technologies, enabling users to gain connectivity in a plethora of scenarios and favoring their mobility. However, the management of network connectivity using multiple interfaces is still starting to be deployed. The lack of smart connectivity management in multi-interface devices forces applications to be explicitly aware of the variations in the connectivity state (changes in active interface, simultaneous access from several interfaces, etc.). In this work, we analyze the present state of the connection management and handover capabilities in the three major mobile Operating Systems (OSs): Android, iOS and Windows. To this aim, we conduct a thorough experimental study on the connectivity management of each operating system, including several versions of the OS on different mobile terminals, analyzing the differences and similarities between them. Moreover, in order to assess how mobility is handled and how this can affect the final user, we perform an exhaustive experimental analysis on application behavior in intra- and inter-technology handover. Based on this experience, we identify open issues in the smartphone connectivity management policies and implementations, highlighting easy to deploy yet unimplemented improvements, as well as potential integration of mobility protocols.

The main conclusions extracted from the complete work, which can be found in Section 4.1 are summarized in the following:

Simultaneous usage of network interfaces: Out of the three OS families under study, Windows Phone 8 and Android Lollipop allow simultaneous usage of cellular and WLAN interfaces. The Windows Phone 8 device keeps active connections over the cellular and the WLAN interfaces simultaneously. Android, only in its latest version (Lollipop) modified its policy to allow simultaneous connection through both interfaces. The cellular connection remains active for 30 seconds after ongoing sessions finish. iOS devices finish the open connections on the cellular interface when

a connection to a WLAN is established.

Network selection: None of the systems under study perform a network selection algorithm to decide on the best network available to connect to. They rely on the network used in the last connection, if it is available. In addition, none of the systems uses information on link quality or performance to change point of attachment if needed. iOS8 WiFi client is particularly sticky, even not responding to user choices and remaining attached to the current Access Point (AP) if the signal is not lost.

Connection establishment: Android and Windows Phone 8 renew their IP address by reissuing a Dynamic Host Configuration Protocol (DHCP) discovery every time they connect to a different AP, but iOS does not if the change of AP takes place within the same Extended Service Set (ESS). In the initial attachment to a WLAN, Android outperforms the other systems in the link layer attachment, but it is considerably slower in the IP configuration. The Windows Phone 8 (WP8) terminal has proven to be the fastest one, on average, for initial attachment to an already visited WLAN. It calls our attention the remarkable impairment of performance in the connection establishment to WiFi networks in Android Lollipop running on Google Nexus 5. The delay in the connection establishment for iOS8 is also slightly higher than in previous versions, but the difference is not as notorious as for Lollipop (on Google Nexus 5). Although the IP configuration has been made faster, the connection to the AP has been considerably damaged. It is also new with respect to previous versions, the use of Gratuitous Address Resolution Protocol (ARP) in iOS8 and the Clear to Send (CTS)-to-self frames sent before authentication frames by iOS8 (iPhone 6) and Lollipop (Google Nexus 5).

IPv6 configuration: The three OS families implement privacy extensions for Stateless autoconfiguration (SLAAC) [7], configuring an IPv6 address that does not match their respective EUI-64 identifiers. Actually, this is not the only Internet Protocol v6 (IPv6) address configured in the terminal interface, so applications should handle this and take into account that this kind of address will change over time, which may affect ongoing sessions. The first Domain Name System (DNS) query sent by the Android phone always requests an IPv6 address, so IPv6 takes precedence over Internet Protocol v4 (IPv4). However, the IPv6 configuration takes a significantly longer period to be completed. On the contrary, WP8 issues first the IPv4 query but the delays for IPv4 and IPv6 configurations are comparable. The cellular networks available for our experiments do not offer IPv6 support for the moment. Although standardization bodies have provided guidelines for the migration to IPv6, to our knowledge, at the time of writing only some LTE networks in North America and Europe support IPv6 access.

Handover: Not having any application running increases delay in case of a handover. Android (except for Lollipop) presents a more regular behavior, having a handover latency of 0.9 s for most of the cases evaluated. WP8 and iOS devices present a more variable performance. Particularly, when a handover is initiated by a deauthentication from the AP (between different Extended Service Set IDs (ESSIDs)), the interruption in connectivity through the WLAN can take approximately 5 s, 10 s or 12 s on average for Android, WP8 and iOS systems respectively. However, when the handover happens within the same ESS it is completed in 0.24 s or 0.4 s by the WP8 phone, a result which outperforms the other two systems. When the handover is initiated by the terminal, the fastest handover (90 ms) is performed by the Google Nexus 5 Android Lollipop device if the user is manually indicating the target network, closely followed by iOS devices. Note that changing to a different channel, even if the user does it manually, increases delay considerably in iOS8, due to its sticky client implementation. However, Android and WP8 offer lower delays than iOS if the terminal needs to scan for available networks to decide on the new AP to connect to ("forget" case in our experimental results). Nevertheless, the new Android version offers a significantly higher delay (around 4 s) and the new iOS version (iOS8) overcomes the issues with the manual connection and lowers the delay to just 170 ms. Only the iOS devices and Android Lollipop perform a re-

association when the handover takes place within the same ESS. This diminishes the handover delay for the Lollipop devices, but not for the iOS terminals. Although all the systems fall back to the cellular connection when they lose WLAN connectivity, this change does not increase the delay in the WiFi-WiFi handover. The change to the cellular network shades the interruption in the WiFi interface, allowing for time to perform the scanning and the association to the new AP. The remarkably bad performance of Android Lollipop and iOS8 deserve a special mention, especially in the case of changing to a different channel manually for iOS8 and as a general consideration for Android Lollipop.

Multicast and network traffic: Unsurprisingly, Hypertext Transfer Protocol (HTTP) is the dominant traffic as it is also the recommended way to access remote content in the development documentation. It calls our attention the intensive use of IEEE 802.11 Null function frames. The use of these frames is not specified by the standard, but WiFi clients, especially in smartphones, send a significant amount of these frames regularly. The most extended use of Null Function frames is power management, so the station informs the AP when it goes to sleep or awakes. However, these frames are sent very regularly, and, specifically, when connection to the current AP is lost. In our handover experiments, we have detected that Android and iOS devices try to reach the AP, whose signal is lost, by sending numerous Null Function frames and Probe Requests, delaying the connection to a new point of attachment. It is also remarkable the number of DNS requests required every time that any connection is open. Regarding multicast support, Inter Group Management Protocol (IGMP) is not supported in some Android devices, including Google Nexus 4 and Google Nexus 5, which we used in our experiments.

2.2.2 SDN Distributed Mobility Management

The current wireless network architecture needs to be re-thought to support mobility in very dense and heterogeneous network deployments. We propose and experimentally evaluate a novel SDN-based architecture which makes use of DMM concepts to deploy fast, flexible, reliable, and scalable mobility management mechanisms at both local and regional scopes. Our solution is compatible with existing protocols deployed in wireless access and backhaul networks, and is therefore technology transparent to the user. Although our architecture is generic and can be used with heterogeneous wireless networks, we prove the validity of our approach in a lab prototype, by implementing our distributed mobility management mechanism on a set of OpenFlow-controlled IEEE 802.11 networks.

In this work we present and experimentally validate the distributed mobility management solution designed in the frame of the CROWD project. In particular, we propose a network-based mobility solution, which benefits of flexibility and scalability properties of DMM and SDN. Our solution is transparent to the mobile terminal and provides a two-fold mobility support: first, our solution supports fast layer-2 mobility within a *district*; second, our proposal includes a DMM-based SDN mobility solution designed for the inter-district mobility (layer-3 mobility). We implement our mobility management mechanism in IEEE 802.11 networks, and demonstrate the very promising features of our solution—in terms of mobility support, data path re-configuration, and transparency to the mobile devices—using the celebrated OpenFlow protocol [8] to control access points and network switches. However, our approach is designed so that it can encompass other technologies (e.g., LTE) and heterogeneous access networks.

To characterize the performance of our prototype we run 30 repetitions of every experiment. Figure 2.4 shows the CDF of the delay due to the intra- and inter-district handovers, measured as the interruption in data traffic, which also includes the link layer association. The intra-district case shows a lower latency, because the Mobile Node (MN) is already known in the district and therefore a simple change in the point of attachment to the network in needed in that case. The inter-district handover implies changes at layer-3 level, including the configuration of the IP-in-IP

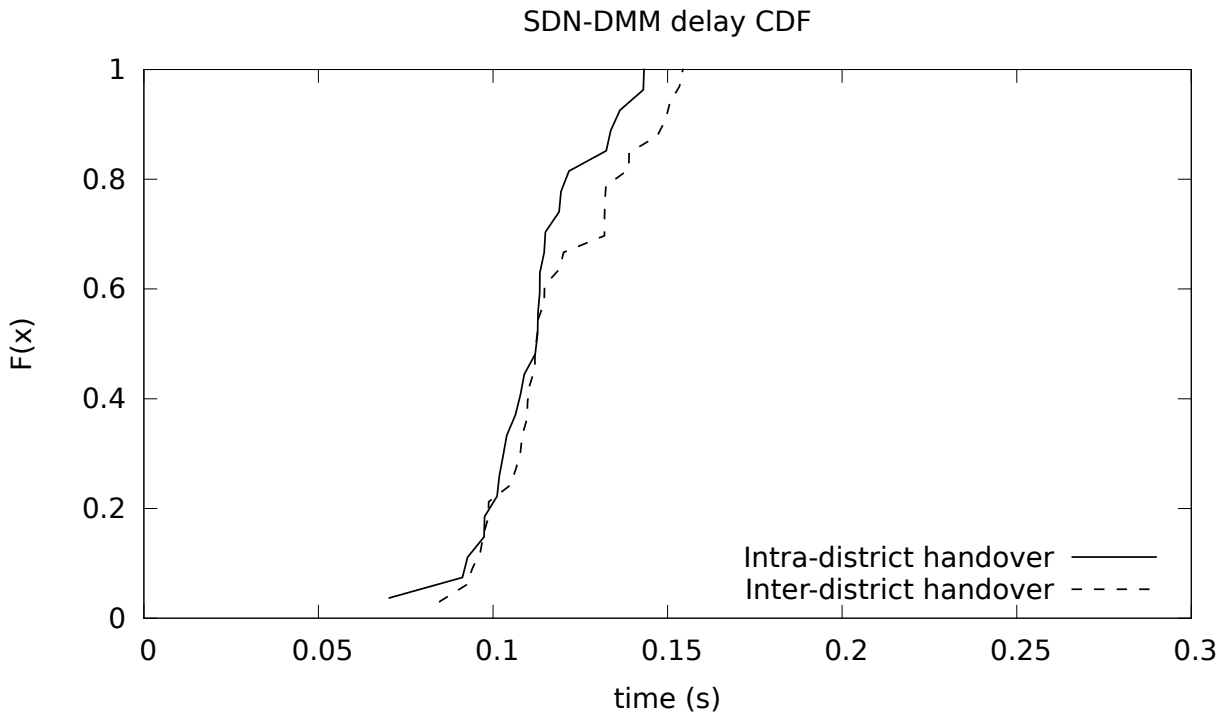

Figure 2.4: CDF of the different handover delays

tunnel between the two DMM-Gateways (GWs) involved. However, the delay is very similar to the intra-district handover, because the controller runs the configuration of the gateways and the establishment of the OpenFlow datapath in parallel.

Following with the evaluation of the performance of our SDN solution, we measure the throughput we can achieve for TCP traffic, using the *iperf* traffic generator. We compare the case the MN is attached to one of the districts and the case that it has performed a handover and its traffic is tunneled between the current and former DMM-GWs. The average throughput is 8.6 *Mb/s* and 8.1 *Mb/s*, respectively, confirming that the impact of this tunneling is very low (the IP-in-IP tunnel just adds 40 bytes of headers). It is noticeable that the maximum throughput available in the SDN framework is quite limited for the moment. This is a well-known open issue and very heavily influenced by the use of the Linksys WRT54GL router as OpenFlow-switches.

It is worth to mention that the signaling due to OpenFlow does not impact the data traffic performance, as we use out-of-band signaling within the SDN elements, and the data transmission from the MN is only delayed at the first packet, when the matching rules are firstly installed in the OpenFlow switches. To evaluate the impact of the SDN operations, we have measured the delay introduced by the operations carried out by the controlled required for our SDN-based DMM solution. From our experience, the choice of the device running the controller has a significant impact on the performance of the prototype, especially in the case of handover.

2.3 Access Selection

2.3.1 Energy efficient access selection

In the last few years, the wireless access network architecture is changing rapidly. In order to cope with the increasing bandwidth demand, macro-cells, femto-cells and pico-cells start to be smoothly integrated in the deployment of cellular networks. Wireless Fidelity (WiFi) is not only

used in dedicated hot-spots or to connect isolated regions to the Internet, but it is effectively used to offload the cellular network. Alongside the classic wireless access network, a wide set of other possibilities is also flourishing. For example, opportunistic relaying of the traffic by mobile devices through Device-to-Device (D2D) communications [9] has also been investigated. We call such access network, an "Heterogeneous Access Network".

On the other hand, recent studies show that the main concern of smartphone owners is the battery lifetime of their devices [10]. Smartphones are assuming the role of convergent devices and they are able to accomplish to an always growing number of functionalities. Therefore, due to their massive use in everyday life, smartphones are becoming increasingly power-hungry, and their batteries lifetime are decreasing accordingly.

In this section we try to exploit the heterogeneous nature of the access network to improve the usage efficiency of device batteries. We leverage the fact that modern devices are multi-homed, i.e., devices are equipped with several network interfaces. Basically, smartphones can access the Internet through any Point of Attachments (PoAs) deployed in the access network. Our proposal consists in a new network-centric access selection mechanism that aims in maximizing the energy efficiency achieved at the terminal side. Specifically, whenever a new device tries to access the Internet, the network chooses the most efficient PoA through which the device will access the internet. The choice is done such that following figure of energy efficiency is maximized:

$$E = \sum_{i \in M} \frac{T_i^U + T_i^D}{P_i^U + P_i^D}, \tag{2.2}$$

where M is the set of devices present in the system, T_i^U and T_i^D are the throughput (in bit/s) achieved by the particular device i in uplink and downlink, while P_i^U and P_i^D are the powers consumed by i while transmitting/receiving. As a major point of the proposed access selection mechanism, uplink and downlink are not considered separately, but they are both part of the maximization. At the best of our knowledge, the proposed analysis is the first one including both directions of transmission. Even if this design choice implies a considerable increase in the complexity, we believe that it represents a much more realistic scenario, since it is rare the case in which a device remains completely silent in one of the two transmission directions. Furthermore, in Appendix 4.5, we show how downlink and uplink performances are strictly correlated when WiFi is involved.

2.3.1.1 Problem Formulation

In the scenario under evaluation, each of the terminals could access the internet selecting among PoAs belonging to different transmission technology, if in coverage. The terminals can be associated to the cellular access network, either to macrocells either to short-ranged femtocells, they could choose to access the internet through IEEE 802.11 APs, or they could choose to relay their traffic to a terminal that is already connected to the cellular infrastructure, by means of a WiFi-Direct connections. Our objective can be written as follows.

$$\max \quad \sum_{S\in V}\sum_{j\in N}\sum_{i\in S}\left(\frac{T_i^U(S,j)+T_i^D(S,j)}{P_i^U(S,j)+P_i^D(S,j)}\right)y(S,j);$$

subject to

$$\sum_{j\in N}\sum_{S\ni i} y(S,j)=1,\ \forall i\in M;$$

$$\sum_{S\in V} y(S,j)\leq 1,\ \forall j\in N;$$

$$RSS_{i,j}\leq y(S,j)\delta,\ \forall i\in S,\ \forall i\in V;$$

$$\sum y(S,j)\in\{0,1\};$$

where V is the set of possible groups of devices that can be assigned to a PoA, N is the set of PoAs in the network (eNBs+APs+WiFi-Group Owners (GOs)) (please note that if j is an AP, no relay scheme is allowed) and $y(S,j)$ is a binary variable that is 1 when the subset of devices S is assigned to the PoA j, 0 otherwise. The constraints introduced ensure that each device $i\in M$ is assigned to exactly one PoA and that each subset of devices S is assigned to at most one PoA. $RSS_{i,j}$ is the signal strength received from UE i and belonging to PoA j, while δ represents a threshold that ensures that UEs attach just to PoAs with a given quality of service. Whenever a particular subset S of devices is assigned to a given PoA j, i.e., $y(S,j)=1$, we compute the throughput and the power consumption taking into account the demands of the users involved and also the saturation of the resources of each PoA. In Appendix 4.5 we present in detail the computation carried out in each of the possible cases.

The optimization problem proposed is the off-line version of our problem, i.e., the problem we should solve if all the devices ask contemporary to access the network. Due to the fact that the access selection mechanism has to take separate decisions every time a new user enters the network, we actually tackle the optimization problem in on-line fashion. As shown in [11], the optimization problem we propose can be written as a Combinatorial Auction Problem (CAP). In CAP, bidders express their interests in acquiring bundles of objects through offers. Given the offers, the auctioneer sells the objects in bundles to the bidders such that the bidders receive at most a bundle and such that the overall benefit is optimized.

In our case, each bundle of objects represents the set of devices we decide to attach to a given PoA. Bidders are instead the PoAs themselves and the offers a PoA does for a given bundle of user represents the overall energy efficiency the bundle of users achieves under the particular PoA under analysis. In our particular CAP, the auctioneer has also to respect an additional constraint, i.e., all the objects have to be sold. In the worst case, a CAP has to explore all the bids, that grows exponentially with the number of devices. Complexity reduces if it exists a computationally efficient way to express the value of the bundles. As we show in Appendix 4.5, this is not our case, therefore the off-line and, consequently, the on-line instance of our problem is actually intractable.

2.3.1.2 Energy Efficient Access Selection/Vertical Handover Mechanism

In order to find a solution to our optimization problem, we propose a new access selection mechanism. We inspire our policy on mechanisms that are typically used to solve CAP problems. We named such policy Marginal Benefit policy. Such policy will be applied anytime a new UE appears in the system, as well as when a UE requests to detach from its current PoA due to the bad channel quality (possibly requiring also a Vertical HandOver (VHO)).

The algorithms works as follows:

- For each of the PoA in the scenario, we first compute the energy efficiency (as the ratio of throughput achieved and power consumed) of all the users attached;
- For each PoA where the UE under analysis can attach, we compute the energy efficiency of all users attached, considering also the presence of the UE entering the network;
- If the UE is in proximity of a given device that can act as GO, then we also evaluate the possibility that such GO relays the traffic of the UE to the cellular infrastructure;
- Among all PoAs, we attach the UE to the one which energy efficiency is improved the best with the insertion of the UE. Please note that in case of eNBs, then the improvement in energy efficiency is computed considering also the possibility to attach the UE to the available WiFi groups in the system.

Such policy is able to achieve a great gain in terms of energy efficiency when compared with classical access selection mechanisms. Indeed, we evaluated the total throughput (downlink plus uplink) as well as the energy efficiency achieved by devices in several scenarios and we compared such results with the one achieved using a " WiFi-First" policy. That is, UEs tend to prefer WiFi AP over eNB, when an "Heterogeneous Access Network" is in place. Figure 2.5 represents the achieved results in several simulation scenarios. We performed simulations varying the number of UEs from 1 to 250, their positioning and their channel quality. Further details on the simulations carried out can be found in Appendix 4.5. On average, our access selection mechanism improves state-of-the-art solutions consistently, also when the number of nodes in the system is small. On average we registered an improvement of the 269.46% on the total throughput achieved by an user, when compared to the throughput achieved by " WiFi-First" policy, while we noted an improvement of the energy efficiency of the UEs of at least the 180% when the scenario was considerably crowded (more than 200 UEs).

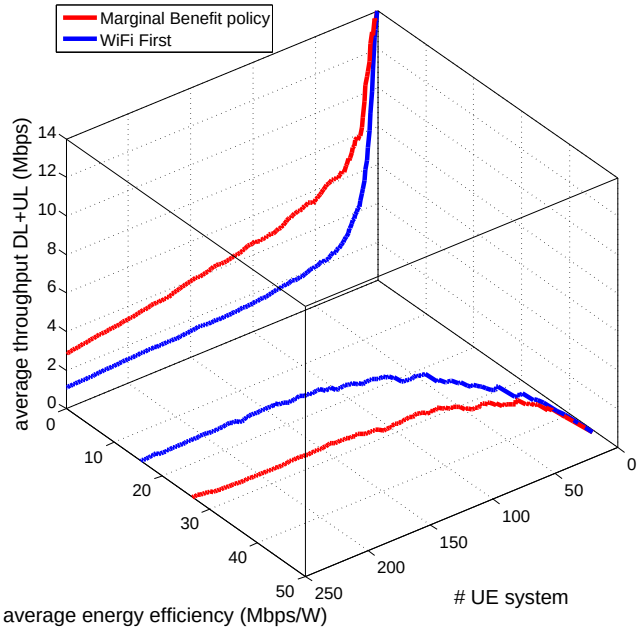

Figure 2.5: Average Energy Efficiency and Throughput - Simulations.

A " WiFi-First" policy does not attach incoming UEs considering the load already present at the PoAs, but just evaluating simple channel quality measurements. Therefore, such results were easy to envision. In WiFi, all devices transmitting on the same bandwidth end up contending for

the same resources, pushing rapidly to saturation the APs in the system. As a result, throughput and energy efficiency quickly decrease. Our access selection mechanism solves this issue, implicitly attaching the incoming UEs to the PoA that improves the best the utility of the system.

2.3.2 Information aggregation and distribution

One of the 5G requirements is to have a radio agnostic approach and allowing simultaneous radio accesses. The scarcity of the radio resources implies to optimize the choice of the used radio accesses and have a deep knowledge of the available radio resources. The information about the radio accesses may be provided by a discovery process done by the UE itself or sent by the operator. In this part we describe solutions that cover the two ways to gather the information. The first solution is based on the IEEE 802.11u protocol that allows a UE to get various information from the beacon frames, without being attached to the WiFi access point. The second solution uses the IEEE 802.21 (Media Independent Handover (MIH)) protocol to gather information about operator managed WiFi access point close enough to the UE to be candidate to an attachment.

Hotspot 2.0 information gathering

IEEE 802.11u standard has proposed some features that allow UE to be able to connect to a WiFi AP in an efficient way.

WiFi APs were usually discovered through the scan process during a handover, and then each potential AP are checked (the order being often the Most Recently Used Service Set IDs (SSIDs)), this requires an authentication which is a slow process. To avoid useless or unsuccessful authentication tries IEEE 802.11u has implemented several features that improve the selection of suitable WiFi APs for connection.

The different kind of information elements are:

- Network identification
  - Roaming Consortium: the list of all roaming partners that are allowed to use the WiFi AP.
  - Homogeneous ESSID: A common ESSID, identifying easily a group of WiFi APs.
- AP identification
- Authentication information: which authentication protocol is used ?
- AP capabilities
  - Access network type and charging information
  - Connectivity to internet
  - AP load: the current load of the WiFi AP. It allows detection of overloaded WiFi AP to avoid (even if the SINR is high).
  - Venue information: characterizes the aim of the use of the WiFi AP.

A partial view of this information is present in the beacon frames. To have a complete description, the UE should send a query using the ANQP protocol, and the WiFi AP will send back the ANQP response.

For example, the beacon frame can contain at most three roaming partners, if the number of roaming partners is greater than three, the ANQP protocol should be used to get all roaming partners and look if the current UE's operator is one of them.

The analysis of the beacon frames may be done at any time, not only during the handover process, this may be interesting to dynamically/periodically detect the best WiFi AP candidates before the decreasing SINR triggers a handover. For instance, this may be used to anticipate some load balancing between the different radio accesses.

MIH information gathering

The operator may also provide information about its own radio accesses and specifically about its operator managed WiFi APs. The IEEE 802.21 MIH protocol offers an information service Media Independent Information Service (MIIS) that allows to receive information about the radio accesses. Usually, the implementation uses a static information base which is rarely modified, but some measurements are changing a lot during time and these variations are not homogeneous. For example, if we consider the bandwidth, its value will likely increase until the busy hour and then will decrease, the busy hour being different depending of parameter such as the type of the area (a residential area and a business area have different bandwidth profiles during the day).

We have extended the information service by updating periodically the information base to provide more accurate values, and adding some fields to allow a better understanding of the behavior of the APs, making it easier to rank them at a given time.

The chosen use case is to demonstrate this with a UE that sends a message with its location, requesting information (here the bandwidth) of the candidates WiFi APs in the neighbourhood. The MIIS server sends the reply containing a list of WiFi APs close enough to the UE to be attached.

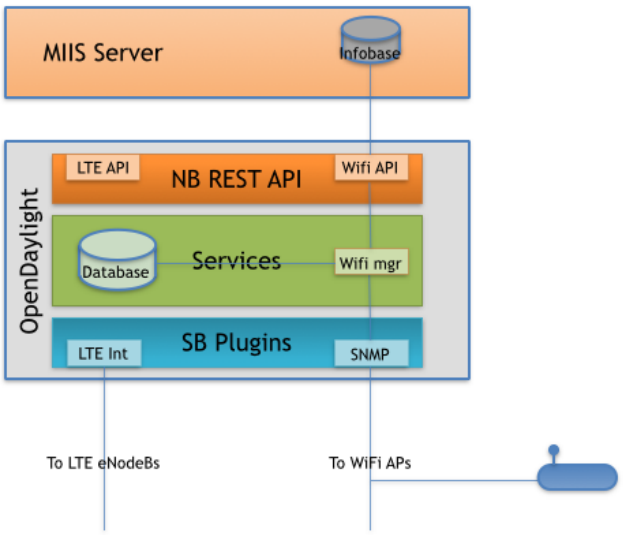

Figure 2.6: MIIS information gathering.

First we had to get the information from the WiFi APs, so we add a service to the CLC to gather periodically (tens of seconds) through a Simple Network Management Protocol (SNMP) request the WiFi AP state, then the service processes the values and stores them in a database. A Representational State Transfer (REST) API allows to query the service to get APs information through the northbound interface.

