Deliverable D6.5

Assessment Report

Public, Version 1.0, 8 May 2015

Authors

FT	Patrick Truong, Bertrand Mathieu
AL-BELL	
IMDEA	Nicola Bui (editor), Foivos Michelinakis, JoergWidmer
TSP	
ALUD	Klaus Satzke
TUD	Christian Koch, Julius Rückert, Fabian Kaup, Leonhard Nobach, Silvie Vidal, Véronique Henry, David Hausheer
TI	Fabio Luciano Mondin, Claudio Venezia
~~*UCAM*~~	
UC3M	Juan Miguel Carrascosa, Rubén Cuevas
Reviewers	Chris Hawinkel

Abstract

This deliverable concludes the activities of WP6 and is meant to collect and summarize the final results obtained by the eCOUSIN project. As per the previous deliverable it is structured according to the use cases defined in the project. In fact, the set of tests performed to validate the project objectives are split into 4 groups. Each of these faced different challenges and adopted a variety of validation techniques. Furthermore the deliverable is split in two main sections, of which, the first lists the tests performed and maps then onto the functional architecture and the final software release, whereas the second, provides short descriptions of the tests that were already discussed in previous deliverables and long descriptions for those tests that were specifically performed for this final evaluation. As an overall summary, the final assessment was able to cover and validate the whole architecture, obtaining successful outcome in all the sets of tests, but in a few minor occasions, where the tests were only partly passed. In all these cases the partial success was due to causes external to the project.

Project funded by the European Union under the
Information and Communication Technologies FP7 Cooperation Programme
Grant Agreement number 318398

EXECUTIVE SUMMARY

Deliverable D6.5 is the eCOUSIN assessment report. The main objective of this document is to verify whether the project could satisfy the proposed objectives. To this end, at the beginning of the second year of the project, WP6 defined a set of tests that should be verified to validate the results by the end of the project.

As per the previous deliverables of WP6, D6.5 is organized according to use case definitions: in particular, four areas are identified, which are:

- Content Placement (CP)
- Personal Sharing Clouds (PSC)
- Information-centric Networking (IN)
- Content Offloading for mobile networks (CO)

These four areas have different requirements as identified by WP2 and pose different challenges. Nevertheless, WP6 aimed at unifying the many perspectives arisen from the use case analysis into a common framework covering the entire functional architecture.

Table 1. Summary of the assessment and architectural mapping

Architecture		**Assessment**				
Layer	**Functionality**	**CP**	**PSC**	**IN**	**CO**	**Outcome**
Social	Social Data Collector	🟧	🟪	🟦	🟥	**Success**
	Content Information Collector	🟧				**Success**
	Data Analysis, Mining, Aggregation	🟧	🟪	🟦	🟥	**Success**
	Social Predictor	🟧			🟥	**Success**
	Social-aware Content Naming Scheme			🟦		**Success**
Content Dissemination	Content Placement Strategies	🟧			🟥	**Success**
	Content Copy Selection Algorithms		🟪			**Success**
	Content Dissemination Algorithms			🟦	🟥	**Success**
	Content Look-up		🟪			**Success**
Network	Network Monitoring				🟥	**Success**
	In-network Content Routing/Caching			🟦		**Success**
	Network Resource Configuration			🟦	🟥	**Success**

For instance, Table 1 summarizes the mapping between the architectural functions and the four test areas: the second column of the table lists the functionalities of the eCOUSIN architecture, columns from the third to the sixth checks whether a test has been performed for the functionality related to the row among the tests performed in area identified by the column. The last column reports the overall status of the tests of each functionality. All functionalities have been successfully tested in at least one test area.

In addition, not only did we verify that all tests proposed in the final assessment plan [D6.2] have been performed and obtained a successful status, but some additional tests have been added to highlight the latest improvements obtained by eCOUSIN.

The rest of the document is organized so that a first section, provides detailed summaries for each of the tests including which aspects of the architecture are concerned, which software modules have been used and where the results of the test appeared first. The last information is necessary, since some of the tests have been performed during earlier phases of the project and are only summarized here. The second and main section of the deliverable reports in more details the new tests that did not appear in any other deliverable of the project and summarizes the others. In order to realize these additional tests some effort have been put to expand the results obtained within the technical work packages.

For instance, the following contributions provide new insight on several topics: Section 2.3.2 adds locality awareness to content delivery; Section 2.3.3 evaluates route calculation on Rocketfuel topologies; Section 2.4.8 elaborates on the accuracy of throughput prediction and propose new models and results from a measurement campaign; Section 2.4.9 enhances the resource allocation optimization to the multi-user variable quality scenario; Sections 2.4.12 and 2.4.13 provides further insights on the measurement campaigns performed by eCOUSIN.