The CLC has been realized using the OpenDaylight platform (Helium version), the SNMP plugin has been used to implement the southbound interface. For the northbound interface, the JSON

REST API, described in [12] has been implemented. The data coming from the SNMP application is processed in order to allow query on larger timescale than the measurement period, some statistical computation is done to avoid non meaningful data.

In our testbed an application linked to the MIIS server is doing the request in order to update the MIIS information base, once updated the MIIS server is taking into account these modifications.

The UE is sending the usual MIH request to the MIIS server when it needs to get information about its neighbour WiFi APs.

The MIH event service has also been implemented (the MIH events are notifications linked to the handover process to inform a registered network element of the handover status and the value of the triggers used by the handover process). The notified network element may be the CLC and in this case, it may assume the role of a mobility manager for all radio accesses.

Conclusion and future works

The two approaches may be used simultaneously, as shown in Figure 2.7. This permits to have a complete view of all WiFi APs located in the neighbourhood, the operator managed Wifi APs described in the MIIS information base, and those that are compatible with 802.11u as the free access WiFi proposed in many places, allowing an opportunistic selection of available WiFi APs. Furthermore in both solutions the information request is not strictly correlated to the handover process, it may be anticipated, allowing a faster handover decision.

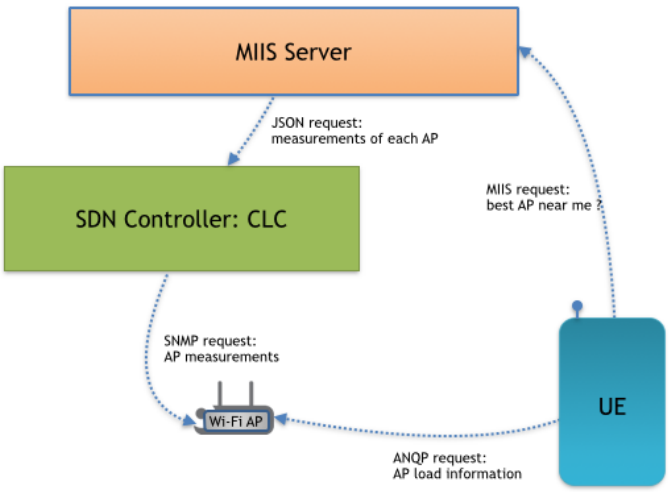

Figure 2.7: AP information example.

An extension that should be done is the use of the CROWD Regional Controller (CRC) as a global controller, to have information about all Wifi APs that are managed by the operator, the needed work is not complex, the use of the northbound interface of the CLC may be reused on the CRC to get the full functionality, but at this time it does not have any interest for the testbed demonstration.

Another improvement is to gather other kind of information from APs, and to extend it to other network elements such as the eNBs and the UEs. Having information from eNBs should permit to have an information base for all radio accesses as described in the 5G, allowing better performance for optimizations that require a knowledge of the complete network such as load balancing. The UEs information may be used to have a radio agnostic mobility management with opportunistic access to available WiFi APs.

3 Conclusion

With this deliverable we conclude the work of WP4 in the CROWD project. As explained in the Introduction section, this deliverable has focused on providing the latest results about the different research items tackled in this WP. Basically the different research items have touched all the areas identified in D4.1 [1] as critical for Connection Management in very dense scenarios. This activities include research on Connectivity Management or Target Network Selection. In this area we have focused on understanding current mobile terminals behaviours and how the selection of the access technology can impact the energy efficiency of the terminal. The second item studied corresponds to the mobility management protocol used in the network. The approach followed by CROWD is twofold. On the one hand the network uses an SDN approach to control at layer-2 the mobility of the user in a CROWD district. Then it uses a DMM-based solution to handle mobility across districts. In the case of handing over to a non-CROWD access, we also propose a host-based solution which integrates in the general architecture. This deliverable provides some measurements on the expected performance. Third, this WP also relies on information gathering to help the terminal taking the best network selection decision. This information gathering uses SDN monitoring tools, integrating them in well known IEEE 802.11u and IEEE 802.21 information services. Finally, some new work has also been reported in this period and is therefore described in this document. This new work corresponds to the analysis of a real cellular operator network based on traffic analysis, to assess the benefits of applying DMM in their network.

With this section we conclude this deliverable and the work in WP4.

4 Appendix

4.1 How does my smartphone manage network connections?

4.1.1 Introduction & Motivation

In the last years we have witnessed the market explosion of a new kind of handheld device, the smartphone. At this point in time it is clear that these devices are going to keep a steady market penetration and are going to be a major source of revenue and data traffic for network operators. According to [13], smartphones account for 88 % of the growth of global mobile devices and connections in 2014, with 439 million net additions in this year. Even though they represent just the 29 % of global handsets in use, they are responsible for 69 % of the total handset traffic, which clearly shows the major adoption of this technology. The use of smartphones has also modified the traditional patterns of mobile data consumption. The typical smartphone user is craving for data-based services, imposing a high burden on the operators, which see their investments on network deployments pushed to the limits due to greater bandwidth requirements. Due to the shift in user profile and data service demand experienced in the recent years, smartphones have become a powerful tool in most people's daily life. In addition, the enhanced capabilities and fast upgrades of hardware in handheld devices have considerably increased their usage. These facts pave the way to advanced research and development that relies on the use of smartphones for carrying out innovative tasks, mostly related to health care or behavioral studies and using the smartphone as a measurement instrument [14]. Moreover, there is a trend towards specialized, almost personalized services, which could benefit from an accurate knowledge on the capabilities supported by smartphones and how they manage their resources.

The current offer in the market is very wide, in terms of manufacturers, hardware resources and operating systems, but do the majority of the devices behave similarly? Is there any difference in the management of their own resources? Is the access to an application homogenized across the different systems? Is the network connectivity management standardized in all the various devices? Answering these questions could back the actual research on mobile devices and their applications to make daily tasks easier, but also more demanding use cases, applied to different fields. However, mainly because these details remain closed by manufacturers, most of this information is still missing or disregarded in research, which focuses on measuring performance or developing applications to serve specific purposes.

Smartphone users have one common and defining characteristic, they move. The need for supporting data services on the move has shaped the design of the cellular network, which must deploy access and core networks able to redirect users' traffic to their current location. In addition to the obvious problem of designing such networks, current smartphones support several wireless technologies, such as IEEE 802.11 [15], complementing the cellular connection. This heterogeneity brings new opportunities and challenges to the industry, since the additional technologies can be used to offload the traffic from the cellular network to local accesses, such as the broadband connection usually deployed at home.

The smartphone operating system provides a set of mobility-related functions from which an application can benefit in case it decides to handle mobility. These functionalities depend on the operating system (e.g., Android, iOS, WP8) and include connectivity events such as network up or down events, and commands exposed to the application layer to extract information on

connection availability. Usually the terminal connectivity is handled by a service widely known as the *Connection Manager*, in charge of deciding which is the best connection for the terminal in a specific moment, and the application has to deal with those decisions.

This work focuses on the analysis of the current state of the art on the mobility support at the Connection Manager in different terminals, providing a functional view of the differences between the major operating systems and the different improvements that can be done to optimize the mobile user experience. Our main contributions are:

- We analyze the default network connectivity management of the three currently most popular families of mobile OSes: Android, iOS and Windows Phone 8,[1] including iOS8 and Android Lollipop, in their latest versions supported to date. We test the same OS versions in different terminals, to avoid biased conclusions, derived from the performance of the terminal rather than from the OS behavior.
- We study how the smartphones running these different OSes perform inter- and intra-technology handover, considering the most widely used access networks: cellular and IEEE 802.11. For this study, we measure the handover latency in different scenarios and we evaluate the differences and similarities in the management and configuration of the networking parameters in each device.
- We evaluate how the handover performance affects the user experience by considering different applications, and whether they can survive to a change in connectivity: changes in IP address and changes in access technology.
- As a result of our experimentation, we identify the challenges and open issues that are present in current smartphones and discuss on potential improvements for the connectivity management that are feasible but not yet implemented and on the integration of connectivity management with current mobility protocols.

The rest of the document is structured as follows: In Section 4.1.2 we present the related work available in the literature and compare it to our work. We present the networking behavior of the systems under study, analyzing their main similarities and differences in Section 4.1.3. In Section 4.1.4 we present the experimental deployment, and in Section 4.1.5 we evaluate the initial attachment to the access network performed by the systems under study. In Section 4.1.6 we present a detailed analysis of the handover and how the connection manager of the three families of OSes handle the changes in connectivity, both inter- and intra-technology, making this analysis extensive to the management of this change in connectivity by some popular applications. Section 4.1.7 summarizes our main results and highlights our findings comparing all the studied systems. In Section 4.1.8 we identify open issues and improvements to tackle the shortcomings of the network management in mobile devices, in light of the data extracted from our experiments. We also discuss on the integration with mobility protocols. Finally, Section 4.1.11 concludes the article and present guidelines for our future work.

4.1.2 Related work

A significant part of the previous works in the literature analyze the energy consumption of smartphones, however, we focus on the network stack and how a multi-interfaced smartphone manages its network connections. We have compiled in Table 4.1 the previous works that address network performance in smartphones for a better comparison. Network connectivity has been addressed,

[1] Product names, logos, brands and other trademarks are registered and remain property of their respective holders (Google Inc., Apple Inc. and Microsoft Corporation). Their use throughout this document aims only at describing the results and the work performed and in no way it indicates any relationship with the holders of such trademarks.

Table 4.1: Comparison with related work on smartphone networking and Connection Manager

Ref.	System	Scope	Main features studied	Contributions	Main conclusions
[16]	Android Windows Mobile	Logging app	User interactions, application use, network traffic, energy drain	Evaluate the diversity range across users and time and their impact on network and energy	Patterns on app usage and user interaction time; apply diversity in usage to predict energy drain
[17]	Android	Logging app	Application popularity and usage patterns	Application usage models, app-launching optimization, personalized optimization framework for task manager	User experience can be improved by knowing usage patterns and context-aware resource management
[18]	iOS Android	Speedtest app	Cellular and WiFi network performance	Temporal and geographical analysis, aggregate performance	Similar throughput performance but, iOS higher latency
[19]	Android	Logging app	Cell, WiFi usage, phone usage	Analysis on application, phone, WiFi and cellular traffic usage	On average, WiFi traffic is 30% of the total data consumption
[20]	Android	Logging app	Traffic volume in 3G and WiFi networks	Aggregated and per-user analysis	Total traffic via WiFi much larger than via 3G, most by a small number of users
[21]	Smartphones Laptops	Wireless traffic captured by a gateway router in campus network	Network traffic, TCP impact, comparison to laptops, app layer parameters	Study network performance and traffic, mainly focusing on TCP-related parameters	Akamai and Google servers are heavily used. Different receive window advertised by iOS and Android, but similar performance
[22]	iOS Android Windows Mobile	Customized app: *3GTest*	TCP performance, RTT, DNS lookup time as metrics for app and network performance	Measurement tool and methodology for app performance comparison, 3G network performance for various operators	Smartphones are often the web browsing performance bottleneck, rather than the network
[23] [24]	Windows PC	Customized connectivity management	Vertical handover, session continuity and handover decision triggers	Connection manager to detect changes between WLAN and Wireless Wide Area Network (WWAN), virtual connectivity manager	Reduce number of handovers by keeping active connection instead of switching to WWAN when RSSI is below a threshold
[25]	Android	Application for connectivity management	WiFi offloading, simultaneous usage of cellular and WiFi interfaces	App for WiFi offloading and content aggregation (*Enhanced Android Connection Manager*)	Different use cases for WiFi offloading: content aggregation, SIM authentication or flow optimization and segregation
[26]	Android iOS	Server logs, sniffed wireless traffic and media player source code	Video streaming performance comparison.	Thorough comparison of iOS and Android clients behavior on streaming	Media players follow different content request and buffer management approaches. Redundant traffic downloaded by iOS

but mostly in terms of application usage and traffic patterns. For instance, [16] conducted a thorough study of application popularity and usage, characterizing the patterns followed by different demographic groups of users and the traffic generated. Their study confirms the high diversity in smartphone usage, leading to the conclusion that the tools in use may provide acceptable performance in average, but it could be considerably enhanced by some specific knowledge on applications performance and usage. Similarly, [17] also uses a logging application installed in the smartphone of a group of users and presents a personalized optimization for Android smartphones, based on application usage patterns per user, showing that the default task manager can be enhanced to improve user experience. We argue that a similar approach to these two can also be extended to network connectivity management.

The use of mobile devices equipped with several network interfaces motivates the performance study in [18], which characterizes consistency and compare the WiFi and the cellular accesses worldwide in terms of download/upload speeds and latency. The first promising application of this diversity in network connectivity is WiFi offloading. However, [19] shows that in spite of the dense WiFi deployment, cellular data consumption is still dominating and analyses the reasons behind that fact. A similar study is conducted in [20], but in this case, the authors conclude that the percentage of offloaded traffic is not negligible, being mostly exchanged at home APs. The differences between these two studies may lie on the geographical differences in their datasets. Yet again, it is proven that a unique solution for connectivity management cannot perform optimally, advocating for a kind of customizable solution per user or based on usage or mobility patterns. Chen et al. in [21] evaluate network performance of handheld devices by monitoring the traffic captured at a university campus. They also confirm the predominant presence of TCP and HTTP flows in the traces analyzed and focus their analysis on parameters such as slow start, the advertised receive window and characteristics related to the TCP flows. However, our analysis is centered on the network connectivity management and the performance in case of a handover. While the study in [21] is restricted to a WiFi connection, Huang et al. [22] evaluate network and application performance over 2G and 3G cellular accesses. They measure UDP and TCP throughput before examining the performance of two widely used applications, as web browsing and video streaming, by comparing them with different combinations of smartphone and network operator.

Devices equipped with multiple interfaces typically rely on the entity of the connection manager

mostly for the interface selection, and then provide networking information to the applications. In the literature we can find alternative designs for the connection manager. Zhang et al. [23] study mobility management between WLAN and WWAN and propose an architecture that relies on a connection manager and a virtual connectivity manager, which integrates end-to-end information to be used to optimize the handover. Their connection manager includes RSSI monitoring and network availability detection modules. This architecture is experimentally tested in [24], achieving promising results such as 2.1 seconds interruption from WLAN to cellular and seamless handover from cellular to WLAN, being able to select the best AP to associate with (in terms of available bandwidth).

To the best of our knowledge, only [25] tackles the shortcomings of the current Android Connection Manager by developing an application that enhances and extends its functionality. Their application plays with the possibilities offered by the usage of multiple interfaces and – by enabling simultaneous usage of cellular and WLAN interfaces – adds support for WiFi offloading, flow segregation and content aggregation. However, they do not provide an assessment on the Connection Manager itself neither analyze the overall behavior under different network scenarios and various conditions. Besides, [26] compares iOS and Android behavior in streaming, finding out that Android and iOS media players request data from the server differently and they also have different buffer management policies. As of the time of writing we are the first ones to provide a thorough analysis of the network management supported by an experimental evaluation of the inter- and intra-technology handover under a wide variety of configurations. In addition, we examine the network attachment executed by each device and we analyze the behavior of several applications in case of a handover, and how they can handle the change in the global connectivity of the terminal – most of the times unsuccessfully. The performance of an application does not only depend on the ability of the developer to handle network connectivity, but also the accessibility and flexibility offered by the operating system and the exposed Application Programming Interface (API). Above all, we compare the three most popular families of operating systems worldwide[2] and state the differences and similarities among the connection manager of an Android, iPhone and Windows Phone 8 devices, establishing the guidelines for further improvements in this unexplored feature.

4.1.3 Mobile terminal networking stack

Even though the developer community for the three families of OSes that we study is considerably large, there is no official documentation on the networking stack and the network management of the system. The effort of the community is focused on the application layer, thereby the main interest of a developer is centered at checking whether Internet connection is available, rather than making an efficient usage of the networking resources and optimizing performance. Still, we identify the most representative elements that define the network management in Android, iOS and Windows Phone 8, and we introduce them comparatively in this section.

4.1.3.1 Android

Android is an open source software stack released by Google, which makes publicly available the source code under Apache or GNU is not Unix (GNU) General Public Licenses. An Android system is based on Linux kernel 3.X (kernel 2.6 in versions up to Android 4.0). On top of the Linux kernel, there are the libraries commonly found also in Linux systems[3], and the Android runtime. The Android runtime consists on a Dalvik virtual machine –where the applications run– and on core libraries, specific for Android devices and used by the applications. In Android Lollipop, the new

[2] `http://www.businessinsider.com/android-is-utterly-dominant-in-europe-2013-7`

[3] Note that not all the Linux libraries are fully supported by Android and some changes have also been introduced to the architecture, such as *wake locks* and power management.

Android Runtime (ART) replaces Dalvik by default. An application framework interacts between the lower layers and the applications, which are on top of the system architecture.

1. **Default interface**: the Android API provides tools for an application to set its preferences in the use of the available interfaces and configure a default interface. It is important to isolate this behavior from the terminal's own networking preferences. In the case of the Android terminal in our experiments, the default interface is the cellular one, and simultaneous active connections over the cellular and the WLAN interfaces are not supported in versions previous to Lollipop. However, it is possible to modify the default Android behavior to use both interfaces at the same time [25]. In Android Lollipop, each interface has its own routing table and the cellular connection is kept for 30 seconds after switching to WiFi. The ongoing communications started over the cellular interface will remain there. In addition, the terminal will not connect to a WiFi AP that has no Internet connection.

2. **WLAN interface**: the hardware abstraction layer (HAL) contains the software modules that talk directly to the kernel wireless stack and drivers. The HAL is a user-space layer developed in C/C++. The application framework uses this C code to interact with the so-called *wpa_supplicant* module[4], which runs in the background and controls the wireless configuration.

3. **Network related events**: Android offers the possibility of tracking the logs of the system and an interested user is able to monitor the different events that happen in the system. This can be done through the Android Debug Bridge (ADB), a command line tool to communicate from a computer to emulated or physical Android devices. For example, we can monitor the active network connection (WIFI or cellular) and its coarse-grained status (*disconnected*, *connecting*, *connected*, *disconnecting*, *suspended*, *unknown*). In the case of the WiFi connection, it is also possible to monitor the state of the *wpa_supplicant* module, which also reports changes on its status (*associated*, *associating*, *authenticating*, *completed*, *disconnected*, *dormant*, *four_way_handshake*, *group_handshake*, *inactive*, *interface_disabled*, *invalid*, *scanning*, *uninitialized*). This information is also made available to application developers by the API framework.

4. **Application Programming Interface (API)**: the Android API framework is published in the form of API levels[5]. An API level is an integer that identifies the API revision exposed by an Android version, from the original API level 1 to the current API level 21. Applications intended to run on a specific Android version must support its correspondent API level, and they usually are backward compatible so that already existing applications can still be used. Network access is handled by the Android core libraries. As we have previously stated, the API provides applications with information of the network connection status and the HAL and kernel modules access to the network interfaces and are in charge of the (re)configuration. However, the network management still keeps a simplistic or conservative approach, as it is very limited in terms of optimization or using the two access network interfaces at the same time. Interestingly enough, the API provides constants and methods to check whether the signal strength has changed, to know when a scanning has been performed and information about surrounding APs is available, compute the difference in signal strength or change the state or configuration of the WiFi connection, so that applications could make a much wiser usage of the network connectivity[6].

[4] `http://w1.fi/hostap.git`

[5] Not necessarily a new Android version have to support a new API level, but commonly a new version upgrades the API too.

[6] The interested reader is referred to `http://developer.android.com/reference/packages.html` for a complete guide of the Android API framework.

5. **Flexibility and restrictions to the user**: because of being Linux-based, Android offers a high degree of flexibility in general configuration and networking support. In addition, the lax licensing has favored the distribution of customized versions of the firmware[7]. By default, Android users are not given root access, contrarily to Linux users, but it is a common practice among the Android community to gain root access, empowering the accessibility to all the terminal features. The root access and the Linux characteristics make the Android OS the most flexible and accessible of the systems under study. In this way, we have full access to the system logs record, we can monitor the status of the network interfaces and we can capture traffic with a network analyzer such as *tcpdump*[8].

4.1.3.2 iOS

The operating system running on Apple mobile devices is known as iOSn, being n the version number, currently the latest version available is iOS8, released in September 2014. iOS is based on the open source OS *Darwin*, however, iOS remains as closed-source. Apple's mobile operating system is built upon 4 abstraction layers, ordered from the top to the bottom: *i)* the Cocoa Touch layer, *ii)* the Media layer, *iii)* the Core Services layer and *iv)* the Core OS layer. The Core OS layer is the one that has access to the kernel, drivers and networking features as the interface to BSD sockets. However, developers are recommended to implement applications by using the framework in the highest level as possible, as the complexity of handling networking events or configuration is hidden by the OS.

1. **Default interface**: the cellular interface is assumed to provide always-on connectivity, but as soon as a wireless AP appears in range, the WiFi connection takes over the cellular connection. If the WLAN has already been visited in the past, and the user has allowed this configuration, the change happens automatically; otherwise, the user can be notified that there are WiFi networks available.

2. **WLAN interface**: the WLAN interface is selected as the primary interface, so it takes over the cellular connection.

3. **Network related events**: although the development platform does not offer access to the lower layers that talk directly to the hardware, the manufacturer gives some development guidelines for an application to be able to react to network connectivity changes. More specifically, examples of these events are latency, changes in network speed and variability of available network technologies. In the latter case, the documentation provides the *SC-NetworkReachability* API (available in the System Configuration framework in Core Services API) to diagnose the cause of the failure of a connection and determine availability of different connections.

4. **Application Programming Interface (API)**: iOS provides three different networking API layers: Foundation layer, Core Foundation layer (these two are specific to iOS) and POSIX layer (as in other UNIX systems). The three of them support common networking tasks, being recommended to use the highest level API that fulfills the developer's requirements. The networking API provided by iOS7 can be categorized into three main groups: BSD sockets, web access and Bonjour. The interested reader is referred to [27] for further details.

5. **Flexibility and restrictions to the user**: the iOS system is closed source and the user is not given much flexibility of configuration with respect to network preferences. Even

[7] One of the most popular ones is CyanogenMod, which reports more than 12 million installs (`http://www.cyanogenmod.org/`).
[8] `http://www.tcpdump.org/`

though the operating system has been hacked and it can be *jailbroken* it does not change the networking behavior or configuration, but increases flexibility application-wise.

4.1.3.3 Windows Phone 8

Windows Phone 8 (WP8) is developed by Microsoft, and, as the OS version for PC, they changed completely what was established in previous releases. Therefore, due to a change in the architecture, applications designed for WP8 cannot run in previous versions of the OS or devices running an older version cannot upgrade to WP8. As part of the features that Microsoft tried to improve in this new version, they include the network stack, focusing specially on speeding up the connection and reducing power consumption.

1. **Default interface**: one of the main changes in the networking stack in Windows 8 is the prioritization of network connections. Despite having a priority, multiple interfaces can be connected simultaneously. By default, the cellular interface is the one that is actively connected, but in the moment that an already visited WiFi network becomes available, the phone will try to connect to it. However, the connection through one interface does not kill a previous connection in another interface. It is only after a short period of time that the cellular connection will be terminated, unless it is being used by an application. In addition to that, existing connections at the moment of the new attachment, are kept alive and only the new connections will use the new interface. WP8 even offers the applications the possibility to select the network interface to use.

2. **WLAN interface**: the wireless network stack builds on top of the hardware device (and its firmware) with the driver and the Wi-Fi service of the OS. With this new generation of their OS, Microsoft targets to optimize power consumption and connection delay, which worsens user experience. In order to do so, they have tried to integrate as much operations as possible into the hardware layer.

3. **Network related events**: as we have mentioned, the access to the configuration of the WP8 device is much more restricted to the user, providing no information on the current connections or the networking events, other than enabling or disabling a network interface and changing the priority of a WiFi network in the list of preferred networks.

4. **Application Programming Interface (API)**: WP8 exposes a set of APIs that comprises the ones for previous releases (WP7) and the ones that the manufacturer recommends for applications built from scratch for WP8. We are only going to mention here the ones recommended for this system, which are *.NET HttpClient* and *WinRT Sockets*, and the so-called "Native code" (*IXML HttpRequest2* and *WinSock*). Although there are several development frameworks, we can differentiate two parts: the one devoted to the web browsing (HTTP-related) and the one directly related to the network connections (sockets). Despite the restrained flexibility given to the user, WP8 offers its applications information about the network connectivity, when a change in connectivity occurs and an application can even set preferences on the use of one interface over the other. Moreover, the emulator provided by the development framework can also simulate changes in the network connectivity to test the application under different scenarios.

5. **Flexibility and restrictions to the user**: WP8 tries to optimize mobile user experience and is designed taking into account the performance of current devices (touch-screen, portable devices, wireless connectivity). However, this optimization takes place by implementing a more integrated system, which speeds up some processes but does not involve more freedom

for the user in configuration nor accessibility. The manufacturer's claim to favor this design is that the user just wants to be connected, but does not care about how they get connected.

4.1.4 Experimental setup

Table 4.2: Main characteristics of the analyzed smartphones

	LG Nexus 4 E960	LG Nexus 5	iPhone 3GS	iPhone 4	iPhone 5	HTC 8S
OS Version	Android 4.2.2, 5.0	Android 5.0	iOS 6.0	iOS 7.0.4	iOS 8.1.2	Windows Phone 8.0
Chipset	Qualcomm Snapdragon S4 Pro APQ8064	Qualcomm MSM8974 Snapdragon 800	Samsung APL0298C05	Apple A4	Apple A6	Qualcomm Snapdragon S4 Plus MSM8227
CPU	Quad-core 1.5 GHz Krait	Quad-core 2.3GHz Krait 400	600 MHz Cortex-A8	1 GHz Cortex-A8	Dual-core 1.3GHz Swift	Dual-core 1GHz Krait
RAM	2GB	2GB	256MB	512 MB	1GB	512MB
WLAN	Atheros WCN3660 Murata SS2908001	Broadcom BCM4339 (5G WiFi combo)	Broadcom BCM4325	Broadcom BCM4329	Murata 339S0171 Broadcom BCM4334	Atheros WCN3660

This section describes the characteristics of the mobile terminals under test and provides an overview of the different experiments performed. We aim at studying the connection manager in several mobile devices running the most representative OSes – iOS, Windows Phone and Android. The characteristics of the smartphones in our experiments are collected in Table 4.2. We have included in our analysis the latest versions available of the OSes, including iOS8 and Android Lollipop. We have tested the same operating system version running in different terminals, so we can avoid dependency on the terminal rather than on the OS. For the sake of fairness we have avoided performing any modification to the terminals.

In addition to the mobile terminals under study, we deploy two IEEE 802.11 Access Points that provide Internet access. These Access Points are under our complete control to keep track of the behavior of the terminals attached to them and to be able to modify network parameters, such as the ESSID (Extended Service Set Identifier), the wireless channel in which the Access Point (AP) operates and the IP subnet managed by the access router.

We intend to characterize the Connection Manager on the systems under study, identifying the main strengths and weaknesses on the handling of network connectivity and comparing their behavior under different study cases. Our analysis focuses on understanding the following mechanisms:

1. **Initial attachment procedure to an 802.11 network**. Regarding this mechanism, we aim at understanding: *i)* how the attachment to a WLAN is carried out by every device, analyzing the differences among them, if any, and, *ii)* how the network selection algorithm works and what criteria are used to choose among the different candidate networks. This is explained in Section 4.1.5.1.

2. **Initial configuration of the protocol stack**. Once the mobile terminal has attached to a point of access, we aim at understanding the main steps and the protocols used to complete its networking stack configuration. Note that, if this procedure takes place entirely whenever there is a change in the point of attachment to the network, and not only as an initial configuration, it may enable potential optimizations for the handover process. This operation is explained in Section 4.1.5.2.

3. **Horizontal handover**. We examine the handover procedure between two IEEE 802.11 APs. We play with different network parameters to have a wide view of the performance for the different mobile terminals. Specifically, the current AP and the target one may have the same or different ESSID, operate in the same or different channels and manage the same or different IP subnets, in which case the handover would imply also a layer three reconfiguration. Through this analysis, we aim at knowing whether there are any dominant factors when the mobile device changes its point of attachment to the network and to what extent the different

changes impact the configuration and connectivity management. The horizontal handover is explained in detail in Section 4.1.6.1.

4. **Vertical handover**. We evaluate the handover procedure when it involves a change in the access technology. We aim at understanding how the mobile devices handle the inter-technology handover, whether they can keep both technologies operative simultaneously and whether they handle the survival of ongoing connections. Characterizing the inter-technology handover is essential to design potential optimizations and flow mobility solutions. However, due to restrictions by the terminal and the network operator, we were not able to obtain direct measurements from the cellular interface. The vertical handover results are presented in Section 4.1.6.2.

5. **Application behavior**. We study application survival to handover in the cases already explained. The objective is to know the perception of the user when an application is running and there is a change in connectivity. In addition, we get to know whether applications can handle an interruption due to horizontal or vertical handover seamlessly. These experiments are presented in Section 4.1.6.3.

We start our analysis in Section 4.1.5, with the evaluation of the initial attachment procedure to an IEEE 802.11 network.

4.1.5 IEEE 802.11 Initial attachment procedure

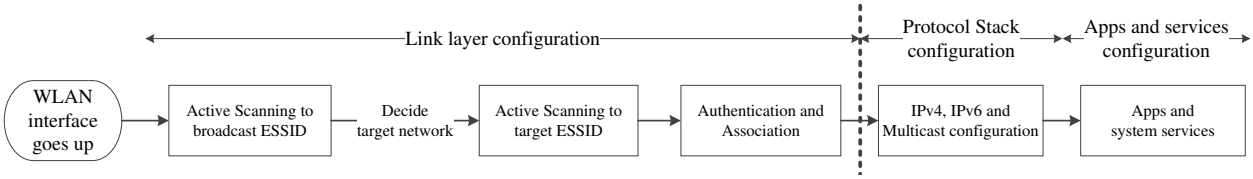

Figure 4.1: Initial attachment to the WLAN.

In order to test the default attachment to an IEEE 802.11 [15] WLAN, we deploy a wireless access point and run thirty experiments for each smartphone. In each experiment, which lasts for 60 seconds, we monitor the traffic by means of a network analyzer[9]. The monitor interface is located close to the access point, so we can likely capture all the frames involved in every exchange. Initially, the WLAN interface of the mobile terminal is down, so we do not miss any packet or reach misleading results because of having the device already connected to another network. We start our experiment by bringing up the interface and checking that the device actually connects to the AP under our control. This experiment describes the case in which the terminal finds an already known network and connects successfully. Note that to connect to a network for the very first time the user must identify and select manually the network to connect to.

Figure 4.1 presents in a diagram the different steps performed during the initial attachment to the IEEE 802.11 network. Note that this is a general diagram and it does not try to highlight the differences among the terminals, but to illustrate the common procedures followed by all of them. In the following, each of the steps in the initial attachment is explained in detail, highlighting the differences among terminals.

[9] http://www.wireshark.org/

4.1.5.1 IEEE 802.11 attachment

Active scanning to broadcast address: when the WLAN interface of a mobile terminal goes up, it detects all the surrounding wireless networks available by initiating an active scanning procedure. The terminal sends Probe Request frames to a wildcard ESSID (and to every previously visited ESSID) in every channel sequentially. Neighboring access points will receive these Probe Requests and answer with Probe Response messages, indicating their capabilities and providing synchronization information. This active scanning phase differs slightly in the systems under study.

By monitoring the traffic in different channels we are able to measure the delay induced by the scanning procedure. The results show high variability in the amount of Probe Requests being broadcast as well as in the time interval between them, but we can identify different patterns:[10]

- The Android terminal scans approximately every 10 seconds in every channel, sending a number of consecutive Probe Requests inter-spaced approximately 15 ms.
- The Windows terminal spends approximately 6 seconds between consecutive scans in the same channel. The time lapse between consecutive Probe Requests is very variable, but shows two different dominant values: the terminal sends several requests every 6 ms and one after 60 ms to continue with 6 ms interval again.
- The iPhone terminal presents an interval of approximately 9 seconds between the scan in every channel and the delay between consecutive Probe Requests is around 20 ms. As the WP8 terminal, there are several requests spaced 20 ms (much higher interval than WP8) and one 700 ms after that, to continue sending every 20 ms again.

It is also interesting to evaluate the behavior of the terminals in channel 14, which is not allowed in Europe, where we are based. The Android and Windows phones do not list the networks operating in that frequency as available, but the iPhone does, regardless of the regulatory domain. However, the three terminals send Probe Request frames in every channel including that out of the allowed frequency band (at a different interval than in the other channels, though). Noteworthy, Android and Windows devices do not give the user the opportunity to attach to these networks although they actively scan that channel.

The scanning policy followed by a terminal has several potential effects. First, the number of Probe messages sent during the scanning phase impacts directly the energy consumption. Second, a terminal can only obtain information regarding the signal level from the AP by the frames received, hence receiving more responses before deciding the target point of attachment can be beneficial. Last, the delay in the attachment to a WLAN AP is directly influenced by the time spent in the active scanning process, and so is the handover delay if the terminal has to scan again before connecting to a new AP.

Target Network Decision: this step corresponds to the actual decision on the AP to connect to. It seems reasonable to expect the terminals to perform some kind of complex algorithm considering, for instance, the signal level received from the different APs. After the analysis, we have discovered through our extensive tests that all the terminals use a simple rule to decide where to connect to. If the AP used in the immediately previous connection is available, the terminals will connect to it, no matter its signal level. In case the last visited AP is not available, the terminal connects to the one previous to the last one, and so on. In the case secured and open networks are available, only the iPhone terminal shows a preference for secured networks if the immediately previous connection is not possible. It is worth to note that the Android phone includes an option in the WiFi settings to connect to a different WLAN or to the cellular network if the signal is weak. However, even if this option is enabled, the terminal follows the same approach and disregards

[10] Due to the high variability we just provide rough numbers.

signal strength information to connect to an AP. This way of choosing the target network leads to weak connections and poor performance.

Active scanning to selected ESSID: in this step, the terminal addresses a Probe Request message directly to the AP selected previously and indicates the target ESSID in the correspondent field in the frame.

Table 4.3: Initial attachment delay

OS Version	Link layer delay (s)	Network layer delay (s)		Total (s)	
		IPv4	IPv6	IPv4	IPv6
Android 4.2 Nexus 4	0.42 ±0.12	1.09 ±0.31	4.75 ±0.43	1.51 ±0.33	5.17 ±0.45
Android 5.0 Nexus 4	0.47 ±0.05	0.68 ±0.36	9.58 ±3.53	1.15 ±0.33	10.05 ±3.54
Android 5.0 Nexus 5	2.58 ±0.05	0.50 ±0.04	3.33 ±1.15	3.09 ±0.06	5.90 ±1.14
iOS6	1.91 ±0.08	0.13 ±0.01	1.16 ±0.06	2.05 ±0.07	3.08 ±0.14
iOS7	1.98 ±0.26	0.13 ±0.02	1.12 ±0.58	2.12 ±0.26	3.1 ±0.67
iOS8	2.28 ±0.47	0.25 ±0.11	0.87 ±0.25	2.54 ±0.45	3.25 ±0.51
WP8	1.08 ±0.28	1.1 ±0.12	1.28 ±0.15	2.18 ±0.30	2.36 ±0.32

Authentication and Association: these are the last two steps before being attached to a WLAN AP. Both of them are standard procedures and are equally executed in the different terminals. Security procedures are out of scope as the main focus in this work is on the Connection Manager.

Table 4.3 gathers the average and standard deviation of the delay during the initial attachment to a WLAN measured for the different OSes versions. We distinguish between link layer and network layer configuration, which in turn is measured for IPv4 and IPv6 configuration, in order to provide more information on the influence of both processes. The link layer delay is measured as the time from the interface going up until the terminal is associated to the AP. The scanning policies presented previously influence considerably these delays. The Android terminal clearly outperforms the rest of the systems. Although it sends a higher number of frames before attaching to the selected AP, these frames are more frequent, whereas the other systems have a longer interval and therefore, the association is delayed. However, Android Lollipop has clearly impaired performance in the connection for Nexus 5 devices (well-known issues are being reported by Nexus 5 users since updated to Lollipop). The WP8 delay doubles that of Android 4.2 and Nexus 4 Lollipop, although the scanning time in every channel is lower for Windows. There is no clear difference in performance between iOS6 and iOS7, so this process seems not to have suffered major modifications from one version to the other. Still, the latest version, iOS8 increases the delay, as it happened with the Android update. We have had access to a new iPhone 6 and been able to perform the same tests, finding no difference, on average, in our measurements with respect to the ones in Table 4.3, which correspond to an iPhone 5. It calls our attention the usage of *CTS-to-self* frames sent by Android Lollipop in Nexus 5 and iOS8 in iPhone 6, but not in previous terminals running the same operating system. Moreover, Nexus 5 just sends one frame right before the authentication frame. The process and the delay for IP configuration is explained in detail in the next subsection.

4.1.5.2 Protocol stack initial configuration

IPv4, IPv6 and Multicast configuration: we group these configuration steps because they are very similar for the systems being evaluated. Table 4.3 shows the delay during the initial attachment due to the configuration of an IP address in the wireless interface of the mobile terminal, once it is associated to the AP (under the column "Network layer delay"). In the case of IPv4, the different terminals follow the same mechanism and use DHCP (Dynamic Host Control Protocol) [28] to

configure the IP address when they attach to the network. In our experiments, the DHCP server is located in a node belonging to the same network but different from the AP. The iPhone terminals are the fastest ones in this case because they start the process much earlier, while Android 4.2 (ICS) and WP8 present a very similar delay, close to one second over the one from iOS. Contrarily to the attachment to the WiFi AP, the IPv4 address configuration has been made faster in the new Android version. Another interesting difference is that the new iOS version, iOS8, uses gratuitous ARP in the IPv4 configuration when the wireless interface goes up, for both iPhone 5 and iPhone 6. Neither Android nor WP8 use this kind of frames for the IP address configuration when they attach to the network.

Regarding IPv6 configuration, the WP8 device tries to configure an IPv6 address by means of DHCPv6 [29], but as we have no DHCPv6 server in our network, all the terminals configure local and global IPv6 addresses following the SLAAC procedure as specified by [30] and [31] for configuration and Duplicate Address Detection (DAD). First, the mobile node acquires a link local IPv6 address and joins the all-nodes and the solicited-node multicast addresses when the connection is established in the interface. Then, in order to perform the DAD the terminal sends a Neighbor Solicitation message to the solicited-node multicast address. As the source address of this message is the unspecified address, any other node will not respond to that message and will identify the tentative target address on the message so as not to be used. Whether the node itself receives its own Neighbor Solicitation depends on the particular implementation of the multicast loopback. In the case of iOS devices, the delivery to upper layers of their own multicast message is disabled, so, if no other node in the network has the same IPv6 address as the target one, no Neighbor Advertisement from any other node will be received and the mobile terminal will silently configure the interface with the target IPv6 address. However, the Android and Windows terminals send out a solicited Neighbor Advertisement with their own source address upon receiving their own Neighbor Solicitation messages to announce this configuration. The Android device unicasts the message to the router, while the Windows device broadcasts it to all the nodes.

In light of the results in Table 4.3, the delay for the IPv6 configuration is comparable in the iOS and Windows Phone systems, but the Android device takes significantly more time, which delays any IPv6 connection. It is worth to mention that the three families of mobile OSes follow [7] considerations to protect privacy. According to SLAAC rules, the IPv6 address is configured from an interface identifier (EUI-64 identifier), and the second half of the global IPv6 address (without considering the 8-byte prefix announced by the router for configuration) is the same regardless of the location, so the device could be tracked. Therefore, IPv6 privacy extensions are defined so the network interfaces are configured with randomized strings, which change over time, instead of the interface identifier in order to complicate the activity correlation. RFC 7217 [32] provides the specification for the generation of these random interface identifiers while keeping IPv6 addresses stable in each visited subnet. Typically, the address derived from the EUI-64 identifier is kept, in addition to a temporary address built upon randomized identifiers. In our tests, we have observed that only the Android device configures an IPv6 address matching its EUI-64 identifier and a randomized one. The WP8 and the iOS devices configure the two IPv6 addresses from random strings. According to the RFC, "devices implementing this specification MUST provide a way for the end user to explicitly enable or disable the use of temporary addresses"; however, none of the systems are compliant with this statement. Table 4.3 shows a considerably higher delay for network layer configuration for the Android (Nexus 4) terminal. It starts the IP address configuration process approximately 1 second after the association, starting with the Router Solicitation message, and approximately 4 seconds after the association, it starts with the SLAAC, which leads to the highest delay among the terminals studied. Unfortunately, this process has been impaired significantly in the updated version.

Finally, the use of multicast for the interface configuration is slightly different in the three families

of systems. For instance, Windows Phone makes use of LLMNR (Link Local Multicast Name Resolution) protocol [33] in addition to IGMP (Internet Group Management Protocol) [34] and MLDv2 (Multicast Listener Discovery) [35, 36] which are used in the iPhone signaling. The Android phones just makes use of MLDv2 messages, as IGMP is not supported. This behavior is specific of some Android devices and constitutes a known issue in the community.[11]

Higher layer configuration: As part of the process to gain Internet connectivity, all mobile

Table 4.4: DNS queried for **initial configuration** of services on **WLAN interface start-up**

Service	Android	iOS	WP8
Initial connection to central servers	clients3.google.com (IPv4 and IPv6)	www.apple.com; (IPv4 and IPv6)	login.live.com clientconfig.passport.net
Network status indication	clients3.google.com/generate_204 www.google.com/blank.html	http://www.apple.com/library/test/success.html (iOS6) randomized (iOS7)	www.msftncsi.com
push notification service	mtalk.google.com	18-courier.push.apple.com (IPv4 and IPv6)	push.live.net
Software updates	play.googleapis.com android.clients.google.com (IPv4 and IPv6)	www.apple.com; (IPv4 and IPv6)	ctldl.windowsupdate.com crl.microsoft.com (IPv4 and IPv6)

OSes perform a technique called Network Connectivity Indicator, to detect captive portals. This protocol issues a DNS query to establish a TCP connection intended to send an HTTP GET method and retrieve a light-weighted web page, which is mainly void. This procedure serves to check the Internet connection at the device and prompt the users with a login form to introduce their credentials, if required by the WLAN administrator. The API in Android includes a way to access the information on whether the terminal is connected or connecting to the Internet, through which interface and allows registering to event notifications in case of network status changes. In addition, application developers have the option to try and reach a website from their app when it starts running, because the connection manager can only detect the status of the interface, but not if there is actual connectivity. With regard to Apple's devices, it is worth highlighting a recent change performed in iOS7 compared to iOS6. The captive portal detection in iOS6 was performed issuing a query to the web page `http://www.apple.com/library/test/success.html`, while in iOS7 this web page check has been swapped by a randomized query to a url selected in a list identified by Apple. Table 4.4 collects the service connections that are preceded by a DNS query when the different terminals attach to a WLAN. As soon as the mobile terminals gain Internet connectivity, all the terminals try to reach the central servers of their correspondent manufacturers to update their location, configure some services - time synchronization or push notification service – and re-initiate some connections – as GTalk, in the case of Android. IP addresses on the server side are expected to change, so the terminal connects by hostname, issuing DNS queries, rather than by IP address. Commonly, servers implement a load balancing scheme, so it is possible that the same query returns a different IP address for the same host name. For that reason, to identify the connections we track an IP address block, instead of a specific name or address. In addition we observe that many providers and applications use CDN (Content Distribution Network) nodes to offer their services, which complicates the identification of the service as the connections are hidden by the CDN. For instance, iPhone uses the Akamai network [37] for all Apple related services.

To sum up, the results show that the Android terminal associated to the AP much faster in the previous versions of the system, and the network layer configuration time is comparable to that of WP8 for IPv4, but it takes significantly longer than in the other terminals for IPv6 configuration and DAD. However, it calls our attention that, as Linux does, the first DNS query sent by the Android phone is always a request for an IPv6 address (type AAAA), so IPv6 takes precedence

[11] Issue 51195: many devices have multicast disabled in the kernel. `http://code.google.com/p/android/issues/detail?id=51195`

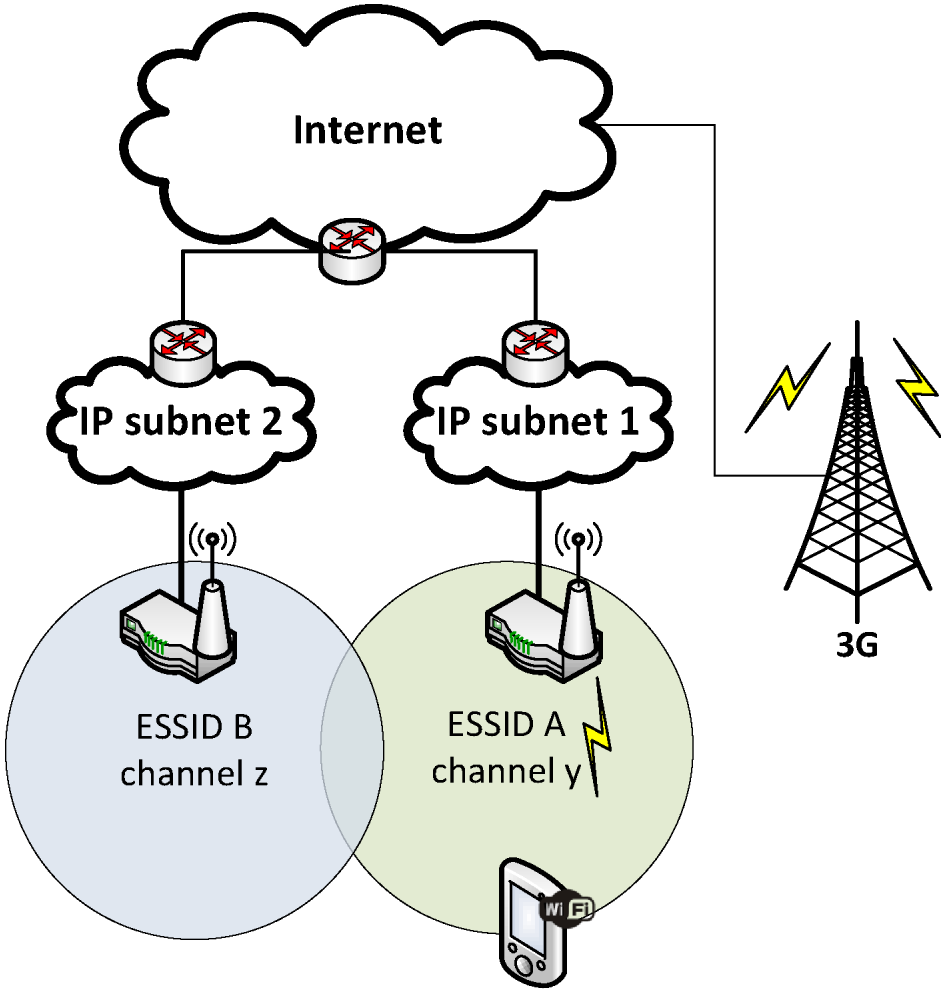

Figure 4.2: General scenario for handover tests.

over IPv4. On the contrary, the WP8 device issues the IPv4 query before the IPv6 one. Note that the differences in hardware, as reported in Table 4.2, should not be responsible for the differences in performance, specially in the case of the Android phone, which is the one with the slowest initial connection. In the case of the iOS terminals, they require much less time than the Windows and Android devices to configure an IPv4 address, but the association to the wireless AP takes longer (except for Nexus 5). The fast configuration in IPv4 makes possible to have comparable total times for iOS and Windows devices, while for IPv6 the Windows terminal is almost 1 second faster than iOS and almost 3 seconds faster than Android.

4.1.6 Handover dissection

This section presents a thorough study of different handover scenarios and their effect on applications and user experience. For these experiments, we vary the configuration of two 802.11 APs in order to cover as many different scenarios as possible. The different variations in the setup are presented in Figure 4.2. We deployed two access points that provide Internet access to the terminals attached to them and we play with different parameters of the network – ESSID, channel and IP subnet – to evaluate their influence when the handover process takes place from one AP to the other.

We do not only analyze the handover from an 802.11 AP to another, but also the inter-technology,

or vertical handover, moving the connection from the cellular interface to the WLAN one and vice versa. Lastly, we evaluate the behavior of several applications when a handover takes place to check handover impact from the point of view of the user.

4.1.6.1 WLAN Horizontal Handover

Initial considerations

Table 4.5: Layer 2 handover delay [s] for the different terminals

			Handover mechanism			
			Manual	Forget	disconnect AP	
					AP deauthenticates	Connection lost
Same ESSID	Same channel	A 4.2	N/A	N/A	0.92 ±0.13	5.54 ±0.09
		A 5.0 Nexus 4	N/A	N/A	7.20 ±2.54	21.09 ±6.39
		A 5.0 Nexus 5	N/A	N/A	9.12 ±1.72	2.96 ±1.06 (reassoc) 13.14 ±2.66
		iOS6	N/A	N/A	1.94 ±0.36	4.63 ±1.12 (reassoc)
		iOS7	N/A	N/A	2.48 ±0.29	4.59 ±0.59(reassoc)
		iOS8	N/A	N/A	2.44 ±0.81	11.23 ±3.33 (reassoc)
		WP8	N/A	N/A	0.24 ±0.1	
	Different channel	A 4.2	N/A	N/A	0.99 ±0.19	5.40 ±0.11
		Lollipop Nexus 4	N/A	N/A	3.25 ±0.26 (reassoc)	2.71 ±0.37 (reassoc)
		Lollipop Nexus 5	N/A	N/A	9.05 ±4.68	13.40 ±4.60
		iOS6	N/A	N/A	1.57 ±0.34	5.66 ±1.38 (reassoc)
		iOS7	N/A	N/A	2.38 ±0.32	4.88 ±0.97 (reassoc)
		iOS8	N/A	N/A	14.80 ±5.94	18.87 ±1.87
		WP8	N/A	N/A	0.4 ±0.097	
Different ESSID	Same channel	A 4.2	0.95 ±0.08	0.9 ±0.03	0.78 ±0.29	5.42 ±0.1
		A 5.0 Nexus 4	0.10 ±0.08	1.04 ±0.45	5.31 ±3.02	10.73 ±3.18
		A 5.0 Nexus 5	0.09 ±0.04	4.70 ±1.78	8.53 ±3.81	9.75 ±4.83
		iOS6	0.14 ±0.054	1.53 ±0.10	1.36 ±0.53	11.87 ±0.83
		iOS7	0.22 ±0.03	1.64 ±0.33	2.41 ±0.67	11.88 ±0.36
		iOS8	0.12 ±0.06	0.27 ±0.06	3.90 ±1.90	11.44 ±2.23
		WP8	0.87 ±0.34	1.52 ±0.62	10.14 ±1.83	
	Different channel	A 4.2	0.93 ±0.07	0.91 ±0.025	0.91 ±0.06	5.41 ±0.15
		Lollipop Nexus 4	0.27 ±0.28	3.29 ±0.31	9.07 ±1.65	15.09 ±3.39
		Lollipop Nexus 5	0.10 ±0.08	4.24 ±1.30	6.87 ±3.26	13.12 ±4.48
		iOS6	0.22 ±0.06	1.43 ±0.45	1.09 ±0.82	12.00 ±0.51
		iOS7	0.28 ±0.04	1.69 ±0.61	1.20 ±0.26	12.16 ±0.887
		iOS8	4.92 ±0.03	0.17 ±0.08	8.61 ±4.12	12.12 ±2.38
		WP8	0.73 ±0.05	0.75 ±0.05	10.41 ±0.065	

In this section we focus on the main core of the handover analysis, which is the intra-technology or horizontal handover, where the mobile node changes the point of attachment to the WLAN and connects to a different WLAN AP. Table 4.5 gathers the link layer delay measured for the different experiments. We have indicated the differences in ESSID and channel of operation between the two APs. Our analysis is centered on the link layer handover, so we do not change the IP subnet. As none of the three mobile OS families bases a handover decision on the received signal strength from the current AP or link quality degradation, we force a handover in our experiments by initiating it in the network or in the terminal side following three different approaches: *i)* the AP deauthenticates the station (since the mobile terminals do not react to signal strength or link quality degradation, we turn off the AP). This is presented in Table 4.5 as "disconnect AP" in light of the obtained results we differentiate two subcases; *ii)* the mobile user terminates the connection by manually omitting – "forgetting" – that network (referred to in Table 4.5 as "forget") or *iii)* the mobile user switches the connection directly to other network ("manual" in Table 4.5). The difference between

the second and the third option is that, in the latter, the user explicitly indicates the target network to switch to.