TABLE OF CONTENTS

1. OVERVIEW OF THE ASSESSMENT REPORT

The following tables map the tests with tested functionalities, software used, where to find the related results within the eCOUSIN corpus and the outcome of the test.

1.1 Content Placement

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
Test CP1	To evaluate the capacity to capture the information regarding the social graph associated with a user as well as the information about a specific uploaded content; and properly store the derived information in a database for further use.	Social Data Collector, Content Info Collector	OSN Crawlers	D3.2 {S2.2}, D3.3 {S3.1}	Success
Test CP2	Evaluation of the social and content information to produce a social- based prediction regarding the most likely locations where a specific content is expected to be consumed.	Data Analysis Mining Aggregation, Social Predictor	Content Extractor, Info Mapping, Social-based Consumption Predictor, Content Placement Engine	D3.2 {S1.2.1}, D4.1 {S2.8}	Success
Test CP3	Evaluation of the emulation environment. Existence of users in the system publishing and consuming content.	Data Analysis Mining Aggregation, Social Predictor	Content Extractor, Info Mapping, Social-based Consumption Predictor, Content Placement Engine	Here {S2.1.3}	Success
Test CP4	Evaluation of the correct functionality of the different content placement strategies: LRU and FIFO.	Content Placement Strategies	Content Extractor, Info Mapping, Social-based Consumption Predictor, Content Placement Engine	D4.3 {S3.4}, Here {S2.1.4}	Success

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
Test CP5	Final evaluation of the emulation environment. Analyze the correct integration and communication of the different modules. Evaluate the performance of social-enhanced content placement solutions and compare them to traditional content placement mechanisms.	Social Data Collector, Content Info Collector, Data Analysis Mining Aggregation, Social Predictor, Content Placement Strategies	Content Extractor, Info Mapping, Social-based Consumption Predictor, Content Placement Engine	Here S{2.1.4}	Success

1.2 Personal Sharing Clouds

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
PSC01	Connect Media Centre to one or more specific Facebook identities and grab the list of friends who did the same and their media centres IP addresses and ports	Social Data Collector, Data Analysis and Mining	Personal Sharing Clouds	Here {S2.2.1}	Success
PSC02	Once Connected, the media centre, should explore the network that was set up browsing for remote resource to show as local	Content Look Up, Content Copy Selection, Content Dissemination	Personal Sharing Clouds	Here {S2.2.2}	Success
PSC03	Media Centre should be able to fully work in an home environment	All modules	Personal Sharing Clouds	Here {S2.2.3}	Success
PSC04	Usability Testing on the prototype	All modules	Personal Sharing Clouds	Here {S2.2.4}	Success

1.3 Information Centric Networking

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
IN1	Deploy a Twitter-like application over an NDN-based architecture, and assert our proposed local-aware and name-based routing scheme using users' social interactions for optimizing content delivery	Social Data Collector, Data Analysis, Mining and Aggregation, Content Naming Scheme, Content Dissemination, In-Network Content Routing/Caching.	OSN server, NDN router for name-based and social-driven routing and caching.	We checked the GUI of our Twitter-like application on end-users to verify that messages were well published and retrieved. D5.3 {S4.2.3} Here {S2.3.1}	Success
IN2	Integrate the video live streaming capability into the Twitter-like application over the local-aware and NDN-based routing architecture.	Social Data Collector, Data Analysis, Mining and Aggregation, Content Naming Scheme, Content Dissemination, In-Network Content Routing/Caching.	OSN server, NDN router for name-based and social-driven routing and caching.	The OSN application is implemented in Javascript. Live streaming of videos was not possible due to poor performance (delay too long) with chunk encryption in Javascript. So we only provide http-based video streaming. Here {S2.3.1}	PARTLY Success
IN3	Social-aware caching: an NDN forwarding node caches a content only if there are more than N users potentially interested in the content.	Content Dissemination, In-Network Content Routing/Caching, ICN controller (for caching configuration)	OSN server, NDN router for name-based and social-driven routing and caching.	Here {S2.3.1}	Success
IN4	Route calculation: evaluation of routes from OSN server	Network Resource Config	ICN Controller ICN Clients	Here {S2.3.3}	Success