Another preliminary consideration is that the cellular interface is the default interface in smartphones, since it provides always-on connectivity. Therefore, we also evaluate the influence of having this interface enabled or disabled in the case of a horizontal WLAN handover. We have noticed that the OS families analyzed always fall back to the cellular connection as soon as the current WLAN connection fails, even if there are other WiFi networks available. This change involves another variation of the IP address that provides global connectivity to the device, however, it does not necessarily worsen the handover latency and the interruption experienced by the user. On the contrary, the change to the cellular connection actually does not increase the handover delay and, as we will see in Section 4.1.6.3, improves the performance in case the handover implies a change of IP subnet, contributing to the survival of a running application, depending on the implementation. For the experiments presented in this section, we have confirmed that having the cellular data connection enabled or disabled makes no significant difference in the horizontal handover delay, so we do not distinguish these two cases in Table 4.5 for the sake of clarity.

Handover latency

The first issue that calls our attention is the considerable handover latency for the systems under study in every scenario. However, it is remarkable that this latency does not always translate into a complete loss of connectivity or killing a running application – see Section 4.1.6.3.

From the figures in Table 4.5 we can clearly see that the fastest way of handing over from one WiFi network to the other is to manually change the connection. All the terminals can change the connection in less than a second, but iOS devices are particularly fast, with times around 200 ms. In the case the handover is initiated by the user but just deciding to disconnect from the current AP, without directly connecting to the new one ("forgetting" the current connection) the handover latency increases to values around 1.5 seconds as the mobile terminal scans again in every channel. However, the Android terminal presents a stable delay as the handover delay remains around 0.9 s when the mobile terminal hands off between two APs from different ESSs (Extended Service Sets), regardless whether the handover is initiated by the terminal or the network. For the rest of the systems, instead, it varies significantly. It is remarkable the stickiness of the iOS8 wireless client, which tries to remain connected to the same AP regardless of network conditions or even user choices, especially when having to change to a different channel, which explains the 4.92 s delay in the "manual" handover and the 0.17 s for the "forget" handover, as the terminal cannot try to remain connected to the previous AP.

In the case of roaming between two APs within the same ESS, the only possibility is to hand off by disconnecting the AP, because the user cannot choose manually to which AP in the ESS it connects. When the handover is triggered from the network side, the AP sends a deauthentication frame when it disconnects. In this case, we have identified two differentiated behaviors that repeat in our experiments, except for the WP8 terminal. In the first case, the mobile terminal receives the deauthentication from the AP and starts active scanning for a new AP to connect to. This case shows lower delays and the signalling is similar to the one described in Section 4.1.5 for the initial attachment. On the contrary, in the other case we have measured much higher delays until the mobile terminal associates to the new AP. The reason for this difference is that the mobile terminal does not get disconnected because it recognizes the deauthentication from the AP, but because the connection is lost (e.g. missed Beacons). The mobile terminal tries to reconnect to the lost AP by sending Null Function frames and Probe Requests. As the AP does not respond anymore, the mobile terminal needs to scan to look for other available APs. Once again, the Android terminal presents a similar delay irrespective of whether the former and the new AP are part of the same

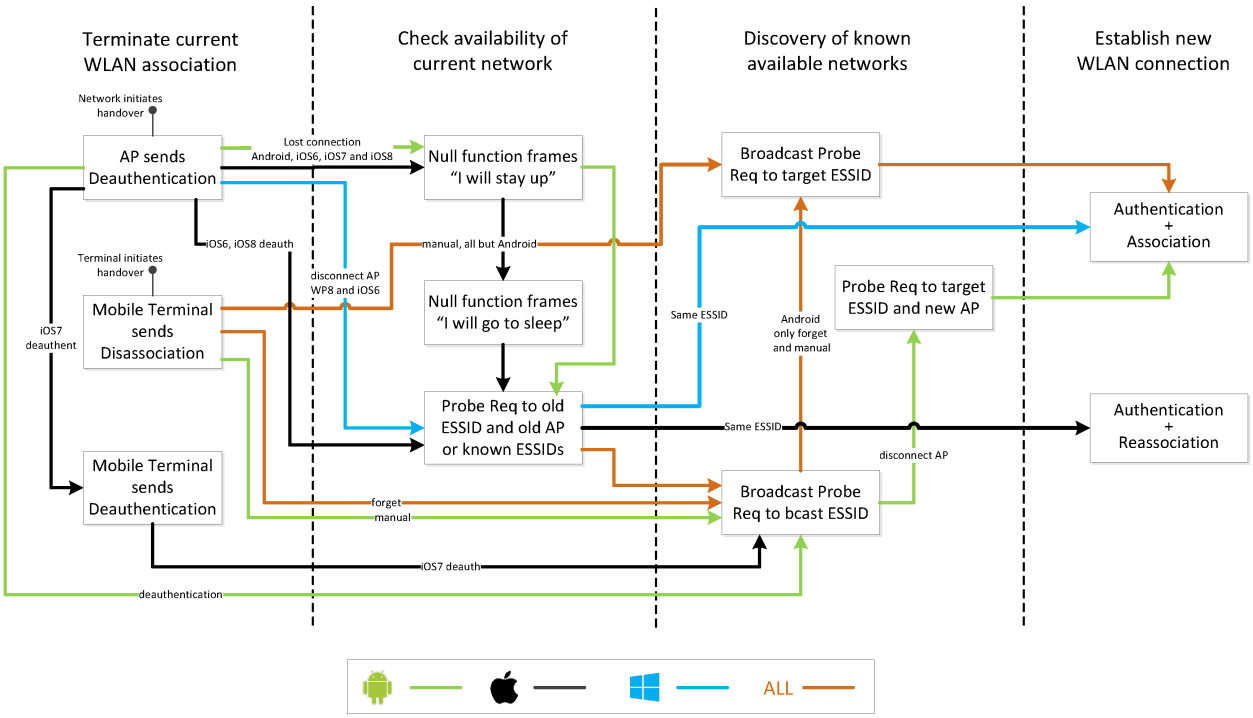

Figure 4.3: Flow diagram of a handover procedure for the different OS families.

ESS (around 5 seconds). However, the iOS devices perform a re-association in approximately 5 seconds when roaming within the same ESS, whereas the delay raises to 12 seconds when the two APs are in different ESSs. We have traced the events that take place in the case with the higher delays in the wireless networking stack until the wireless driver. However, we cannot confirm if this is a buggy behavior of the implementation, but it clearly gives us some room for improvement in the connection management. The Windows Phone terminal does not present these differences in its behavior. Its handover process is more stable although moving within the same ESS clearly decreases the delay and reduces the scanning phase. It is worth to mention that, when roaming within the same ESS the iOS devices send a Reassociation Request frame instead of an Association Request to the new AP. The main difference between these two frames is that the Reassociation includes the BSSID (Basic Service Set Identifier) of the previous AP that the mobile terminal was connected to. This is because from iOS6 Apple has implemented support for 802.11r (amendment for Fast BSS transition). Since our APs do not implement 802.11r mechanisms, this feature reduces to a regular handover, although having it enabled, should influence significantly aspects like security or QoS.

If the handover is performed between two APs within the same ESS, this change should have an effect only at the link layer, being transparent to the IP layer. Therefore, the mobile terminal should not renew its DHCP lease until it is expired even though it changes from one AP to another inside the same ESS. This behavior is confirmed by the iPhone terminal, but both Android and Windows phones initiate a DHCP discover process when they connect to the new AP, regardless of being part of the same ESS.

In these experiments we do not consider changes in the IP layer, as we are focusing on the link layer delay. Nevertheless, there is another relevant characteristic, as for the Android and Windows phones any kind of handover involves a reconfiguration in the IP layer (new DHCP and IPv6 configuration), even though the target network is managing the same IP subnet. As we see, the rest of features do not introduce significant changes in the handover process, neither in signaling or in delay.

Link layer behavior

Figure 4.3 illustrates the signaling differences among the four systems under study for the four handover cases we have analyzed. The handover process starts either when the AP sends the deauthentication to the mobile terminal ("disconnect AP" case in Table 4.5) or the mobile terminal disassociates ("manual" and "forget" cases in Table 4.5). One of the main differences we can appreciate is that Android terminal does not make a difference between the "manual" and "forget" handovers. Both trigger the active scanning with broadcast Probe Request frames to the wildcard ESSID; then it follows targeting the specific ESSID to finally authenticate and associate. We also notice that the WP8 terminal follows a shorter path than the other systems when the AP sends the deauthentication (and the ESSID does not change), which matches the low delay of the WP8 handover in this case. Another difference is that the mobile terminal sends a Deauthentication frame only when it receives the deauthentication from the AP and only for the iOS7 device. Finally, the two different cases for the "disconnect AP" handover are illustrated by the two different paths that part from the "AP sends Deauthentication" box. The case that incurs the highest delay is the path through the Null Function frames, while the direct path to the active scanning box also involves considerably lower delays (as abovementioned in Table 4.5). In this last case, the terminals may follow two different paths, sending Null Function frames or not, depending on whether they accept the deauthentication from the AP or think the connection has been lost.

The use of Null Function frames is a common practice and, as their usage is not defined by the standard, different implementations make different use of them. On the one hand, they are used for power saving, notifying the AP when the station is going to sleep mode and waking up. On the other hand, they are used during active scanning to get the AP buffering the frames addressed to the station while it is sensing other channels to avoid retransmissions. A third alternative is to use the Null Function frames as a keep alive for the connection. However, the flexibility provided by these frames is also applied for attacking wireless networks [38].

Upper layers behavior

It is worth mentioning that the handover delay increases significantly in the case of not having an application running. Moreover, although the behavior on the link layer is quite similar in the different devices for any of the scenarios, the behavior in the upper layers is very diverse.

We aim to characterize also the interruption at upper layers due to the handover. However, we have not been able to extract similar results to the link layer measurements, due to the variability in the behavior of the application running and the high delay to re-establish the flow with the main servers. The application behavior is not under our control, but several cases can be identified. In the case of Android, every handover case involves a complete reconfiguration also at the network layer, reissuing DHCP discovery and performing DAD. Therefore, the interruption of any traffic flow is considerably long. When the device gains connectivity again, the TCP connection is reset (RST flag set) and any connection of an application running at the moment of the handover has to be renewed, sometimes even connecting to different servers – e.g. due to load balancing at the server. On the other hand, in the case both APs handle different IP subnets, this re-issue of DHCP discoveries enables a faster re-configuration of the IP layer. Otherwise, even though the connection at link layer takes place successfully, no data will be delivered to or from the mobile terminal as it does not reconfigure its IP address according to the new network. However, passing through the cellular interface during the transition between two WLAN APs enables the IP re-configuration, also within the same ESS. This is the case for iOS devices, which do not re-issue a DHCP discovery when there is a handover within the same ESS.

4.1.6.2 3G-WLAN Vertical Handover

One of the main differences among the operating systems under study is in the simultaneous usage of the cellular and 802.11 interfaces. iOS does not allow one interface to remain active when the other is getting started too, e.g. finishes the open connections and turns down the cellular interface when a known WLAN appears in range and tries to connect to it. On the other side, Android keeps the cellular connection on until the IP address is configured in the WLAN interface. Finally, Windows Phone allows the simultaneous connectivity through both interfaces, even keeps an active communication through the cellular interface although it gets attached to a WLAN AP, while applications started from that moment use the WLAN. This is an important advance over its competitors, as for instance, it allows to keep a VoIP (Voice over IP) conversation over the cellular network while connecting to an 802.11 AP, ensuring that the call will not be interrupted. In the case of Android 5.0, it includes a significant change in the network management. First of all, the simultaneous usage of cellular and WiFi interfaces is allowed. Therefore, the connections established through the cellular connection remain active through that interface even in the case the mobile terminal connects to a WLAN. If there are not active connections, the WLAN interface is still the preferred one, deactivating the IP connectivity in the cellular interface 30 seconds after the WiFi interface gets a connection. In order to introduce these changes, the development team has defined a routing table per interface, instead of a system-wide routing table, as in the previous Android versions. This has allowed to improve, as we show in 4.1.6.3 the performance of applications running when a handover occurs, for instance, in the case of an ongoing Skype call.

4.1.6.3 Application survival to handover

Following the different approaches for handover described in Section 4.1.6, we have studied the influence of these handover procedures in the behavior of several applications. We have chosen some of the most widely used applications, varying their scope. The applications we have evaluated are Skype, as a VoIP application (Android v3.2.0.6673, v5.1.0.57240 (Nexus4 Lollipop) and v5.1.0.58677 (Nexus 5 Lollipop); iOS6 v4.2.2601; iOS7 v4.17.0.123; iOS8 v5.11; WP8 v2.1.0.241); Youtube, as a video streaming application (Android 4.2 v4.2.16 and v6.0.13 (Nexus 4 and Nexus 5 Lollipop); iOS6 v1.1.0; iOS7 v2.3.1.11214; iOS8 v10.09.11358); Facebook, as the most relevant social network at the moment[12] (Android v3.3, v23.0.0.22.14 (Nexus 4 Lollipop) and v24.0.0.30.15 (Nexus 5 Lollipop); iOS6 v5.6; iOS7 v6.8; iOS8 v26.0; WP8 v4.1.0.0) and three applications for radio streaming, which will be referred to in the following as Radio 1[13] (Android v1.08.16 (ICS), v4.1.2 (Nexus4 Lollipop) and v1.08.33 (Nexus 5 Lollipop); iOS6 v2.1.7216; iOS7 v2.1.9327; iOS8 v3.0.0; WP8 v2.1.0.0), Radio 2[14] (Android v1.0 and v2.1.3 (Nexus 4 and Nexus 5 Lollipop); iOS6 v2.4; iOS7 v3.0; iOS8 v3.5; WP8 v1.3) and Radio 3[15] (Android v2.2.2 and v2.2.6 (Nexus 4 and Nexus 5 Lollipop); iOS6 v2.1.1; iOS7 v2.2.1; iOS8 v2.2.2; WP8 v1.2). We choose three different radio stations to avoid biasing the conclusions, because some features may be implementation-dependent.

The test procedure consists on start running the application under stable network conditions and perform a handover to analyze the effect of this change. The results of these experiments are presented in Tables 4.6 and 4.7 for intra-technology WLAN handover and inter-technology handover respectively. We have observed that the mobile devices under study fall back to the cellular interface as soon as the WLAN connection is lost, although other 802.11 networks are available and even attach to one of them immediately. Because of that, we have included another variant in our experiments apart from the different configurations of the WLAN, which is to have

[12] http://www.dreamgrow.com/top-10-social-networking-sites-by-market-share-of-visits-may-2013/
[13] The application tested is the one by *Los 40* radio station.
[14] The application tested is the one by *Cadena 100* radio station.
[15] The application tested is the one by *RNE* radio station.

the cellular data connection in the mobile terminals enabled or disabled. However, regardless whether the cellular interface is on or off, the delay (presented in Section 4.1.6.1) is not affected but only has an influence on the survival of the application, which is more likely to overcome the handover when the cellular interface is on. We have identified different application behaviors:

a) The application interrupts and does not recover even when there is global connectivity unless you restart the application. This is indicated in the table by a black cell.

b) The application interrupts for a small lapse, continues working intermittently and finally stops. This is indicated by a dark grey cell.

c) The application interrupts for a small interval, but it continues working properly and go back to its normal operation with a noticeable but acceptable glitch for the user. This is indicated by a light grey cell.

d) The application tolerates the handover, which happens smoothly and the interruption is seamless for the user. This is indicated by a white cell.

To gain some space, Youtube has been excluded from the table because there is no difference in the behavior of this application in any of the handover combinations. As long as the buffer does not empty, the user will be able to watch the video without noticing that there is a handover in progress. However, the application may stop if the handover does not allow to keep filling the buffer to continue playing the video. Facebook does not appear in the table neither for the sake of clarity, as this application can always recover from the loss of connectivity, retrying to load the user profile (or the target page) in several attempts, either automatically triggered by the application or reloaded by the user.

As recommended by the developer documentation of the three OS families, HTTP or HTTPS are the way to send or receive small pieces of information, and this is the way that all the Radio applications use for streaming their content. The only application in our experiment set that uses a different protocol is Skype.

The different mobile terminals analyzed fall back to the cellular connection as soon as the current WLAN connection fails, even if there are other known WiFi networks available. In order to mitigate the interruption in network connectivity, when the connection to an AP is lost, the mobile terminal uses the default connection through the cellular interface. Regaining IP connectivity by the 3G, gives more time to the mobile terminal to scan in the WiFi channels and connect to another AP if there are any other networks available. However, the change in the technology involves a change in the IP address in use for currently ongoing connections, and this may impact open connections more than the link layer handover, depending on the implementation. There are applications that can survive these two changes in the current connection, while others are interrupted as soon as the current network interface losses connectivity. Note that the applications that easily survive are those that can benefit from buffering the content (Youtube) but note that buffering does not necessary imply a seamless handover, as not all the applications that buffer the content can survive a handover (Radio 3) concluding that the survival of an application is implementation dependent. It does not depend neither on the OS nor the API offered to application developers, as in the same system different applications can overcome the handover interruption whereas others do not regain connectivity anymore – Radio 1 and Radio 3 in Windows Phone, respectively. In addition, in the case of the applications that hand-off successfully, even though the change of access technology back to 3G involves an additional change in the IP configuration, it helps improving the user experience making the interruption smoother.

In the light of the results in Tables 4.6 and 4.7 the first issue that calls our attention is the poor performance of the application Radio 3, which cannot survive the change of the point of attachment

Table 4.6: Survival to handover for applications of different nature in the three OS families. Intra-technology handover

			Skype				Radio 1				Radio 2				Radio 3			
			3G on		3G off		3G on		3G off		3G on		3G off		3G on		3G off	
			AP	user	AP	user	AP	user	AP	user	AP	user	AP	user	AP	user	AP	user
Same ESSID	Same IP subnet, same channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
	Same IP subnet, different channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
	Diff IP subnet, same channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
	Diff IP subnet, different channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
Different ESSID	Same IP subnet, same channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
	Same IP subnet, different channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
	Diff IP subnet, same channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																
	Diff IP subnet, different channel	A 4.2																
		A 5.0 Nexus 4																
		A 5.0 Nexus 5																
		iOS6																
		iOS7																
		iOS8																
		WP8																

Table 4.7: Survival to handover for applications of different nature in the three OS families. Inter-technology handover

		Skype	Radio 1	Radio 2	Radio 3
3G → WiFi	A 4.2				
	A 5.0 Nexus 4				
	A 5.0 Nexus 5				
	iOS6				
	iOS7				
	iOS8				
	WP8				
WiFi → 3G	A 4.2				
	A 5.0 Nexus 4				
	A 5.0 Nexus 5				
	iOS6				
	iOS7				
	iOS8				
	WP8				

Color guide for application behavior during handover in Table 4.6 and Table 4.7:
Application interrupts | Annoying, finally interrupts | Acceptable glitches | Seamless handover

to the network. Secondly, we identify that Radio 2 performs significantly better in the Windows Phone 8 than in Android or iOS. A user of this application in the Windows terminal can continue listening to the radio station with a seamless handover or with a minimal interruption in every case, while an Android or iPhone user would stop being provided the service. However, while the Android Radio 2 application fails in every handover, under any circumstances, the iPhone Radio 2 application shows different behavior under different scenarios and for iOS6, iOS7 and iOS8: *i)* for iOS6, the application cannot handle a handover that involves a change in the IP subnet; a change in the point of attachment within the same ESS is supported but, changing to a different ESS this application only survives if the 3G interface is enabled *ii)* for iOS7, Radio 2 interrupts only when IP subnet and channel change within the same ESS; *iii)* for iOS6, iOS7 and iOS8, the application tolerates inter-technology handovers, although the app for iOS8 is the one that interrupts the most. It is also remarkable the improvement from iOS6 to iOS7 versions of the same application, especially for Radio 2 and Radio 3, as well as in the case of iOS8, where Radio 1 does not interrupt when 3G connection is on and Radio 3 can always maintain the connection. However, Radio 2 does not show an improvement, as it interrupts playing.

The issue of changing the IP address is not trivial. A similar behavior is observed for Radio 1: the Windows terminal can handle the change of AP unless it involves a change in the IP address of the target subnet. Similarly, the iPhone terminal starts experiencing trouble even when the target wireless network has the same ESSID, but a different IP. When the target AP operates in the same channel, as the scanning process takes less time, the application can recover, but that is the only case. When the ESSID is different between both networks, the application running on the iPhone terminal can only handle the handover when the action that triggers the change comes from the network, but not when the terminal decides to terminate the connection. This has been overcome in iOS7, only if the connection can fall back to the cellular interface, as waiting for having configured the new WLAN network connection adds too much delay. However, the Android terminal can manage the change in connectivity for Radio 1 without interruption in every case.

By observing the results in Table 4.6 we can see that the applications running in the Android 4.2

device present a more homogeneous behavior and mostly do not tolerate the handover, but for the application Radio 1. However, the Android update 5.0, has allowed to maintain an ongoing Skype call in several scenarios, while applications Radio 2 and Radio 3 still show the same poor performance and interrupt when handover occurs. It is important to note that although the application Radio 1 does not interrupt and keeps the transmission, the user actually hears the last packets before the handover repeated, so the sound flow keeps being heard, but if, for instance, listening a song, a part of it will be heard twice before recovering the flow streaming, with the subsequent impairment of user experience.

Last, in the case of Skype, the application running on the Windows phone outperforms the other two versions. This result was expected, being a proprietary solution by the same manufacturer. Among the rest, the Android implementation offers the poorest performance, not being able to survive the handover in any case. It is worth pointing out that in the case of inter-technology handover from 3G to WiFi, Windows Phone 8 and iPhone terminals are able to keep the call active for two different reasons: the iOS device turns down the 3G interface when the WiFi network becomes available, then, the call just survives the change of connection and takes advantage of the higher bandwidth of the WiFi network to overcome packet losses during the interruption. On the other hand, the Windows Phone 8 device keeps the ongoing call on the 3G connection, even though the WiFi connection becomes ready and active to be used by other applications. We have confirmed that this only happens when the WiFi connection becomes available while there is an ongoing call, but if the call starts when the device is already connected to a WLAN, every call will use that connection. In any case, Skype is the most sensitive application among the ones we tested. We cannot claim that it does not handle a change in network connectivity, but it will interrupt an ongoing call in case the communication between the two endpoints is lost for more than 15 seconds.

4.1.7 Summary

In this section we present a summary of the main findings for the different families of OSes and highlight the differences and features that called our attention, categorizing them into five groups:

Simultaneous usage of network interfaces: Out of the three OS families under study, Windows Phone 8 and Android Lollipop allow simultaneous usage of cellular and WLAN interfaces. The Windows Phone 8 device keeps active connections over the cellular and the WLAN interfaces simultaneously. Android, only in its latest version (Lollipop) modified its policy to allow simultaneous connection through both interfaces. The cellular connection remains active for 30 seconds after ongoing sessions finish. iOS devices finish the open connections on the cellular interface when a connection to a WLAN is established.

Network selection: None of the systems under study perform a network selection algorithm to decide on the best network available to connect to. They rely on the network used in the last connection, if it is available. In addition, none of the systems uses information on link quality or performance to change point of attachment if needed. iOS8 WiFi client is particularly sticky, even not responding to user choices and remaining attached to the current AP if the signal is not lost.

Connection establishment: Android and Windows Phone 8 renew their IP address by reissuing a DHCP discovery every time they connect to a different AP, but iOS does not if the change of AP takes place within the same ESS. In the initial attachment to a WLAN, Android outperforms the other systems in the link layer attachment, but it is considerably slower in the IP configuration. The WP8 terminal has proven to be the fastest one, on average, for initial attachment to an already visited WLAN. It calls our attention the remarkable impairment of performance in the connection establishment to WiFi networks in Android Lollipop running on Nexus 5. The delay in the connection establishment for iOS8 is also slightly higher than in previous versions, but the difference is not as notorious as for Lollipop (on Nexus 5). Although the IP configuration has been made faster, the connection to the AP has been considerably damaged. It is also new with respect

to previous versions, the use of Gratuitous ARP in iOS8 and the CTS-to-self frames sent before authentication frames by iOS8 (iPhone 6) and Lollipop (Nexus 5).

IPv6 configuration: The three OS families implement privacy extensions for SLAAC [7], configuring an IPv6 address that does not match their respective EUI-64 identifiers. Actually, this is not the only IPv6 address configured in the terminal interface, so applications should handle this and take into account that this kind of address will change over time, which may affect ongoing sessions. The first DNS query sent by the Android phone always requests an IPv6 address, so IPv6 takes precedence over IPv4. However, the IPv6 configuration takes a significantly longer period to be completed. On the contrary, WP8 issues first the IPv4 query but the delays for IPv4 and IPv6 configurations are comparable. The cellular networks available for our experiments do not offer IPv6 support for the moment. Although standardization bodies have provided guidelines for the migration to IPv6, to our knowledge, at the time of writing only some LTE networks in North America and Europe support IPv6 access.

Handover: Not having any application running increases delay in case of a handover. Android (except for Lollipop) presents a more regular behavior, having a handover latency of 0.9 s for most of the cases evaluated. WP8 and iOS devices present a more variable performance. Particularly, when a handover is initiated by a deauthentication from the AP (between different ESSIDs), the interruption in connectivity through the WLAN can take approximately 5 s, 10 s or 12 s on average for Android, WP8 and iOS systems respectively. However, when the handover happens within the same ESS it is completed in 0.24 s or 0.4 s by the WP8 phone, a result which outperforms the other two systems. When the handover is initiated by the terminal, the fastest handover (90 ms) is performed by the Nexus 5 Android Lollipop device if the user is manually indicating the target network, closely followed by iOS devices. Note that changing to a different channel, even if the user does it manually, increases delay considerably in iOS8, due to its sticky client implementation. However, Android and WP8 offer lower delays than iOS if the terminal needs to scan for available networks to decide on the new AP to connect to ("forget" case in our experimental results). Nevertheless, the new Android version offers a significantly higher delay (around 4 s) and the new iOS version (iOS8) overcomes the issues with the manual connection and lowers the delay to just 170 ms. Only the iOS devices and Android Lollipop perform a re-association when the handover takes place within the same ESS. This diminishes the handover delay for the Lollipop devices, but not for the iOS terminals. Although all the systems fall back to the cellular connection when they lose WLAN connectivity, this change does not increase the delay in the WiFi-WiFi handover. The change to the cellular network shades the interruption in the WiFi interface, allowing for time to perform the scanning and the association to the new AP. The remarkably bad performance of Android Lollipop and iOS8 deserve a special mention, especially in the case of changing to a different channel manually for iOS8 and as a general consideration for Android Lollipop.

Multicast and network traffic: Unsurprisingly, HTTP is the dominant traffic as it is also the recommended way to access remote content in the development documentation. It calls our attention the intensive use of IEEE 802.11 Null function frames. The use of these frames is not specified by the standard, but WiFi clients, especially in smartphones, send a significant amount of these frames regularly. The most extended use of Null Function frames is power management, so the station informs the AP when it goes to sleep or awakes. However, these frames are sent very regularly, and, specifically, when connection to the current AP is lost. In our handover experiments, we have detected that Android and iOS devices try to reach the AP, whose signal is lost, by sending numerous Null Function frames and Probe Requests, delaying the connection to a new point of attachment. It is also remarkable the number of DNS requests required every time that any connection is open. Regarding multicast support, IGMP is not supported in some Android devices, including Nexus 4 and Nexus 5, which we used in our experiments.

4.1.8 Open issues and future directions

The thorough assessment of the connection management in the three mobile operating system families under study highlights some flaws in current implementations. Our study reveals that the design of current mobile terminal OS and applications only takes into account the availability of Internet connection, but does not consider the presence of several access networks as a resource. In addition, current implementations do not optimize network access selection or handover and pay minimal attention to connection management, beyond identifying the interface being used and detecting Internet connection. To fill these gaps we identify some potential implementation changes that would be feasible, even easy to perform, and that would enhance user experience, reducing latency in the connection and the handover and improving efficiency in the handling of several interfaces.

4.1.9 Enhanced network selection

If several known WLANs are in range, all the systems analyzed connect to the one they were connected the last time. None of them takes into account the signal strength or link quality towards the different APs as a criterion to choose what network to connect to. If this was considered, handover after a short time could be avoided. Current drivers and firmware as the ones in smartphones have full access and capabilities to monitor certain key indicators of the performance or link quality. Keeping track of changes in these parameters and other decision policies are easily implementable in software tools like *wpa_supplicant*. Even further improvements, as supporting some of the recent IEEE 802.11 standard amendments, are already including in recent versions. The potential changes that we suggest as an example for access network selection are mainly related to the WLAN connection:

- **Connection to the best WLAN**: when the WLAN detects that already visited networks are available, instead of connecting to the last visited one, we suggest to connect to the one that provides the best link quality at that moment. Another variant of this selection is to keep track of the performance offered by the network in the previous connection – or keep an average measurement of historic connections – at that given location (similar approaches exist in the context of vehicular networks [39]). This choice would be customizable and applicable to different criteria, like security or delay instead of just throughput or signal quality when several of the networks offer similar characteristics.

- **Reduced scanning during handover**: since the mobile terminal keeps sending Probe Request frames after attaching to the WLAN AP, this information could be used to speed up the process of handover, shrinking the interval that the terminal spends scanning again after disassociating from the previous AP and connecting to the new one. Moreover, the mobile terminals keep sending Probe Requests frames to all the visited networks. As the terminal already has all the information about the user location and movement, the scanning could adapt to the user location, only scanning for nearby networks, reducing considerably the number of frames being sent regularly.

- **Early detection of low link quality**: similarly to prioritizing the connection to the AP that offers the highest signal quality, this link quality should be monitored, given the dynamic nature of wireless networks. This monitoring would allow a quick reaction when the link quality gets low, making the current connection likely to fail. Moreover, we could take advantage of the considerable amount of frames that are exchanged with the AP constantly, as we have checked that, apart from the active scanning, the three OS families that we have evaluated send 802.11 Null Function frames at all times for power management and signaling

purposes. Depending on the terminal capabilities, the current open connections could be handed-off to the cellular interface and start trying to connect to a new WLAN in order to avoid interruptions before the current one fails.

- **Get information from network repositories**: standardization bodies have made an effort in the specification of different information repositories and distribution such as ANDSF [40], ANQP [41], ALTO [42] and MIIS [43] [44]. Their generalized deployment and the access to this information can help the mobile device to choose the best access network to connect to, increase performance in network-assisted handovers and contribute to more efficient network management.

4.1.10 Multi-interface management and integration of mobility protocols

The IP layer constitutes a reasonable level to offer inter-technology mobility support, being the most widespread network layer protocol and in use with different access technologies underneath. IP mobility has been a research topic for a long time and the different solutions designed to allow a user to freely roam across different points of attachment are a clear example of the evolution of the research on this topic, continuously adapting to the new requirements imposed by operators. Although there are hundreds of different solutions, none of them has been a clear market success and none is massively deployed. The current panorama on mobility management is somehow mixed, since mobility solutions have only been deployed within the cellular operator boundaries, e.g., a user can freely roam across the different access networks defined by 3GPP, but there is no solution for inter technology handover or IP mobility within non-3GPP technologies in the wide sense[16], due to the lack of support in the network and in the terminal. The lack of a common IP mobility solution implemented in the majority of smartphones and networks, results in the inter-dependence between the mobile user experience and the smartphone mobile services exposed to the applications.

The network connectivity management in multi-interfaced devices can mainly follow two models, widely known as *weak host* and *strong host* models. The weak host model will accept any packet destined to one of its IP addresses, regardless of the interface where the packet is received. On the contrary, the strong host model will only accept the packet if the destination address matches that one of the interface which received it. Different operating systems decide to implement one or the other. For instance, Linux implements the weak host model, whereas Windows Vista and Windows 7 default to the strong host, although weak host model behavior is configurable. Such implementation decisions affect the performance of the devices, especially when different access technologies are available. We argue that a flow mobility solution [45, 46] may enhance the user experience when a handover takes place with an ongoing communication. Current smartphones, which have multiple interfaces and can connect to different access technologies, need a connection manager that enables this feature. For Internet access, the two main technologies currently used are cellular (UMTS, LTE) and IEEE 802.11. Nevertheless, the most common approach is to use only one of the interfaces at a time, missing the benefits that the usage of both interfaces simultaneously could provide. One of these benefits, currently being discussed by standardization bodies is 3G offloading. By offloading some of the flows at the mobile terminal over a congested cellular network to a WiFi network whenever it is possible not only is convenient for the user, who enjoys greater bandwidth with less delay and at a lower cost, but also for the network operator that frees resources to serve other users. The offloading can be selective depending on the application running or respond to user requirements.

From our experiments, the only OSes that make possible the simultaneous usage of 3G and WLAN interfaces are Windows Phone 8 and the latest version of Android (Lollipop). In this way,

[16]Some technologies have their own mobility support at link layer but the connections at the terminal will not be able to survive an IP address change.

e.g., a Skype call can continue without interruption even though the mobile terminal attaches to a WLAN that just became available. The attachment to the new network is totally seamless for the ongoing call as it keeps going through the cellular interface, but the applications that can tolerate a hand off, or that start after it, will be bound to the WLAN interface. This, as we have reported in Section 4.1.6.3 improve considerably the performance in case of an inter-technology handover, which is seamless for the user as the ongoing communication is kept at the cellular interface while the applications started from that moment on are connected through the WLAN. However, once the ongoing session has finished, the new connections will use the WiFi interface mandatorily, without a choice, for instance, if the user needs to establish a longer session and they will be on the move or if they need to ensure session continuity. According to the IETF [47] there are different approaches for connectivity management in multi-interface devices: *i)* per-application, *ii)* centralized, system-wide or based on user input *iii)* stack-level solutions to specific problems. If flow mobility were enabled, applications could choose their default interface, mobility requirements if more than one access technology is available or specify minimum resources or QoS needs according to the network interface. Both hardware and software tools in current smartphones allow the implementation of these kind of policies, although it increases complexity both in the development of the applications and the connectivity management. In addition, the mobile OS has to deal with multiple applications running in parallel, which may or may not specify network connectivity requirements in the same way. Therefore, the system needs to provide a combined approach, with a default policy, system-wide, based on information available and at the same time offer the possibility of a more advanced connectivity management per-application, if it is specified. Moreover, the management of simultaneous connectivity opens issues as routing, default address selection [48] and the selection of parameters to be configured on a per-interface basis [47].

4.1.11 Conclusions

In this article we have studied a feature that commonly goes unnoticed, not existing much documentation about its operation: the network manager in smartphones. To that purpose, we have analyzed the connection procedure of the three most popular mobile OS families – Android, iOS and Windows Phone 8 – and we have studied the performance when a handover, both inter- and intra-technology takes place. We have also examined how this procedure in case of handover impacts the performance of some applications and their effects in user experience. Finally, we introduced some optimizations that are directly extracted from the conclusions gathered as a result of our experiments. The potential optimizations that we propose are the base for our future lines of research. The mobile terminal that presented the least network attachment delay and the most advanced features in terms of connectivity management is the Windows Phone 8 terminal. WP8 and Android Lollipop allow both the cellular and WiFi interfaces to be active at the same time. Unfortunately, WP8 is also the one that offers the most restricted access to the device's features and the least flexibility for potential modifications resulting from our research. Indeed, in terms of flexibility and room for modifications, the mobile terminal chosen for performing further improvements is the Android device. The open source licensing and the root access provide a convenient development environment to continue improving the connectivity management in current mobile devices. As we have confirmed in this article, the three OS families analyzed access the network in a very similar way and performs also similarly when the point of attachment to the access network changes, so the results of the experimentation with the Android phone would be applicable to other platforms, by adapting it to their corresponding software stack.

4.2 Experimental Evaluation of an SDN-based Distributed Mobility Management Solution

4.2.1 Introduction and Motivation

Recently, mobile users demand on data traffic is increasing dramatically. Operators statistics show that the usage of mobile data traffic has doubled during the last year, and it is estimated that, in three years, mobile data traffic will incur a 13-fold increase with respect to the traffic level reached in 2012 [49]. Mobile and IEEE 802.11 devices are expected to account for 55% of such traffic. In view of the expected growth in users and traffic, wireless network operators are moving towards a dense deployment of cellular base-stations and 802.11 access points, so to be able to cope with the demand generated by 7 billion people and 7 trillion objects under very heterogeneous and mobile conditions, as stated by the 5G PPP.[17]

In this framework, mobile multimedia is the next killer application, and quality of experience represents the primary metric for network planning and design. Therefore, there is an urgent need for innovation in wireless networking solutions to support mobility at unprecedented levels. This includes the creation of a reliable and energy-efficient wireless mobile internet architecture, which is one of the goals of the ICT CROWD project[18]. Specifically, to achieve network-wide reliability and energy-efficiency in very dense and heterogeneous deployments, mobility mechanisms have to be distributed, flexible, and scalable. In fact, mobility mechanisms cannot be centralized, otherwise reliability would suffer the well known *single point of failure* issue; they cannot be rigid, otherwise they would not allow for time- and traffic-dependent energy optimization; and they cannot be unscalable, otherwise architecture and mobile mechanisms would not be of any practical use in dense networks of realistic size. Therefore, the solution for mobility management in future dense and heterogenous networks requires distributed monitoring and control mechanisms, to implement dynamic network optimizations in a scalable manner. Moreover, to reduce costs, a novel architecture should be designed to introduce changes incrementally, possibly in a way that is transparent to customers.

The Distributed Mobility Management (DMM) paradigm—currently under investigation in IETF [50]—presents most of the aforementioned characteristics, and specifically accounts for distributed IP flow mobility managed via distributed anchoring. However, DMM does not manage layer-2 mechanisms and lacks control plane mechanisms for updating forwarding rules dynamically in the various network elements in the backhaul and in access networks, e.g., network switches and wireless points of access. In CROWD, we propose an architecture that exploits the DMM paradigm in a Software-Defined Networking (SDN)-based control framework [51], so that the resulting architecture presents all the tools needed to support mobility in a reliable and energy-efficient manner at both protocol layers 2 and 3. Indeed, while layer-3 (IP) mobility mechanisms are convenient for macro-mobility (at a *regional* scope), we show that SDN-enabled layer-2 mobility mechanisms yield quite faster and reliable solutions in the micro-mobility case, i.e., when the mobility of the terminal is geographically limited to a *district*. Here, the district represents an isolated single-technology domain consisting in a dense set of neighboring cells, while a *region* is the composition of districts, each of which does not necessarily adopt the same wireless technology. In line with the distinction between districts and regions, the proposed architecture presents a two-tier control structure, with local and regional controllers responsible for districts and regions, respectively. Note that, interestingly, both DMM and SDN paradigms reuse existing data-plane protocols and do not affect the legacy operation of wireless access and backhaul technologies, so that they can be suitably used to define a roadmap

[17] The 5G-Infrastructure-PPP is a joint initiative between the European ICT industry and the European Commission. For more details consult the official webpage at `http://5g-ppp.eu/`

[18] ICT CROWD is an ongoing project funded by the European Commission in the Seventh Framework Programme. For more details on the project, consult the official webpage at `http://www.ict-crowd.eu`

towards a fast but smooth network evolution, which would be technologically transparent to the customers.

In this document we present and experimentally validate the distributed mobility management solution designed in the frame of the CROWD project. In particular, we propose a network-based mobility solution, which benefits of flexibility and scalability properties of DMM and SDN. Our solution is transparent to the mobile terminal and provides a two-fold mobility support: first, our solution supports fast layer-2 mobility within a *district*; second, our proposal includes a DMM-based SDN mobility solution designed for the inter-district mobility (layer-3 mobility). We implement our mobility management mechanism in IEEE 802.11 networks, and demonstrate the very promising features of our solution—in terms of mobility support, data path re-configuration, and transparency to the mobile devices—using the celebrated OpenFlow protocol [8] to control access points and network switches. However, our approach is designed so that it can encompass other technologies (e.g., LTE) and heterogeneous access networks.

The rest of the document is organized as follows. Section 4.2.2 introduces the main technologies in our architecture, which is presented in Section 4.2.3. The experimental results are presented in Section 4.2.4. The related work found in the literature is discussed in Section 4.2.5 and Section 4.2.6 concludes the document.

4.2.2 Background

In this section we describe the main technologies involved in our mobility management solution, namely SDN (with OpenFlow) and DMM.

4.2.2.1 SDN and OpenFlow

Software Defined Networking (SDN) has implied a paradigm shift in the network management. The concept of SDN refers to the dynamic reconfiguration of the functions of an operational network, without changes in hardware and basic software of its elements. SDN decouples the data and control planes in the network, relying on an external controller that manages the behavior of the network elements. As it is often linked to OpenFlow, some times these two terms are used as synonyms, but actually, they are not exchangeable. While SDN refers to the concept of decoupling the data and control planes in the switching elements of the network, OpenFlow is the mechanism that enables the SDN operation by standardizing the communication between switches and controllers, which is commonly known in the field as the *southbound* interface. OpenFlow, standardized by the Open Networking Foundation[19] (ONF), has become the protocol of reference and it has been integrated into a number of SDN frameworks with wider scope and objectives, including Network Function Virtualization (NFV) (e.g., OpenNaaS[20], Project Floodlight[21], OpenDaylight[22]). OpenFlow, currently in its version 1.4.0, can be added to switches, wireless access points and routers and some vendors already commercialize OpenFlow-enabled switches. Each OpenFlow switch maintains at least one flow table with entries formed by a set of fields to match and an action to be performed when there is match. When a data packet does not match any entry, the switch notifies the controller, which decides how to handle that packet. The flexibility of OpenFlow has led to the implementation of numerous controllers—e.g., NOX, POX, Ryu, FloodLight—to interact with OpenFlow switches, developed in a wide range of programming languages and supported in many platforms.

[19] `https://www.opennetworking.org/index.php`
[20] `http://www.opennaas.org/`
[21] `http://www.projectfloodlight.org/`
[22] `http://www.opendaylight.org/`

4.2.2.2 Distributed Mobility Management

Traditionally, IP mobility management relies on a centralized mobility anchor, such as the Home Agent (HA) in MIPv6 or the LMA in PMIPv6, being that central entity the one that manages all the bindings. However, such a centralized architecture may encounter scalability and performance issues as the number of mobile nodes and the volume of data traffic increases. In order to define a distributed mobility management mechanism that can adapt the rigid previously existing solutions, IETF chartered the DMM working group in 2012 [50]. The major difference with respect to traditional IP mobility management is that DMM distributes the mobility anchoring at the edge of the access network, effectively flattening the network by removing hierarchies. When the mobile node changes its point of attachment to the network, the ongoing sessions keep anchored to the previous anchor while the new sessions will be managed by the anchor in the target network. Data traffic is tunneled between both anchors and forwarded to the mobile node, which can deregister from the previous anchor once the ongoing sessions are terminated. Assuming that most of the sessions are relatively short, most of the data traffic is routed optimally without tunneling. DMM extends and adapts already existing IP mobility protocols to facilitate the networking architecture migration. In our solution, we consider a PMIPv6 approach together with SDN concepts. In this way, the SDN controller manages the binding of a mobile node that attaches to the network and handles its mobility by coordinating with the controller in the target domain. This approach enables the controller to modify the data paths so the data flows are delivered to the actual location of the terminal, without the need of going through a central node, such as the HA or the LMA.

4.2.3 Architecture

In this section we provide the detailed description of our architecture for distributed mobility management. We propose a network-based mobility solution, which inherits the flexibility and scalability typical of SDN, and which is transparent to the mobile terminal. Our mobility management architecture relies on two main entities, the CROWD Local Controller (CLC) and the CROWD Regional Controller (CRC), and provides mobility at two differentiated levels, which we present next. In particular, we present first the mechanisms enabling fast layer-2 mobility within a domain (the so called *district*); afterward, we present the SDN-based DMM solution designed for the inter-district (regional), layer-3 mobility.

We define a district as an isolated single-technology domain, similar to the localized mobility domains of Proxy Mobile IPv6. A district is composed of a dense deployment of APs connected together by a switched network of SDN-enabled interconnection nodes (OpenFlow switches). A district includes, at least, one gateway (DMM-GW) that connects the district to the Internet and plays a central role in the inter-district mobility solution. In Fig. 4.4 we present an example of the complete architecture with two districts and the flow diagram for the initial connection of a mobile node (MN) to the network. In this case, the districts are composed of IEEE 802.11 Access Points, an OpenFlow-capable backhaul connecting the APs to the DMM-GW and a local controller (CLC). The regional controller (CRC) is located at the operator core network and coordinates the attachment of a mobile terminal to a district as well as the inter-district mobility.

Upon attachment of the terminal ($MN1$ in Fig. 4.4), standard APs (bridged to a wired network) generate a Logical Link Control (LLC) message that serves as the mechanism to update the forwarding tables in the switched network. As all the network is OpenFlow capable, this LLC message is encapsulated in an OpenFlow message and sent to CLC_1. The LLC message contains the MAC address of the terminal and the MAC address of the AP. In the case depicted in Fig. 4.4, CLC_1 does not have any previous entry of the terminal on its Binding Cache (BC)[23], hence it has not been previously attached to its controlled district. In order to check if the node is already registered

[23]Inherited from PMIPv6, this table stores bindings between terminals and points of attachment in our architecture.

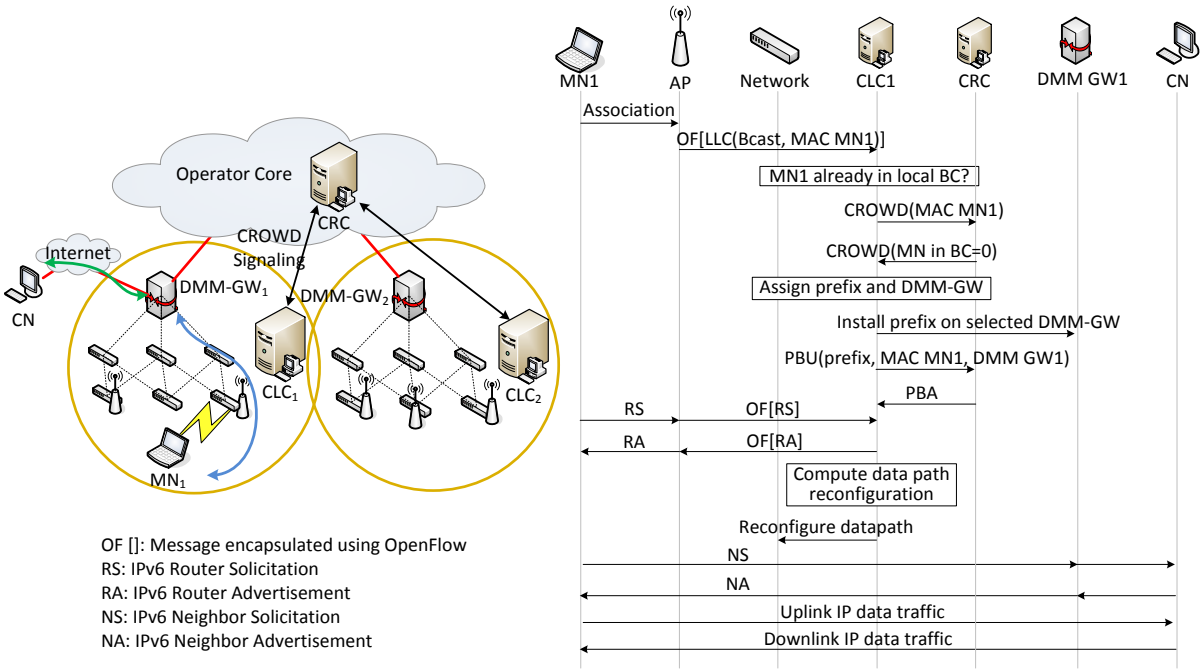

Figure 4.4: Initial attachment.

in a previous district, CLC_1 will contact the CRC. In this first example, the terminal has not been attached previously to any CROWD network, so CLC_1 assigns an IPv6 prefix and a DMM-GW to the terminal, stores this information on its BC and notifies the CRC about the assigned prefix, DMM-GW and MN identifier (e.g., MAC address). In this way, the CRC is able to keep track of every MN attachment. After successfully attaching to a new network, the standard procedure for the terminal is to send a Router Solicitation (RS) to configure its IP address using IPv6 SLAAC. As in the case of the LLC message, the network encapsulates the RS message and sends it to CLC_1. The CLC answers the RS with a Router Advertisement (RA) message, providing the prefix and default router $DMM\text{-}GW_1$ selected before. Hence, through the mediation of the CLC, hijacking the RA functionality of the network, we are able to control the IP level attachment of the terminal within the CROWD network. Note that, although we assume the MN sends a RS message every time it attaches to a new point of attachment, in our solution we can trigger the RA message even if the RS is not sent, which is a very useful feature in case the MN does not completely follow the standard (most Linux boxes do not send RS messages unless the interface goes down). At this point in time, CLC_1 is able to compute the required matching rules and data path modifications to forward the terminal's packets to $DMM\text{-}GW_1$. These modifications are configured into the network through the OpenFlow protocol, requiring several message exchanges among CLC_1 and the different switches conforming the path between the terminal and $DMM\text{-}GW_1$. Once the data path is configured, packets originated at the terminal with layer-2 destination the DMM-GW are transparently forwarded at layer-2. For the layer-3 stack of the terminal the path to the DMM-GW is a single hop. Finally, after performing a Neighbor Discovery procedure, the terminal is able to exchange packets with any CN through $DMM\text{-}GW_1$.