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
	towards testbed users calculated by the ICN controller (Orange testbed and Rocketfuel topologies)				
IN5	Programming of forwarding elements Check the ICN controller GUI with ICN extension for appearance of ICN nodes and ICNIDs. Check the ICN controller logging info for registration of new ICN clients. Check if the ICN Client receives the ICN_MOD messages issued by the controller.	Network Resource Config	ICN Controller ICN Clients	Here {S2.3.3}	Success
Test IN6	Overall evaluation of the testbed: Twitter-like application with video sharing over the NDN-based architecture with the ICN controller for dynamically configuring caching and routing. We notably verify that the ICN controller can dynamically configure our local-aware and social-driven routing scheme (e.g. add route in the local NDN routers if close	Social Data Collector, Data Analysis, Mining and Aggregation, Content Naming Scheme, Content Dissemination, In-Network Content Routing/Caching, ICN controller for social-aware caching and dynamic local-aware and social-driven routing configuration.	OSN server, NDN router for name-based and social-driven routing and caching, ICN controller, Monitoring tool for visualizing NDN traffic.	Analytical evaluation available here {S2.3.2} [TRUO15a]	Success

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
	friends/followers are on-line, remove it if not).				

1.4 Content Offloading for Mobile Networks

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
CO1	Different video posts including videos from different video platforms are posted on a Facebook feed. Afterwards, this Feed is crawled and the information of the video posts retrieved is compared with the expected ones.	Social Data Collector, Data Analysis, Mining, and Aggregation ([D2.4], Section 2.2.1.3)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success
CO2	Aggregated information regarding a user's Facebook feed are visualized and assessed manually.	Social Data Collector ([D2.4], Section 2.2.1.3)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success
CO3	The amount and the integrity of the data which is sent from the app to the database server is assessed.	Data Analysis, Mining and Aggregation([D2.4], Section 2.2.1.3)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success
CO4	The assignment and effectiveness of priorities to prefetching video candidates is assessed.	Social Predictor([D2.4], Section 2.2.1.3)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success
CO5	The video player is tested with videos from different video hoster and under different network	Video Player ([D6.2], Section 2.4.5)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
	connectivity settings.				
CO6	The capability of the app to download videos from different portals ordered in order of the priority assigned to them is assessed under different network connectivity settings.	Download Client ([D6.2], Section 2.4.9)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success
CO7	The download scheduling is assessed with respect to the time and exceptions which are well known from video hosters, e.g. as video which has been deleted.	Download Scheduler ([D6.2], Section 2.4.8), Download Client (D6.2, Section 2.4.9)	Social Prefetching App ([D6.4], Section 2.4.4)	[WILK15], [KOCH15], [KOCH14], [KOCH14b]	Success
CO8	Evaluation of Bandwidth Availability Prediction. Verify whether standard prediction techniques can be used to estimate bandwidth fluctuations. In addition, the reliability of these techniques is studied and modelled. The goal was demonstrating that it is possible to use prediction techniques with known reliability to drive optimization solutions.	Network monitoring,	Bandwidth optimization with imperfect knowledge, Lightweight measurements, Tracing App	D4.3 {3.2.4} D5.2 {S3.1.2, 5.3.4} D5.3 {3.1, 4.3.3.1.1} [BUI14], [BUI14b], [MICH15], [BUI15b]	Success
CO9	Evaluation of Bandwidth Allocation Optimization. The objective was to	Network Monitoring, Content Placement Strategies, Content Dissemination	Bandwidth optimization on perfect knowledge,	D4.2 {2.2.2, 2.3.3.2} D4.3 {3.2.4}	Success

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
	enhance network efficiency and increasing users' quality of experience. To this end bandwidth prediction techniques are used to forecast when it is more efficient to use network resources.	Algorithms, Network Resource Config	Bandwidth optimization with imperfect knowledge, NS-3 Simulator Extensions, NS-3 Simulator Scenarios, Network Visualization	[BUI14], [BUI14b], [MICH15], [BUI15], [BUI15b]	
CO10	Evaluation of the Network Visualization. The test was expected to verify the capabilities of the network visualization tools. These capabilities include, but are not limited to, KPI visualization as well as network dynamics.	External	Network Visualization	D4.2 {2.3.3.2} D5.3 {3.1.3, 4.2.2.5}	Success
CO11	Evaluation of the Passive Measurement Module. The passive measurement module is expected to feed prediction and resource configuration modules with the needed samples. The objective of this test is to evaluate the reliability of such a measurement technique.	Network Monitoring	Passive measurements, Lightweight measurements, Bandwidth optimization with imperfect knowledge	D4.3 {3.2.4} D5.2 {5.3.4} D5.3 {3.1.2} [MICH15], [BUI15b]	Success
CO12 (R1)	The Social Prefetching App's usage by the user study's participants is investigated.				Success
CO13	Final Analysis of the	With reference to	D2D Opportunities		Success

Test Code	Description	Involved functionalities	Involved Software Components	Results Available	Test Outcome
(R2)	D2D Traces provided by Orange	WP2 architecture	Evaluator "OTraces" ([D6.4], Section 2.4.2)		

2. DETAILED TEST INFORMATION

2.1 Content Placement

2.1.1 Test CP1: Data Collection

The goal of this test is to check the data collection functionality. Collecting, processing and saving data from Twitter using the API is a complex task due to limitations and the large amount of data needed to collect.