4.2.3.1 Intra-district Mobility

The intra-district handover is illustrated in Fig. 4.5, where the mobile terminal $MN1$ attaches to a second AP within the same district. In this case the only required change is to modify the

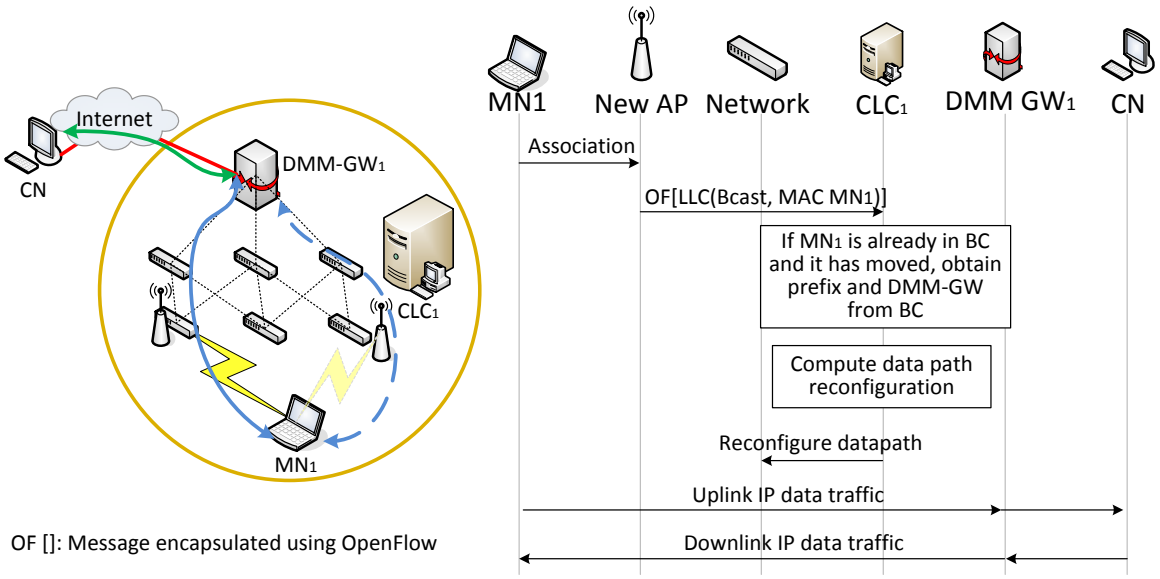

Figure 4.5: Intra-district handover.

data path, so packets are forwarded between the new AP and $DMM\text{-}GW_1$. The flow diagram depicted in Fig. 4.5 represents the procedure. The new AP, upon attachment of $MN1$, generates an LLC message. Upon reception of this message, CLC_1 is able to notice the movement of the terminal looking up in its BC the MAC address of the AP the terminal was connected to.[24] With this information, the CLC is able to compute the required modifications to the data path for the terminal's packets. Accordingly, the OpenFlow configuration in the district is modified, and the terminal's packets will flow from the new AP to the old DMM-GW, i.e., $DMM\text{-}GW_1$ (we assess the re-configuration and handover-related delays in Section 4.2.4).

The advantages of this approach over traditional layer-2 mobility mechanisms are many-fold:

- Through the control of the prefix delegation by the controller, the MN can be attached to any of the DMM-GWs in the district, or even to several of them at the same time. This allows a new degree of freedom, since the network can balance the load at the different exchange points to the Internet.
- The control of the data path, operated by the controller, enables the network to provide fast mobility of the terminal. The data path reconfiguration scales better with the number of switches than standard spanning tree approaches, whose reaction time can be measured in seconds for large networks [52].
- The algorithm used for controlling the data path can be arbitrarily complicated, allowing complex traffic engineering operations. The data path forwarding decision engine can consider load balancing metrics, or even complex mechanisms minimizing the number of nodes requiring changes in their forwarding tables. In addition, we could employ different forwarding decision engines for different nodes or traffic classes, prioritizing specific metrics, such as delay or available bandwidth along the path (leveraging the OpenFlow monitoring capabilities).
- Standard mobile nodes, such as Linux boxes, do not send Router Solicitation (RS) messages unless the interface goes down, hence they do not typically do it when performing a handover.

[24] Note that using the MAC address represents just a practical example for terminal identification, although any other kind of terminal identifier could be used instead.

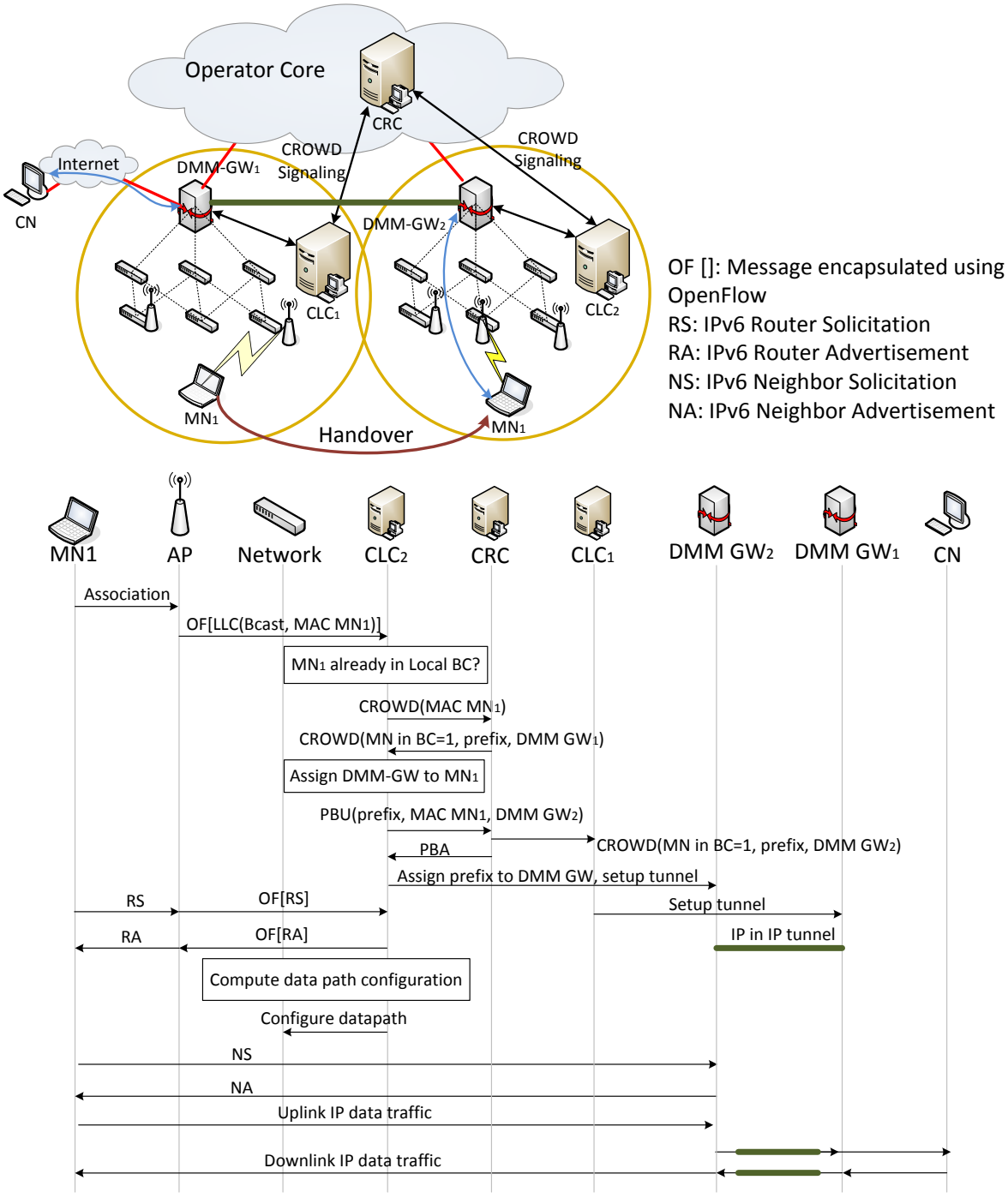

Figure 4.6: Inter-district handover.

This forces developers to add the transmission of this RS message or specific attachment-detection traps to the access points. This is the case of PMIPv6, for example. By leveraging the SDN approach, our architecture does not require the transmission of the RS message, hence making the deployment easier than standard PMIPv6 approaches.

4.2.3.2 Inter-district Mobility

Fig. 4.6 presents the procedure of a handover between two districts. This procedure relies on the communication among CLCs being orchestrated by the CRC. Basically, the CRC behaves as a data-base containing the list of DMM-GWs to be considered while performing a handover. The configuration of the IP layer on the DMM-GWs and the tunnel setup between them is handled locally by the CLCs in each district. The procedure starts when the MN attaches to an access point in a different district. As in the previous cases, this event triggers the CLC (CLC_2) to check if the node is registered on its internal BC. As this is the first time the terminal attaches to this district, CLC_2 asks the CRC for previous registrations. In this case, the CRC has information regarding the terminal, which is transmitted to the CLC_2 that choses the DMM-GW to be used within this district ($DMM\text{-}GW_2$) and returns this information to the CRC. The CRC stores this information on its local BC for future reference. At this point in time several procedures are performed in parallel. First, the CRC informs CLC_1 of the new location of $MN1$. With this information, CLC_1 configures $DMM\text{-}GW_1$ with an IP-in-IP tunnel connection to $DMM\text{-}GW_2$ and changes the routes at $DMM\text{-}GW_1$ so that the prefix used by the terminal is routed through the tunnel. In parallel, CLC_2 configures the new prefix in $DMM\text{-}GW_2$ and sets up the IP-in-IP tunnel towards $DMM\text{-}GW_1$. Once the tunnel is established, the configuration of the data path in the new network is performed as in previous cases. When all the procedure is complete, packets from the core network to $MN1$ are forwarded first to $DMM\text{-}GW_1$, which tunnels them to $DMM\text{-}GW_2$. After $DMM\text{-}GW_2$, the OpenFlow configured data path takes care of forwarding the packets to the appropriate location of $MN1$ within the district.

The key advantage of this procedure over non-SDN DMM approaches is the flexibility on selecting the most appropriate DMM-GW to be used for each terminal. Since the CLC is able to decide and attach the prefix used by the terminal to any of the DMM-GWs available at the district, with our approach, the target network is able to implement any load balancing or complex algorithm to decide the best DMM-GW to use. In addition, the data path forwarding within the district is decoupled from the layer-3 mobility. In order to minimize the security implications of this approach, our design enforces that the only entity talking with the DMM-GWs belonging to a certain district is the local controller of that district. In such scenario it is possible to think of specific security associations between the CLC and DMM-GWs of the district. Finally, although we have not implemented or included this functionality in the current design, it is worth highlighting that the same design can be used to provide IPv4 DMM capabilities to the network by using the Address Resolution Protocol (ARP) instead of IPv6 Neighbor Discovery protocol.

4.2.4 Implementation and Experimental evaluation

From Fig. 4.4, where we present the architecture of our solution, we can identify the main architectural elements, splitting our testbed into three kinds of entities: *i)* the regional and local **SDN controllers** manage the forwarding rules in the OpenFlow-enabled backhaul and handle the mobility of the MN; *ii)* the **OpenFlow switches**, which include, due to the fact that the wireless interfaces in the APs are controlled as in normal OpenFlow switches, both wired backhaul and wireless access network elements; *iii)* the **DMM Gateway**, which is the only entity not related to OpenFlow and acts as the gateway in the district. Table 4.8 gathers the main characteristics of these elements. Specifically, in our testbed we have two 802.11-OpenFlow enabled APs, an OpenFlow switch, and a DMM-GW in each district. Additionally, each district is managed by its own controller (CLC), and CLCs are coordinated by a CRC.

The OpenFlow signaling is distributed between the switches and the controllers in an outband connection. The wireless access is part of the OpenFlow-enabled network, so the connection between MN and DMM-GW is a single-hop connection at network layer (IP). Since the solution works at

Table 4.8: Main HW and SW characteristics of the testbed

Entity	Device (OS, kernel version)	OF-enabled
Switched backhaul	Linksys WRT54GL (OpenWrt Backfire, 2.6.32)	✓
IEEE 802.11 APs	Alix 2d3 (Ubuntu 12.04, 3.2.0)	✓
DMM-GW		X
Regional and local controllers	Desktop (Debian 7.4, 3.2.0)	Ryu controller

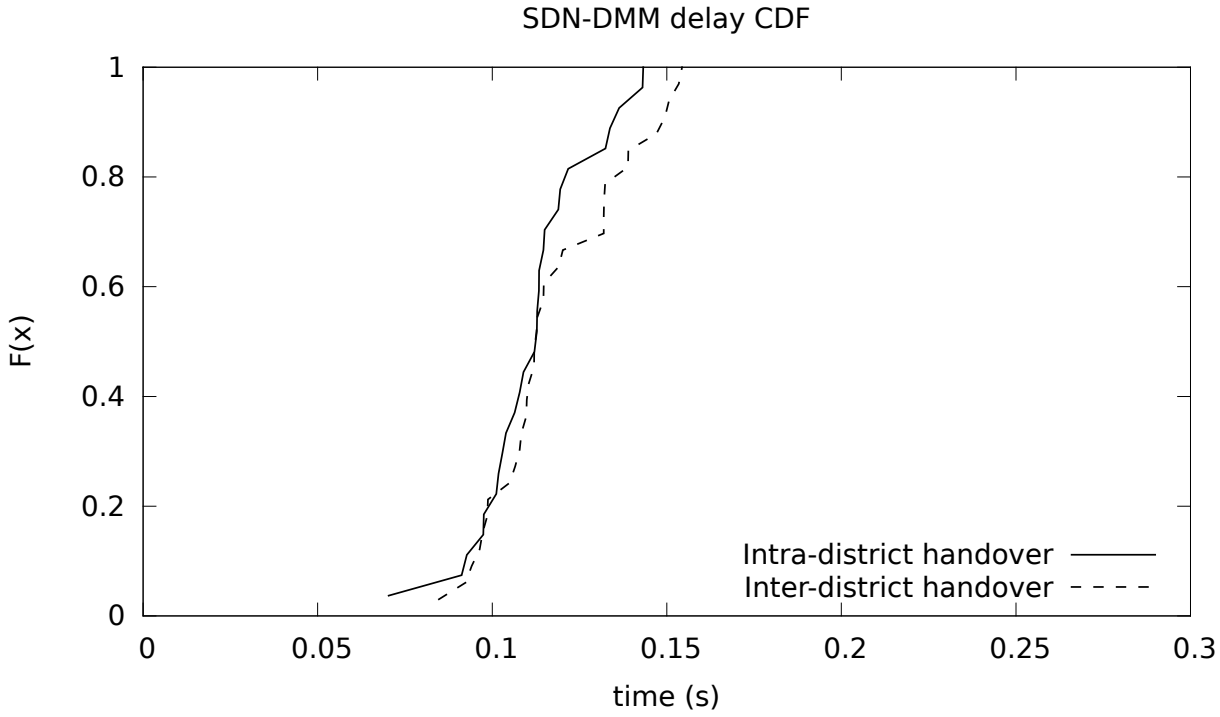

Figure 4.7: CDF of the different handover delays

layer-3, it is necessary to define new APIs to control the IP layer configuration of the terminal's session anchor point (DMM-GW). Specifically, we need two new APIs: one to convey information on the mobile node to the CRC, such as the IPv6 prefix and DMM-GW assigned to it, and a second one to configure the IP layer of the DMM-GW, the prefixes reachable through the interfaces and to setup an IP-in-IP tunnel. We have implemented our local and regional controllers using the Ryu framework.[25] Ryu provides a comprehensive API that eases the network management. In particular, our CRC has been developed in 150 lines of code and our CLC is coded in about 1000 lines.

To characterize the performance of our prototype we run 30 repetitions of every experiment. Fig. 4.7 shows the CDF of the delay due to the intra- and inter-district handovers, measured as the interruption in data traffic, which also includes the link layer association. The intra-district case shows a lower latency, because the MN is already known in the district and therefore a simple change in the point of attachment to the network in needed in that case. The inter-district handover implies changes at layer-3 level, including the configuration of the IP-in-IP tunnel between the two DMM-GWs involved. However, the delay is very similar to the intra-district handover, because the controller runs the configuration of the gateways and the establishment of the OpenFlow datapath

[25] `http://osrg.github.io/ryu/index.html`

in parallel.

Following with the evaluation of the performance of our SDN solution, we measure the throughput we can achieve for TCP traffic, using the *iperf* traffic generator. We compare the case the MN is attached to one of the districts and the case that it has performed a handover and its traffic is tunneled between the current and former DMM-GWs. The average throughput is 8.6 *Mb/s* and 8.1 *Mb/s*, respectively, confirming that the impact of this tunneling is very low (the IP-in-IP tunnel just adds 40 bytes of headers). It is noticeable that the maximum throughput available in the SDN framework is quite limited for the moment. This is a well-known open issue and very heavily influenced by the use of the Linksys WRT54GL router as OpenFlow-switches.

It is worth to mention that the signaling due to OpenFlow does not impact the data traffic performance, as we use out-of-band signaling within the SDN elements, and the data transmission from the MN is only delayed at the first packet, when the matching rules are firstly installed in the OpenFlow switches. To evaluate the impact of the SDN operations, we have measured the delay introduced by the operations carried out by the controlled required for our SDN-based DMM solution. From our experience, the choice of the device running the controller has a significant impact on the performance of the prototype, especially in the case of handover.

4.2.5 Related Work

SDN has raised in popularity very recently, and it has attracted the attention of the research community, as it eases testing protocols and networking algorithms. The flexibility and programmability of SDN architectures has contributed to the proliferation of several large deployments designed by leading research institutions. Among the first deployments we find B4 [53], which is Google's SDN-based wide area network (WAN) to interconnect its data centers around the world. For B4, an extensive analysis is available on how to manage the routing and traffic engineering through OpenFlow and the designers of B4 provide interesting insights on design, performance, scalability and failure resilience of their solution. Although providing mobility is not within the objectives of B4, we have leveraged the knowledge provided by the B4 large scale implementation example to design our solution.

Despite the usage of a wired structure to transfer the OpenFlow signaling in our network, we focus on the wireless access and the mobility management. In that regard, one of the most remarkable SDN deployments applied to wireless networking is *OpenRoads* [54] (also known as *OpenFlow Wireless*), open to researchers for running their algorithms concurrently by means of virtualization at Stanford University. OpenRoads incorporates different wireless technologies, namely WiFi and WiMAX, and one of its early proofs of concept was based on providing mobility across multiple technologies [55]. In addition, the performance of OpenRoad has been demonstrated by means of an n-casting transmission solution [56]. All these approaches are based on the same principle as our layer-2 mobility approach, reconfiguring the data-path, although they do not consider IP mobility or the design of a scalable architecture as we do. All the tools used by OpenRoads are open source, so as to make the infrastructure reproducible by other research groups in their own networks. Likewise, our implementation shows the flexibility of current SDN software tools available as open source and is built upon commercial-off-the-shelf devices. At the moment we only focus on IEEE 802.11 access points, but we are planning to extend our testbed to include heterogeneous access technologies in the short term.

A full software-defined mobile network (SDMN) is defined in MobileFlow [57], and its authors provide a comparison to the current Evolved Packet Core (EPC) architecture. A prototype implementation is also proposed in [57], based on the concept of the MobileFlow Forwarding Engine (MFFE), which encompasses all the user plane protocols and functions, and the MobileFlow Controller (MFC), which is a logically centralized entity that configures dynamically the MFFEs (i.e., the data plane). Although MobileFlow is OpenFlow-based, MFFEs must also support operations

that are not carried out at the switch-level, as layer-3 tunneling, for instance. Mobility management can be supported as the controller can update forwarding rules according to the tunnel encapsulation or decapsulation requirements. This approach is also followed in our implementation, where we can set the tunneling and forwarding rules above link layer and the controller updates the forwarding rules in the OpenFlow domain. A different approach presented in [58] proposes to move the EPC to the cloud by means of virtualization and implementing GTP extensions for OpenFlow for mobility management. The mobility solutions proposed by both works are based on the same mobility concepts currently used in cellular networks, hence inheriting the scalability issues commented in Section 4.2.2.2.

In order to mitigate the impact of the re-association to the new AP during a handover, Odin [59] proposes an SDN framework for enterprise WLANs that leverages light virtual access points (LVAPs) as an abstraction to the association to the wireless AP. In this way, each client appears to be connected to its own AP, so the physical AP hosts an LVAP for each attached client (note that an LVAP is not a virtual interface). When there is a need for handover, the Odin Master migrates the corresponding LVAP to the new physical AP. This approach differs highly from our work, since in the first place it requires modifications at the APs that are burdened with the need of running virtual APs. In addition, in our prototype, we have focused on improving performance of the SDN configuration and layer-3 mobility, as our measurements proved those were the dominant factors in the attachment and handover delays.

An architecture to support flow-based routing in wireless mesh networks by means of OpenFlow is proposed in [60], which has been evaluated with a mobility management implementation that focuses on network-initiated handovers triggered by IEEE 802.21 MIH Events. This implementation includes an OpenFlow controller and a Monitoring and Controlling Server (MCS) that decides to which Mesh Access Points (MAP) the mobile node should connect and update the forwarding rules. During the handover, the controller configures temporary routes that forward the traffic to the two MAPs involved and will be removed when the handover is complete. The minimum outage due to handover that they achieve is on average 200 ms and their results also confirm that the main contribution to this delay is due to the association to the new MAP. A similar approach is followed in [61], but in this case with in-band signaling and relying on a centralized controller for data rules and a local distributed controller that takes care of the control rules. In our architecture, we also propose two hierarchy levels for the controller deployment, namely the local controller (CLC) and the regional controller (CRC) with the aim of managing the mobility between different domains. However, we go beyond and provide mechanisms for IP mobility continuity while roaming to different domains.

4.2.6 Conclusion

In this document we have presented an SDN-based architecture for supporting distributed mobility management in dense wireless networks. Our architecture presents the flexibility and scalability provided by both SDN and DMM approaches. Although it is an early-stage prototype, our implementation shows very promising features in terms of mobility support, load balancing, data path re-configuration and transparency to the user, as no modification of terminals is required.

4.3 Generic IEEE 802 Network Reference Model Proposal

This appendix corresponds to a copy of the document omniran-14-0068-02-CF00, which was presented in November 2014 IEEE 802.1cf meeting. This document is a working on progress document which will be integrated in the main recommended practices document generated in this task group.

Generic IEEE 802 Network Reference Model Proposal			
Date: [2014-11-03			
Authors:			
Name	Affiliation	Phone	Email
Antonio de la Oliva	UC3M	+34657079687	aoliva@it.uc3m.es
Juan Carlos Zuniga	InterDigital		
Luis Miguel Contreras	Telefonica		
Roger Marks	EthAirNet Associates		
Notice: This document does not represent the agreed view of the OmniRAN TG It represents only the views of the participants listed in the 'Authors:' field above. It is offered as a basis for discussion. It is not binding on the contributor, who reserve the right to add, amend or withdraw material contained herein.			
Copyright policy: The contributor is familiar with the IEEE-SA Copyright Policy <http://standards.ieee.org/IPR/copyrightpolicy.html>.			
Patent policy: The contributor is familiar with the IEEE-SA Patent Policy and Procedures: <http://standards.ieee.org/guides/bylaws/sect6-7.html#6> and <http://standards.ieee.org/guides/opman/sect6.html#6.3>.			

Abstract

This contribution addresses a generic reference model for IEEE 802 networks. It is based on the following contributions:

- omniran-14-0051-01-CF00-omniran-network-reference-model-with-backhaul, Roger Marks, 15 Jul 2014
- omniran-13-0018-00-ecsg, OmniRAN Introduction to IEEE802.1, Max Riegel, 18 Mar 2013
- omniran-14-0044-02-0000, SDN Use Cases Summary, Antonio de la Oliva, Juan Carlos Zuniga, Roger Marks, 17 Jul 2013
- omniran-13-0048-04-ecsg, OmniRAN ECSG Results and Outlook, Max Riegel, 25 Jun 2013
- omniran-13-0060-00-ecsg, OmniRAN SDN Use Case for external communication, Max Riegel, 7 Aug 2013
- omniran-13-0067-00-0000, OmniRAN architecture suggestions, Yonggang Fang, 11 Sep 2013
- omniran-14-0030-00-0000, Backhaul in OmniRAN, Max Riegel, 19 Mar 2014
- omniran-14-0038-00-CF00, 802.1CF R3 Considerations, Max Riegel, 14 May 2014

Generic IEEE 802 Network Reference Model Proposal This document considers an evolution of the NRM proposal presented in DCN-68r0, updated with the latest agreements. Annex I presents text explaining the actual view of the TG on the general NRM as future reference.

4.3.1 Agreed reference model in Athens September 2014 Meeting

4.3.1.1 Nomenclature:

AN: Access Network
SS: Subscription Service
CNS: Core Network Service
CIS: Coordination and Information Service
TE: Terminal

4.3.1.2 Core Network Reference Model

In the following we provide an evolving Network Reference Model based on the discussions held in Atenas September 2014 meeting.

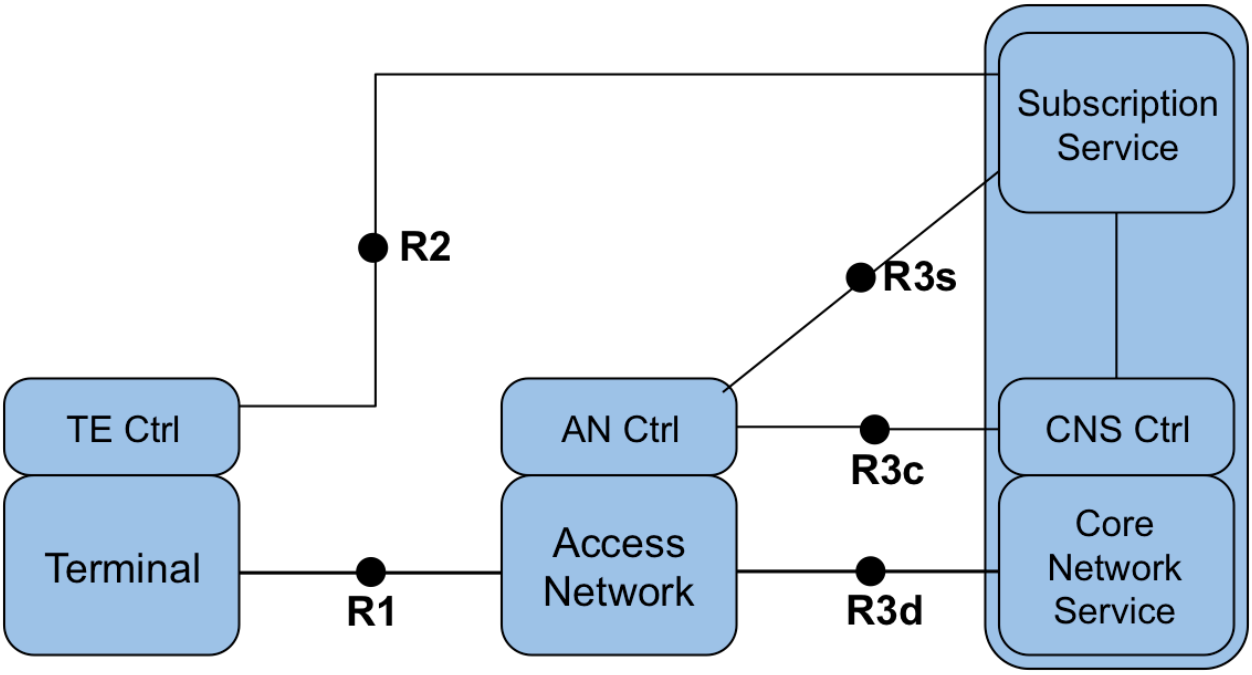

Figure 4.8: Core Network Reference Model

Figure 4.8 presents the Core Network Reference Model. This NRM is the basis of further models and includes the basic differentiation between services and the reference points for their communication. This reference model is composed of 3 main domains; i) the terminal, ii) the Access Network and iii) the Core Network. The basic NRM differentiate 3 services: i) Access Network service, ii) Subscription service and iii) Core Network service. Please note that currently no assumption on the service providers is made.

In the NRM depicted in Figure 4.8, for each domain we assume a control entity which we will call Controller (Ctrl). Each of the domains has a specific Controller.

4.3.1.3 Reference Points

- R1:represents the PHY and MAC layer functions between terminal and base station, which are completely covered by the IEEE 802 specifications
- R2:represents a control interface between terminal and the subscription service, e.g. for authentication

- R3: represents the reference point for the communication between the access network to the core network up to the interface between L2 and L3 in the first L3 router.
- R3d:represents the IEEE 802 data path interface between access network and the first hop router of the Core Network Service
- R3c: represents a control interface between the Access Network and Core Network Controllers
- R3s:represents a control interface between the access network controller and the subscription service, this interface can be used to influence the authentication mechanisms

4.3.2 Network Reference Model including Terminal Controller Reference Point

The following evolution of the NRM includes the communication reference point between the terminal and the Access Network Controller. Some of the functionalities of this reference point will be related to the configuration of the logical interface in the terminal and the control of the actual path followed by data flows in the terminal. In addition, the terminal controller may include some configuration parameters so the access controller can influence the configuration of the terminal.

4.3.2.1 Reference Points

- R8c: represents a control interface between the Access Network Controller and the Terminal Controller.

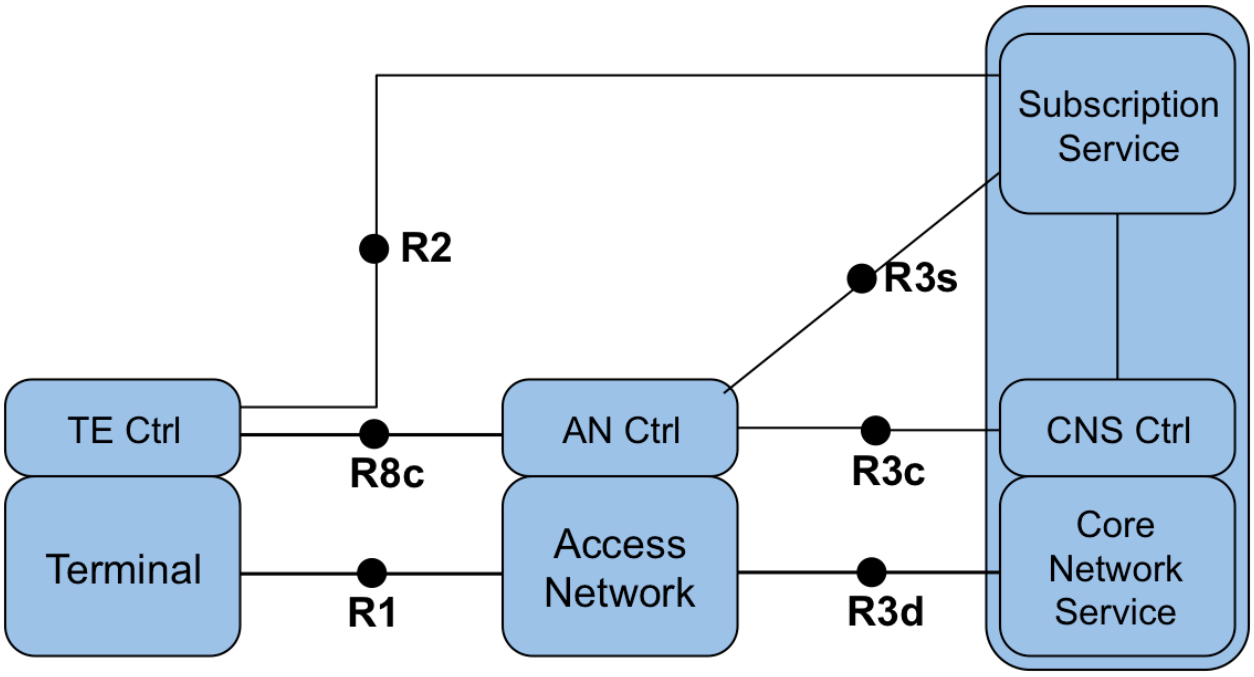

Figure 4.9: NRM with R8c

4.3.3 Network Reference Model including Coordination and Information Service

Some deployments include Coordiantion and Information Services to provide advance services such as spectrum management, coexistence, information services for mobility and so on. This reference model includes the possibility of having CIS entities in the network and provides a reference point to configure and control the information provided by these services.

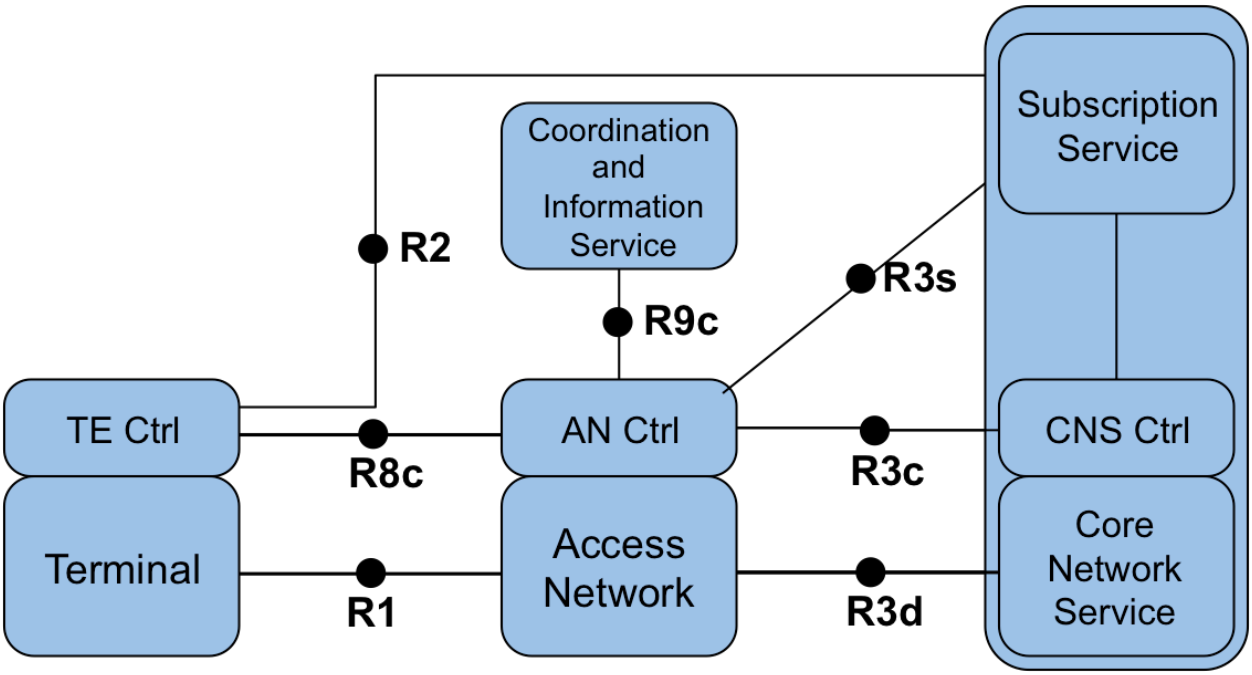

Figure 4.10: NRM including CIS

4.3.3.1 Reference Points

- R9c: represents a control interface between the Access Network Controller and the CIS system. It allows the configuration of the CIS and the control of the information provided by this system to the user.

4.3.4 OmniRAN Network Reference Model (NRM)

4.3.4.1 Complete Reference Point Descriptions

Herein we will describe the Reference Points 6 and 7, which have not been previously discussed.

- R6: Access Point Interface:
  - R6d: Data-only Access Point interface Data-plane interface carrying user data between the access point and the backhaul.
  - R6c: Control interface to Access Point Control-only interface for the configuration of the access point. It includes the configuration of the access point to the backhaul, as well as to the access link.
- R7: Backhaul Interfaces
  - R7c: Control interface to Backhaul This interface is used to control and configure the data-path of the user flows within the backhaul, connecting the access point to the core.

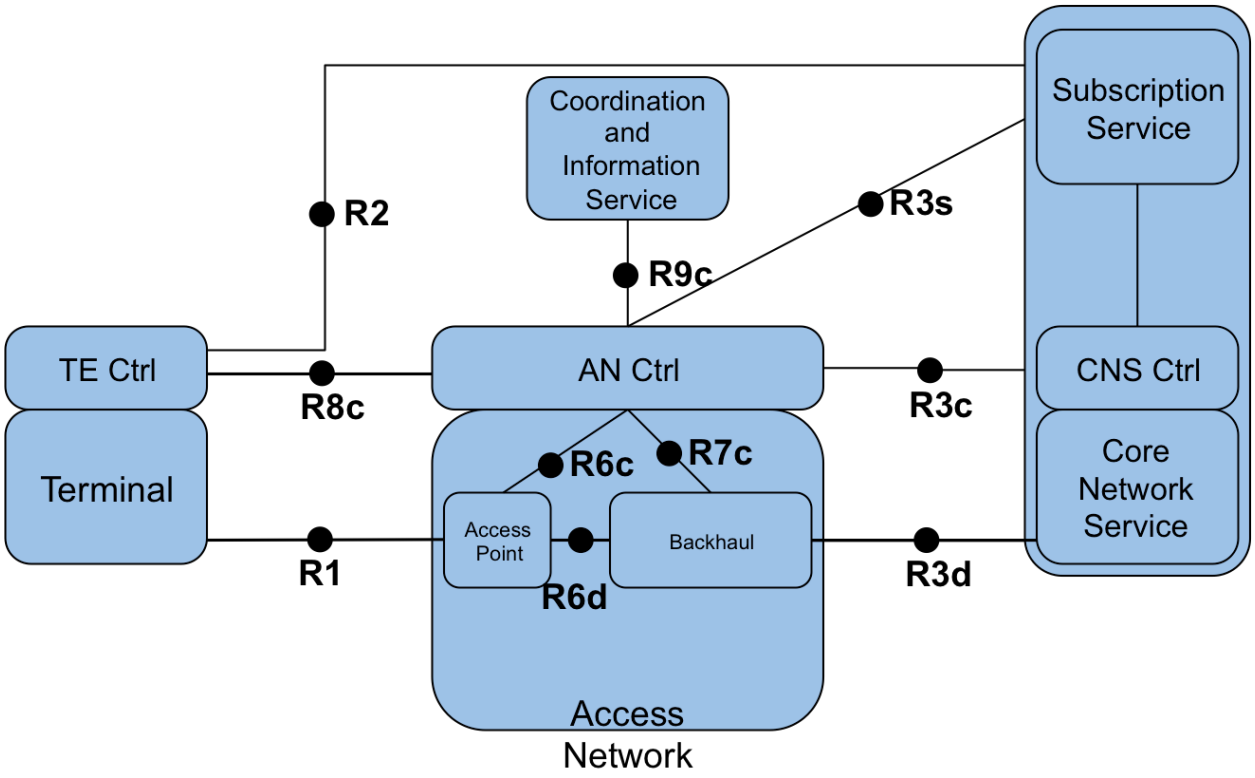

Figure 4.11: Generic IEEE 802 Network Reference Model

4.4 OMNIRAN (IEEE 802.1cf) SDN Functional Decomposition

This appendix corresponds to a copy of the document omniran-15-0018-00-CF00, which was presented in March 2015 IEEE 802.1cf meeting in Berlin. This document is a working on progress document which will be integrated in the main recommended practices document generated in this task group.

SDN Functional Decomposition			
Date: 2015-01-13			
Authors:			
Name	Affiliation	Phone	Email
Antonio de la Oliva	UC3M	+34657079687	aoliva@it.uc3m.es
Juan Carlos Zuniga	InterDigital		
Luis Miguel Contreras	Telefonica		
Notice: This document does not represent the agreed view of the OmniRAN TG It represents only the views of the participants listed in the 'Authors:' field above. It is offered as a basis for discussion. It is not binding on the contributor, who reserve the right to add, amend or withdraw material contained herein.			
Copyright policy: The contributor is familiar with the IEEE-SA Copyright Policy <http://standards.ieee.org/IPR/copyrightpolicy.html>.			
Patent policy: The contributor is familiar with the IEEE-SA Patent Policy and Procedures: <http://standards.ieee.org/guides/bylaws/sect6-7.html#6> and <http://standards.ieee.org/guides/opman/sect6.html#6.3>.			

Abstract

This document proposes initial draft text for the SDN chapter of IEEE 802.1cf

4.4.1 SDN Functional Decomposition

4.4.1.1 Introduction

Software Defined Networks (SDN) is a new paradigm based on the splitting of control and data plane of networking elements. Basically it works by pushing the intelligence related to the operation of a certain service to a central controller, while the data path (user data packets) is handled based on the orders of the central controller in separate and specialized elements. Within the IEEE 802 set of technologies, there are multiple functionalities, which can be designed based on the SDN paradigm. This document presents several use cases showcasing these functionalities:

1. Setup of interfaces and nodes
2. Detection of node attachment
3. Path Establishment
4. Path Tear Down
5. Path Maintenance
6. Path relocation
7. Affecting the behavior of Coordination and Information System
8. Configuration of connection between the Core Network and the Access Network
9. Event handling
10. Statistic gathering

4.4.2 Acronyms

CIS Coordination and Information System
CN Core Network
AN Access Network
SDN Software Defined Networking
VLAN
OAM
MAC
QoS
LLC
SNAP

4.4.3 Roles and identifiers

1. Controller: Application that manages different behaviors (e.g., flow control) in a Software Defined Networking environment.
2. Data path element: Hardware/Software entity in charge of executing the orders from the controller, affecting the path through which data is forwarded.

3. Data path element ID:

4. Controller ID:

4.4.4 Use Cases

Setup of interfaces and nodes Through SDN a central controller can implement a control logic enabling it to configure several parameters in the nodes and interfaces of the data path. Within the possible set of configuration parameters there are three main families:

1. SDN control configuration

2. Short time scale configuration

3. Long time scale configuration

SDN control configuration refers to the required setup of the parameters ruling the communication between the data path element and the controller. These parameters may include IP address of the controller, kind of protocol, VLAN or interface used for the communication and certain timers governing the transmission of keep alive messages or the tear down of it in case of failure. These configuration parameters must also include the different timers, ports and protocols used for the communication between the controller and the data path element.
Short time scale configurations are related with the configuration of parameters, which may change in very short time scales. For example transmission power, MAC QoS parameters, antenna selection, etc..
Long time scale configuration refers to the long-term configuration of the node or interface; the parameters used by the controller are the typical ones an OAM system may use. Examples of these parameters include the operational frequency, configuration regarding credentials or authentication servers to use, supported authentication modes, VLAN configuration, etc.

Detection of terminal attachment A very important operation required to use SDN control in the access network is the possibility of detecting the attachment of new terminals. The user's terminal, typically will not include any kind of SDN software nor they are aware of connecting to a network using an SDN controller. In this way, it is needed some mechanism to handle the detection of the terminals while attaching to the network PoAs. In a PoA where the wireless interface (IEEE 802.11) is bridged to a switch, this detection of terminal attachment can be done thanks to the switch sending a LLC SNAP message upon attachment of a new terminal. In other technologies some other mechanisms should be analysed.

Data Path Establishment An IEEE 802.1cf network does not include IP layer, hence path establishment mechanisms using above layer-2 information are out of the scope of this document. In order to establish a path, a controller requires to be informed of the new flow, including its requirements in terms of capacity, delay and jitter. After receiving this information the controller can compute the best possible path and communicate this decision to the data path elements. After this, the data path elements will enforce the controller decision on the different packets traversing the data path.
In order to perform the data path establishment, the following information/functionality is required:

1. Topology information

2. Mechanisms to compute best path based on some criteria

3. Communication mechanisms to set specific rules in the data path to decide output port/modifications to frame

4. The data path must support some mechanism for packet matching. This mechanism can be arbitrarily complex. It can include simple mechanisms as input port matching, VLAN tag matching or complex rules indicating logical combination of parameters, including internal state of the data path element.

5. The data path must support the application of forwarding rules to the input traffic. In this way the decisions taken by the controller will be applied and the packet will be sent through the appropriate output port.

6. The data path element may support actions over the packets and internal state. The data path element may be able to modify certain parts of the packet and modify internal state variables, such as counters, monitoring variables, etc..

Data Path Teardown A certain path can be created for a specific flow. Once the flow finishes, there is no need to have the path established any longer, freeing resources allocated to the path. In order the controller to tear down a path, it needs to discover that the path can be deactivated. This can be done through monitoring metrics or flow information coming from the flow originator. Once the controller knows that the path can be removed, it can communicate the data path elements its decision. At this moment all data path elements will remove the stored state corresponding to this path.

Data Path Maintenance Typically a data path element will configure forwarding rules with a certain lifetime. The rules must be updated within their lifetime, or the data path element will remove them. Upon expiration of the rule, the data path element should inform the controller about the removal of the rule. In this way, the controller can keep record of the current status of the paths in the network.

Control Path Maintenance In the same way as with data paths, the communication between the controller and the different data path elements must be kept alive through the exchange of some control packets. The actual configuration of the timers to use should be one of the parameters considered in in 1.4.1.

Path relocation Due to several reasons such as traffic engineering, movement of the terminal or QoS degradation, it may be necessary to relocate a data path. With relocation we meant to change the data path elements the data path goes through, while keeping the most similar allocation of resources. This functionality can be divided in a sequence of Data Path establishment and Data Path Tear Down.

Affecting the behavior of Coordination and Information System Terminals and network nodes can relay on Coordination and Information Services (CIS) to gather information helping them to take some decision, such as candidate network selection, channel to use, etc. A controller may interact with the CIS in a standalone way o it may mediate in the communication between the terminal and the CIS. This later approach allows the controller to modify, apply policies, add more information or simply query different servers based on terminal information such as its user profile.

Configuration of connection between the Core Network and the Access Network TBD

Event Handling TBD

Statistic gathering TBD

4.4.5 Functional requirements

The following requirements apply to the SDN procedures.

Support of a control connection between the different data path elements and the controller Elements in the network subject to be controlled or communicate with a controller SHOULD use a secure control connection for the communication. This includes the terminal in case it communicates with the controller.

Support for data path elements with heterogeneous technology interfaces Controllers SHOULD support the configuration of parameters for multiple technologies. Abstract parameters, common for multiple technologies SHOULD be used when possible. The data path element SHOULD provide common controlled behaviors to all the interfaces attached to it, regardless of their technology.

Support of communication mechanisms between the terminal and the controller The terminal SHOULD use a secure communication channel to communicate with the controller.

Support of per packet matching, forwarding rules and actions in the data path element Data path elements SHOULD include mechanisms for packet matching, forwarding rules and actions (packet modifications).

Support of state recording in the data path elements Data path elements SHOULD be able to store operational parameters so they can be retrieved for monitoring.

Support of security associations between the controller and the data path elements (including terminal) TBD

Support of security associations between controllers belonging to the AN and CN, which communicate TBD

Support of security associations between AN controllers and CIS servers TBD

4.4.6 SDN specific attributes

This section lists possible parameters for the different functions involved in the SDN operation.

Abstract parameters

1. Supported Rates
2. TxPower, TxPower levels supported
3. Operational Frequency
4. Statistics: Tx error, Rx error, Number of stations

Terminal Configuration

1. Terminal Controller:
   a) LIST of control capabilities
   b) LIST of interfaces and their capabilities
   c) LIST of protocols to manage interfaces
      i. E.g., interface X supports CAPWAP+OF
2. Interface
   a) Abstract parameters
   b) LIST of parameters to be configured
      i. Technology specific: e.g., BSSID to connect to, RTS Threshold, Short retry, long retry, fragmentation threshold, Tx/Rx MSDU lifetime, enable Block Ack, etc.
      ii. Security parameters (technology dependent)

Access Network Configuration

1. Configuration of interfaces
   a) Abstract parameters
   b) LIST of parameters to be configured
      i. Technology specific: e.g., BSSID, RTS Threshold, Short retry, long retry, fragmentation threshold, Tx/Rx MSDU lifetime, enable Block Ack, etc.
      ii. Technology specific security parameters: WPA/WPA2/WEP, parameters for key management, etc.
   c) Queue configuration: Capacity, max number packets, rate limitation
2. Configuration of nodes
   a) Parameters to configure the connection to controller:
      i. Protocol+port
      ii. Credentials
      iii. Output physical port to use to connect to controller
      iv. ID
3. Configuration of data path ports
   a) VLAN configuration
   b) Number of tables

Data path Establishment

1. Matching rule and actions
   a) Matching rule definition and associated actions

Triggering technology specific features

1. Type for feature
   a) E.g., send 802.11v frame, configure 802.11aa groupcast mode
2. Content of feature
   a) E.g., BSS to attach to, groupcast mode, concealment address, stations to be added

Interacting with CIS

1. Parameters to enable the communication
   a) Protocol to be used, credentials
2. Adding/removing/modifying information at the CIS
   a) CIS specific
      i. E.g., IE elements to add to ANQP

Communication between CN and AN TBD

Event handling TBD

Statistic gathering TBD

4.4.7 SDN basic functions

Controller discovery and configuration is out of the scope of Omniran

Configuration of interfaces Once a data path element or a terminal is attached to a controller, it can configure the different characteristics of the interface. Typical example of this can be taken from the world of IEEE 802.11 where WLAN controllers configure the different parameters of the technology. The typical parameters that the controller will setup are operational frequency and transmission power. Depending on the technology the controller will also be able to configure additional parameters such as RTS threshold or ESSID/BSSIDs of the different APs. The actual configuration to be installed may come from different sources ranging from fully automatic algorithms computing the best allocation of e.g., frequency/transmission power to static allocations.

Data path establishment/modification Once the controller is connected to the data path elements it must decide the path data flows will follow. Depending on the technology of choice (e.g., OpenFlow, MVRP, SNMP, etc.) the data path will be installed based on static rules such as port allocated VLANs or it will be installed based on intelligent packet matching rules. As result of this operation each data path element should have a rule stating the forwarding behavior for packets belonging to a certain flow. The computation of the path to be installed depends on the technology of choice for the controller, since there are technologies computing a path in a distributed way and technologies that can run traffic engineering+policying algorithms.
In addition to the pro-active instantiation of a path described in the above text, a data path element may reactively interact with the controller due to some event, new packet arrival or pre-installed rule among others. Following this, the data path element may interact with the controller in order to build a path for a specific new flow that has just appeared and for any reason should not be forwarded through the pre-configured paths.

Data path teardown Once a data path is no longer in use, the rules indicating the forwarding behavior for each data path element may be removed. Generally this is done through lifetime timer expiration but the controller can choose to remove the rules actively. Note that rules installed in a data path can also be permanent or semi-permanent, not requiring the refreshing of the controller.

CIS communication and controller as proxy for CIS Nowadays CIS databases are filled with information provided by multiple sources but controlled by the operator. It would be desirable for the AN controller to be able to communicate with the CIS system in order to add/remove/modify its information based on its knowledge about the network. One example would be to update the list of services being advertised in 802.11aq based on information obtained from the network.
In addition to this, a controlled network can be configured in such a way that the controller is used as proxy for the CIS communication. In this way the controller will get the answer from the CIS and can modify it accordingly to get some expected behavior in the network. For example, a controller may be used as proxy to access a MIIS and after receiving the response filter it in order to remove the surrounding networks that do not belong to a specific operator, in this way enforcing some policy in the terminal.

Triggering technology specific functionality from the controller Although SDN controllers have been used typically to just setup data paths in the network and configure characteristics of the interface, the complete possibilities of controlling the specific features of the technologies have not been yet analyzed. The use of technology specific features by a controller can yield to further advantages for the network. For example, a controller may configure the QoS across a mix of wireless and wired domains, by triggering the QoS configuration mechanisms of each technology. Another example may use management frames of IEEE 802.11v to control the point of attachment of the user terminal. To open all this functionality the controller needs of a clear view of the capabilities of the interface and new APIs to trigger it in a remote way.

Event Handling TBD

Statistics Gathering TBD

4.4.8 Detailed procedures

TBD

4.5 Energy efficient Access Selection

4.5.1 System Model

The analysis we exploited for this contribution is similar to the one we presented in [12]. For the sake of completeness, we report here the main points of such contribution, adapting it when needed to the new approach presented. Due to the complexity of the scenario we decided to analyze, we introduce some minor assumptions. In the LTE environment, we assume that power control is just used during uplink transmissions. Following [62], the power control mechanism used achieves a particular Signal to Noise Ratio (SNR) target, thus coping at the receiver with path loss and shadowing. We also assume that when a device is scheduled, a full subframe is reserved to the device itself (Vertical First Scheduling (VFS) - [63]). In VFS, a device is scheduled into a subframe if it has a sufficient amount of traffic so to completely fill it (92400 bits when the best Modulation and Coding Scheme (MCS) is used). Due to the fact that the power consumption of a LTE card is directly proportional to the airtime of the transmission/reception and given the fact that a vertical scheduling reduces the airtime by design, VFS achieves high energy saving. Please note that VFS just determine the granularity at which the real scheduling of the resources at the eNBs runs. For example, when a Round Robin Scheduler is used, the devices with traffic are still scheduled one after the other, but a full subframe is reserved to each of the devices when their turn comes.

In the WiFi environment, we assume that the Ready to Send (RTS)/CTS mechanism is used both in legacy 802.11 and WiFi-Direct communications. In this way, we reduce the effects of the "hidden station" problem, relevant in a dense scenario. We also assume that devices do not change their transmission parameters, i.e., 802.11e is not enabled.

We assume that a feedback exists such that uplink and downlink demands (in bit/s) of each device that tries to connect to the access network are known. We consider the traffic of devices as best effort, i.e, the amount of traffic actually served can be lower than the one the devices demands. We also assume that some sort of feedback mechanism is in place (channel controls are already in use in both transmission technology we consider), so that controllers know in advance which would be the bit efficiency (or MCS) of devices if connecting to each of the PoA they are willing to accept.

Furthermore, we assume that the the interfaces of the devices are always on. In practice, we do not consider the consumption of the devices in idle mode, both for the WiFi network card, neither for the cellular transceiver. In practice, we analyze the efficiency of the devices while transmitting receiving. Such design choice is based on the normal behavior of the UEs, which rarely turn off its network interfaces. As side effect, the cost of relaying traffic is limited to the transmission and reception of WiFi-direct group members' traffic for each GO, and do not considers also the fact that the GO has to turn on its WiFi card. If we consider that WiFi transmissions are relatively energy efficient, it is easy to foresee that WiFi-relays plays an important role in the access selection mechanism we presented in this document.

4.5.2 Throughput and Power Consumption Computation

LTE Throughput Estimation: As we already mentioned in Section 4.5.1, we assume that each eNB has an estimation both of the bit efficiency (in bit/Orthogonal Frequency Division Multiplexing (OFDM) symbol) and the demand (in bit/sec) of the devices that access the internet through it. Such assumption holds both in uplink and downlink and, therefore, the computation we present holds in the two directions of transmission. We present the following computation without specifying which is the direction of the transmission. Nevertheless, please note that in uplink the average MCS used by each UE is the result of the power control mechanism in place. The key component of the power control mechanism is the interference sensed by the destination of the uplink transmissions, i.e., the eNB itself, that conversely depends on the number of Resource Blocks (RBs) assigned

to each other UEs in the system. The number of RBs assigned to each other UEs depends on the MCS they are able to use, i.e., on their bit efficiency. Therefore, in our analysis, we recursively computed the average bit efficiency achieved by each UE in the scenario, and the eNB is supposed to do so too, in case it does not possess bit efficiency statistics.

Let us assume that the set I is the set of devices associated to P. For each device $i \in I$ we assume to know the bit efficiency $b_i(P)$ used during its transmission and the demand, D_i. Then, for each i is possible to compute the number of RB/s, $R_i(P)$, that P should use to fully serve the demand of i, as follows.

$$R_i(P) = \frac{D_i}{b_i(P) \cdot N_S} \tag{4.1}$$

where N_S is the useful number of symbols per RB. Please note that $R_i(P)$ does not depend on the other devices that are scheduled on the same eNB P. Due to limited number of RB/s available at the eNB, it is possible that not all the demands can be served by P, i.e., the eNB is in saturation. If saturation happens, we assume that the scheduler used by the eNB shares the RB through simple water filling procedures [64]. When saturated, an eNB schedules exactly the same number of RB to the associated devices. If the demand of some of the devices is inferior to the scheduled quota, the exceeding RB are equally distributed among the remaining devices. If we denote with $R_i(I)$ the actual number of RB scheduled to i under eNB P when the set of associated devices to P is I, the throughput achieved by each device is:

$$T_i = R_i(I) \cdot b_i(P) \cdot N_S. \tag{4.2}$$

When computing the throughput achieved by each GO relaying traffic for their WiFi direct group, the demand considered is the sum of the personal demand of the GO, plus all the traffic it receives on his WiFi wireless interface.

LTE Power consumption estimation: If we analyse power consumption in LTE, downlink and uplink behave differently. In downlink, power consumption is mainly dependent on the percentage of time the device receives traffic. In uplink, power consumption is still dependent on the percentage of time a device remains active, but, mainly, it depends on the actual transmission power used by the device [63]. If we denote generically as $R_i^D(I)$ the generic number of RBs scheduled to i in downlink, the downlink power consumption is:

$$P_i^D = \frac{R_i(I)}{N_{RB}} \cdot (P_R - P_{idle}), \tag{4.3}$$

where N_{RB} represents the number of RB/s available in the downlink channel, while P_R is the power consumption of i while transmitting, P_{idle} the one consumed when idle. (4.3) holds because we assume that the VFS is the scheduler used by the eNB and, therefore, $\frac{R_i(I)}{N_{RB}}$ also represents the percentage of sub-frames scheduled for transmission for i. Please note that by design, we do not consider the power consumed when idle, due to the fact that we consider network interfaces as always active in devices.

In uplink, the main difference in (4.3) is that the power consumed while transmitting, P_R, is not fixed, but it is strictly dependent on the transmission power used by i, P_TX. If we denote as P_{R_i} the power consumed by i when transmitting, P_{R_i} can be computed as:

$$P_{R_i} = \alpha_H \cdot P_TX, \tag{4.4}$$

where α_H is a proportional factor. P_TX is expressed in dbm.

802.11 Throughput Estimation: In order to compute the throughput achieved by a set of devices I scheduled under the same AP/GO, we exploit the well known saturation throughput

computation shown in [65] and the linear relationship among devices demand and throughput, shown in [66].

Specifically, if we consider as saturated the generic set of devices $G \subseteq I$ and turned off the set S of the remaining devices, i.e., $S = I \setminus G$, an approximation of the saturation throughput of each device $i \in G$ is achieved through [65]. We denote such saturation throughput as $R_i(G)$. Exploiting the result in [66], if we activate one of the devices $j \in S$ and we increment its demand, the throughput achieved by each device i decreases linearly with the increase of the demand of j. The throughput achieved by i passes from $R_i(G)$ to the saturation throughput achieved when $G \cup \{j\}$ is the set of saturated devices, i.e., $R_i(G \cup \{j\})$. Let us now pick a generic order for S, $S_O = [s_1, ..., s_n]$, with n the number of devices in S. The maximum achievable rate of $i \in G$ can be computed as a function of the demands in S. In particular, such maximum throughput is obtained considering a hyperplane in an n-dimensional space that contains all the values achieved turning on and leading to saturation the devices of S, following the sorting S_O. Such values are achieved through the linear approximation mentioned above. Through the mechanism we propose, the maximum throughput for i, $T_i^{MAX}(S_O)$, can be achieved solving the following:

$$det \begin{vmatrix} T_i^{MAX}(S_O) - R_i(G) & D_{s_1} & \cdots & D_{s_j} & \cdots & \cdots & D_{s_n} \\ R_i(G \cup s_1) - R_i(G) & R_{s_1}(G \cup s_1) & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & & \ddots & & & \vdots \\ \vdots & \vdots & & & & & \vdots \\ R_i(G \cup \{s_1, ..., s_j\}) - R_i(G) & R_{s_1}(G \cup \{s_1, ..., s_j\}) & \cdots & R_{s_j}(G \cup \{s_1, ..., s_j\}) & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots \\ R_i(I) - R_i(G) & R_{s_1}(I) & \cdots & R_{s_j}(I) & \cdots & \cdots & R_{s_n}(I) \end{vmatrix} = 0.$$

Given the set of the demands, it is not easy to know the set of devices in saturation, i.e., the set of devices which demands are not completely served. Therefore, we can approximate the throughput of i as follows:

$$T_i = \min(D_i, \max_{G}(\max_{S_O} T_i^{MAX}(S_O))), \tag{4.5}$$

independently from the actual set of devices in saturation G. Note that the AP/GO is just one of the devices that access the medium following the Distributed Coordination Function (DCF) mechanism. Therefore, also the aggregate downlink traffic can be approximated as described above. We assume that the actual downlink throughput of the single devices is achieved splitting the aggregate downlink throughput proportionally to the individual downlink demands offered.

802.11 Power Consumption Estimation: Following [67], the uplink power consumption of a generic device i associated to a given AP/GO, when the set of devices associated to the same AP/GO is I, can be expressed as follows:

$$P_i = \gamma_{TX} \cdot \lambda_i + \rho_{TX} \cdot A_i + \rho_{RX} \cdot \lambda_i \cdot T_{ACK}, \tag{4.6}$$

where γ_{TX}, ρ_{TX} and ρ_{RX} are device dependent proportionality coefficients, λ_i is the frame generation rate, i.e., $T_i/E[L]$ where $E[L]$ is the average packet length, A_i is the percentage of time where the device is transmitting and T_{ACK} is the time of an ACK when a transmission is correctly received. A similar expression holds in downlink, where the reception frame rate, the downlink airtime and the power consumed to transmit an ACK are taken into account.

In order to compute P_i, the only variable whose value is unknown is the percentage of time in which i transmits, i.e., A_i. Considering as saturated a generic set of devices $G \subseteq I$ and turned off the set $S = I \setminus G$ of the remaining devices, through [65] it is possible to compute the probability τ that a generic time slot contains a transmission of $i \in G$. In such conditions, the airtime of i,

$A_i(G)$, is:

$$A_i(G) = \frac{\sum_{g \supseteq i; g \subseteq G} \tau^{|g|}(1-\tau)^{(|G|-|g|)} \cdot T_g}{\sum_{g \subseteq G} \tau^{|g|}(1-\tau)^{(|G|-|g|)} \cdot T_g}, \tag{4.7}$$

where the operator $|\cdot|$ represents the number of devices in the set under brackets, while T_g is the duration of the time slot where the set $g \subseteq G$ of devices is contemporary transmitting.

Given (4.7), in order to compute the airtime of i, i.e., A_i, we follow the same reasoning used to compute the throughput of the devices. Assuming that we turn on and lead to saturation a device $j \in S$, we assume that A_i increases linearly with the throughput achieved by j. Specifically, we assume that the airtime of i passes from $A_i(G)$ to $A_i(G \cup \{j\})$, when j reaches saturation. Then, as a result of the assumption we do, the airtime that i experiences is a function of the throughput achieved by the devices in S. If we consider the sorting $S_O = \{s_1, ..., s_n\}$ as the order in which we turn on and lead to saturation the devices of S, the airtime that i experiences, i.e., $A_i^{S_O}$, lay on a n-dimensional hyperplane which expression is the following:

$$det \begin{vmatrix} T_i & T_{s_1} & \cdots & T_{s_j} & \cdots & \cdots & T_{s_n} & A^i_{S_O} \\ R_i(G) & 0 & \cdots & 0 & \cdots & \cdots & 0 & A_i(G) \\ R_i(G \cup \{s_1\}) & R_{s_1}(G \cup \{s_1\}) & 0 & \cdots & \cdots & \cdots & 0 & A_i(G \cup \{s_1\}) \\ \vdots & \vdots & \ddots & & & & \vdots & \vdots \\ \vdots & \vdots & & & & & \vdots & \vdots \\ R_i(G \cup \{s_1, ..., s_j\}) & R_{s_1}(G \cup \{s_1, ..., s_j\}) & \cdots & R_{s_j}(G \cup \{s_1, ..., s_j\}) & 0 & \cdots & 0 & A_i(G \cup \{s_1, ..., s_j\}) \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots \\ R_i(I) & R_{s_1}(I) & \cdots & R_{s_j}(I) & \cdots & \cdots & R_{s_n}(I) & A_i(I) \end{vmatrix} = 0.$$

Since in this case we use as input the throughput achieved by the devices, it is possible to know what is the set G of devices that is in saturation. In order to compute A_i, is therefore sufficient explore the different sorting of S. Given G, we can compute A_i as follows.

$$A_i = \min_{S_O} A_i^{S_O}. \tag{4.8}$$

Dealing with the presence of GOs: In order to correctly compute the throughput achieved by devices in presence of GO, we should add some additional consideration. In WiFi indeed, the demand of all the users involved is needed in order to know the final throughput achieved. While the demand of all devices are given, the demand of the GO is represented by its downlink throughput in the cellular network. When the computation of throughput shown above is applied in each of the WiFi cliques that are present in the scenario, we achieve the downlink throughput of the users belonging to the WiFi-Direct groups, and the total demand of the GO in the uplink part of the transmission in the cellular infrastructure. Just thanks to this calculation, it is possible to obtain the uplink throughput achieved by all devices in LTE, and therefore also the effective throughput the members of WiFi groups can relay in uplink to the network. All in all, it is important to notice that the strict correlation among the uplink and downlink throughput in WiFi creates also a strict correlation among the uplink and downlink throughputs achieved by devices in the LTE part of access network. Indeed, the effective throughput of the GOs in downlink at the cellular network affects the throughput achieved by all devices in WiFi. On the same extend, the throughput achieved by devices in WiFi affects the demand in uplink of the GOs and, therefore, the throughput achieved by devices in the LTE network.

Table 4.9: Simulation Setup

Size Scenario	$400 \times 400\ m$
Demand	$7\ Mb/s$
LTE	
LTE Carrier	$2.45\ GHz$
Bandwidth	$20\ MHz$
Subframe Duration	$1\ ms$
Symbols per RB	84
Max Power Uplink	$200mW$
Power Downlink	$400mW$
SNR target Power Control	$17dB$
Max Symbol Efficiency	$5.55b/sym$
Noise Level	$3.18 \times 10^{-18} W/Hz$
Path Loss Model	Log Normal
$\delta = RSS_{min}$	$0.5\ b/sym$
WiFi	
WiFi Carrier	$2.4\ GHz$
Bandwidth	$20\ MHz$
MAC Layer Specifics	$80.11n$
Power Uplink and Downlink	$100\ mW$
Noise Level	$3.18 \times 10^{-18} W/Hz$
Path Loss Model	Log Normal
$\delta = RSS_{min}$	$0.5\ b/Hz$

4.5.3 Simulation set-up scenarios

In this section we present the details of the results presented in Section 2.3.1. We selected a dense scenario with 7 eNB, which position was fixed during the different simulation. 6 eNBs were disposed on a circle, at constant distance to each other, where the last eNB was disposed at the center of the circle. We also introduce in the scenario 10 APs and 250 UEs, entering the system one after the others. The position of the devices and of the APs was picked randomly. The demand of the devices was instead the same for all devices, and remained constant also in the different evaluations of the system. To benchmark the performances of our proposal, we use the traditional access selection mechanism used in heterogeneous access networks, i.e., a "WiFi first" policy. Following this approach, devices first try to access the network through the AP with higher Received Signal Strength (RSS), and then, if unsuccessful, they try to access the network through the eNB having the higher RSS.

In Table 4.9 we report the simulation parameters used.

Bibliography

[1] Antonio de la Oliva, Hassan Ali-Ahmad, Erick Bizouarn, M. Isabel Sanchez, Vincenzo Mancuso, Claudio Cicconetti, and Christian Vitale. D4.1, initial specification of connectivity management concepts and architecture, 2013.

[2] Isabel Sanchez, Antonio de la Oliva, Vincenzo Mancuso, Erick Bizouarn, Ahmet Serdar Tan, Martin Dr axler, Arianna Morelli, Christian Vitale, and Engin Zeydan. Consolidated specification for the connectivity management functions and interfaces, 2014.

[3] Peter Molnar Dirk Helbing. Social force model for pedestrian dynamics. *The American Physical Society*, 1995.

[4] 3GPP TS 29.060. `http://www.3gpp.org/DynaReport/29060.htm`. [Available online].

[5] Apache Hadoop. `http://hadoop.apache.org/`, 2015. [Online; accessed 02-April-2015].

[6] Cloudera. `http://www.cloudera.com/content/cloudera/en/documentation.html`, 2015. [Online; accessed 02-April-2015].

[7] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), September 2007.

[8] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[9] Arash Asadi and Vincenzo Mancuso. On the compound impact of opportunistic scheduling and d2d communications in cellular networks. In *MSWiM*, pages 279–288, 2013.

[10] J. D. Power and Associates. *2012 US Wireless Smartphone and Traditional Mobile Phone Customer Satisfaction Study*, 2012. Available at *http://www.jdpower.com/press-releases/2012-us-wireless-smartphone-and-traditional-mobile-phone-satisfaction-studies-volume*.

[11] Sven de Vries and Rakesh V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.

[12] Simone Mattiacci, Claudio Bottai, Vincenzo Sciancalepore Vincenzo Mancuso, Arash Asadi, Christian Vitale, Pablo Serrano, Laurent Roullet, Engin Zeydan, and Maria Isabel Sanchez. CROWD Deliverable D2.2: "Final specification for the wireless enhancements functions and interfaces". 2014.

[13] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019. White paper, February 2015.

[14] P. Daponte, L. De Vito, F. Picariello, and M. Riccio. State of the art and future developments of measurement applications on smartphones. *Measurement*, 46(9):3291 – 3307, 2013.

[15] IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012.

[16] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in Smartphone Usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.

[17] Wook Song, Yeseong Kim, Hakbong Kim, Jehun Lim, and Jihong Kim. Personalized Optimization for Android Smartphones. *ACM Transanctions on Embedded Computing Systems*, 13(2s):60:1–60:25, January 2014.

[18] Joel Sommers and Paul Barford. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, IMC '12, pages 301–314, New York, NY, USA, 2012. ACM.

[19] Shu Liu and Aaron Striegel. Casting Doubts on the Viability of WiFi Offloading. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, CellNet '12, pages 25–30, New York, NY, USA, 2012. ACM.

[20] Kensuke Fukuda and Kenichi Nagami. A Measurement of Mobile Traffic Offloading. In *Proceedings of the 14th international conference on Passive and Active Measurement*, PAM'13, pages 73–82, Berlin, Heidelberg, 2013. Springer-Verlag. `http://dx.doi.org/10.1007/978-3-642-36516-4_8`.

[21] Xian Chen, Ruofan Jin, Kyoungwon Suh, Bing Wang, and Wei Wei. Network Performance of Smart Mobile Handhelds in a University Campus WiFi Network. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, IMC '12, pages 315–328, New York, NY, USA, 2012. ACM.

[22] Junxian Huang, Qiang Xu, Birjodh Tiwana, Z. Morley Mao, Ming Zhang, and Paramvir Bahl. Anatomizing Application Performance Differences on Smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 165–178, New York, NY, USA, 2010. ACM.

[23] Qian Zhang, Chuanxiong Guo, Zihua Guo, and Wenwu Zhu. Efficient mobility management for vertical handoff between WWAN and WLAN. *IEEE Communications Magazine*, 41(11):102–108, Nov 2003.

[24] Chuanxiong Guo, Zihua Guo, Qian Zhang, and Wenwu Zhu. A seamless and proactive end-to-end mobility solution for roaming across heterogeneous wireless networks. *IEEE Journal on Selected Areas in Communications*, 22(5):834–848, June 2004.

[25] G. Bollano, D. Panno, F. Ricciato, M. Turolla, N. Vaccaro, and D. Ettorre. Enhanced Android Connection Manager: An Application-Based Solution to Manage Mobile Data Traffic. In *World Telecommunications Congress (WTC), 2012*, pages 1–6, 2012.

[26] Yao Liu, Fei Li, Lei Guo, Bo Shen, and Songqing Chen. A Comparative Study of Android and iOS for Accessing Internet Streaming Services. In *Proceedings of the 14th international conference on Passive and Active Measurement*, PAM'13, pages 104–114, Berlin, Heidelberg, 2013. Springer-Verlag.

[27] Apple Inc. iOS7 Developer Library. [Online]. Available: https://developer.apple.com/library/ios/navigation.

[28] R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), March 1997. Updated by RFCs 3396, 4361, 5494, 6842.

[29] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), July 2003. Updated by RFCs 4361, 5494, 6221, 6422, 6644, 7083, 7227, 7283.

[30] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), September 2007. Updated by RFCs 5942, 6980, 7048.

[31] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007.

[32] F. Gont. A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC). RFC 7217, April 2014.

[33] B. Aboba, D. Thaler, and L. Esibov. Link-local Multicast Name Resolution (LLMNR). RFC 4795 (Informational), January 2007.

[34] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), October 2002. Updated by RFC 4604.

[35] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810 (Proposed Standard), June 2004. Updated by RFC 4604.

[36] H. Holbrook, B. Cain, and B. Haberman. Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast. RFC 4604 (Proposed Standard), August 2006.

[37] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.

[38] Wenjun Gu, Zhimin Yang, Dong Xuan, Weijia Jia, and Can Que. Null Data Frame: A Double-Edged Sword in IEEE 802.11 WLANs. *IEEE Transactions on Parallel and Distributed Systems*, 21(7):897–910, July 2010.

[39] Aruna Balasubramanian, Ratul Mahajan, Arun Venkataramani, Brian Neil Levine, and John Zahorjan. Interactive wifi connectivity for moving vehicles. *ACM SIGCOMM Computer Communication Review*, 38(4):427–438, 2008.

[40] 3GPP TS 23.402. Architecture Enhancement for Non-3GPP Accesses (Release 12) v12.5.0, June 2014.

[41] IEEE Standard for Information Technology-Telecommunications and information exchange between systems. Local and Metropolitan networks. Specific requirements. Part II: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 9: Interworking with External Networks. *Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11w-2009, IEEE Std 802.11n-2009, IEEE Std 802.11p-2010, IEEE Std 802.11z-2010, and IEEE Std 802.11v-2011*, pages 1–208, Feb 2011.

[42] J. Seedorf and E. Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement. RFC 5693 (Informational), October 2009.

[43] IEEE Standard for Local and metropolitan area networks - Media Independent Handover Services. *IEEE Std 802.21-2008*, Jan 2009.

[44] L. Sarakis, G. Kormentzas, and F.M. Guirao. Seamless service provision for multi heterogeneous access. *Wireless Communications, IEEE*, 16(5):32–40, October 2009.

[45] Telemaco Melia, Carlos J. Bernardos, Antonio de la Oliva, Fabio Giust, and Maria Calderon. IP Flow Mobility in PMIPv6 Based Networks: Solution Design and Experimental Evaluation. *Wireless Personal Communications*, 61(4):603–627, 2011.

[46] A De La Oliva, C.J. Bernardos, M. Calderon, T. Melia, and J.C. Zuniga. IP flow mobility: smart traffic offload for future wireless networks. *IEEE Communications Magazine*, 49(10):124–132, Oct 2011.

[47] M. Wasserman and P. Seite. Current Practices for Multiple-Interface Hosts. RFC 6419 (Informational), November 2011.

[48] D. Thaler, R. Draves, A. Matsumoto, and T. Chown. Default Address Selection for Internet Protocol Version 6 (IPv6). RFC 6724 (Proposed Standard), September 2012.

[49] Cisco. The Zettabyte Era.

[50] IETF DMM WG: http://datatracker.ietf.org/wg/dmm/charter/ .

[51] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, V. Mancuso, M. Reddy Sama, P. Seite, and S. Shanmugalingam. An SDN-Based Network Architecture for Extremely Dense Wireless Networks. In *IEEE SDN4FNS*, pages 1–7, November 2013.

[52] Andy Myers, Eugene Ng, and Hui Zhang. Rethinking the service model: Scaling ethernet to a million nodes. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, 2004.

[53] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a Globally-deployed Software Defined WAN. In *SIGCOMM '13*, pages 3–14, New York, NY, USA, 2013. ACM.

[54] Kok-Kiong Yap, Masayoshi Kobayashi, David Underhill, Srinivasan Seetharaman, Peyman Kazemian, and Nick McKeown. The Stanford OpenRoads Deployment. In *WINTECH '09*, pages 59–66, New York, NY, USA, 2009. ACM.

[55] Kok-Kiong Yap, Rob Sherwood, Masayoshi Kobayashi, Te-Yuan Huang, Michael Chan, Nikhil Handigol, Nick McKeown, and Guru Parulkar. Blueprint for Introducing Innovation into Wireless Mobile Networks. In *ACM SIGCOMM VISA Workshop 2010*, pages 25–32, New York, NY, USA, 2010. ACM.

[56] Kok-Kiong Yap, Masayoshi Kobayashi, Rob Sherwood, Te-Yuan Huang, Michael Chan, Nikhil Handigol, and Nick McKeown. OpenRoads: Empowering Research in Mobile Networks. *SIGCOMM Comput. Commun. Rev.*, 40(1):125–126, January 2010.

[57] K. Pentikousis, Yan Wang, and Weihua Hu. Mobileflow: Toward software-defined mobile networks. *Communications Magazine, IEEE*, 51(7):44–53, July 2013.

[58] J. Kempf, B. Johansson, S. Pettersson, H. Luning, and T. Nilsson. Moving the mobile Evolved Packet Core to the cloud. In *WiMob 2012, IEEE*, pages 784–791, Oct 2012.

[59] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. Towards Programmable Enterprise WLANs with Odin. In *HotSDN '12*, pages 115–120, New York, NY, USA, 2012. ACM.

[60] P. Dely, A. Kassler, and N. Bayer. OpenFlow for Wireless Mesh Networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6, July 2011.

[61] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi. Wireless Mesh Software Defined Networks (wmSDN). In *WiMob 2013, IEEE*, pages 89–95, Oct 2013.

[62] 3GPP. Evolved universal terrestrial radio access (e-utra); physical layer procedures (release 12). TS 36.213 - RP-63, 2014.

[63] Flavia Project. Analysis and design of solutions for scheduled technology enhancements. Deliverable 5.3, 2013.

[64] John Proakis. *Digital Communications*. McGraw-Hill Science/Engineering/Math, 4 edition, August 2000.

[65] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE JSAC*, 18(3):535–547, 2000.

[66] Antonio De La Oliva, Albert Banchs, and Pablo Serrano. Throughput and energy-aware routing for 802.11 based mesh networks. *Computer Communications*, 35(12):1433–1446, 2012.

[67] Andres Garcia-Saavedra, Pablo Serrano, Albert Banchs, and Giuseppe Bianchi. Energy consumption anatomy of 802.11 devices and its implication on modeling and design. In *CoNEXT*, pages 169–180, 2012.