In order to achieve the goal of this test next steps are followed:

- Connection to the Twitter API.
- Collect random public tweets.
- Collect the information associated to those users who published tweets: number of friends, number of followers, location, tweets…
- Store the information in Databases.

2.1.2 Test CP2: Data Analysis and Social Prediction

The purpose of this test is to evaluate the social and content information to produce a social- based prediction regarding the most likely locations where a specific content is expected to be consumed. In this test we also evaluate the performance of the new functionality provided to the tool: geographical distribution of users. This is a new option to define social scenarios where users, during the emulation, are not distributed homogeneously around the world.

2.1.3 Test CP3: Emulation Environment

During this test we check the new emulation environment. The addition of geographical distribution of users has an important impact in the emulation environment due to non-homogeneous load of the system. Therefore, this test assures proper operation of users in the system publishing and consuming content.

2.1.4 Test CP4: Content Placement Strategies

Test CP4 is in charge of evaluating the correct functionality of different content placement strategies included in the emulation environment. In this test we evaluate the performance of all the different caching algorithms included: LRU, s-LRU, FIFO, s-FIFO where LRU and FIFO are associated with the traditional content placement algorithm and, s-LRU and s-FIFO are the social-enhanced caching algorithms (see D4.3, section 3.4).

2.1.5 Test CP5: Integrated Demonstrator Test

Initially, the "Emulation Environment" was designed to allow a flexible configuration depending on the requirements and scenarios. During the project the emulation environment has gradually incorporated a large number of new parameters to provide an adaptive and flexible tool to suit different scenarios. The purpose of this test is to assess these new inclusions. A detailed list of the most recent inclusions is shown below:

- Option to provide geographical distribution of users.
- Integration of two additional content placement algorithms: FIFO and s-FIFO in the demonstrator.
- Option to include content popularity functionality on the emulation environment. By using long-tail content the tool provides a more realistic emulation of an online social network where popular content is more likely to be consumed and therefore, to be cached.
- Inclusion of push version for all content placement algorithms in the emulation environment. Therefore the tool includes two final strategies: pull and push. In the push algorithm when a user uploads a content item, it is automatically pushed to the caching servers of the distributed caching system (in the case of s-LRU and s-FIFO only if the defined condition holds). In the pull algorithm a content item is cached in a local cache only after a request happens and in the case of s-LRU and s-FIFO only if the defined condition holds.

In order to check the integrated demonstrator dozens of experiments were conducted to compare the performance of the demonstrator and the level of enhancement between traditional content placement strategies (i.e. LRU and FIFO) and social-enhanced caching algorithms (i.e. s-LRU and s-FIFO).

Considering the flexibility of the tool and the large number of configurable parameters: number of caches, size of cache, social threshold, distribution of users and so on; six different scenarios were tested using LRU/s-LRU and FIFO/-s-FIFO. Table 2 summarizes the main parameters of the experiments.

Configuration	Cache Size	Social Threshold	Number of Caches
1	25	20%	5
2	25	30%	5
3	20	30%	5
4	20	20%	5
5	20	25%	5
6	30	25%	5

Table 2: List of the configuration parameters for the six scenarios.

The metric used to evaluate the performance of each solution (LRU/s-LRU and FIFO/s-FIFO) is the Hit Ratio (HR), which is, the fraction of users requests that found the requested content in the local cache server. Each configuration is run separately in 5 caches. Thus Table 3 shows the median value of the aggregated results conducted.

Configuration	HR of LRU	HR of s-LRU	HR of FIFO	HR of s-FIFO
1	25.31%	26.09%	22.29%	23.45%
2	33.98%	34.42%	23.91%	25.19%
3	29.78%	30.93%	26.85%	30.35%

4	30.53%	32.44%	27.75%	29.09%
5	20.58%	22.52%	29.63%	31.58%
6	38.71%	39.57%	23.47%	26.06%

Table 3: Median values of the Hit Ratio for each algorithm and configuration.

Based on this metric (i.e. HR) we compute the final enhancement value which is obtained as the percentage of HR of s-LRU over LRU (s-FIFO over FIFO). Figure 1 and Figure 2show the results comparing LRU with s-LRU and FIFO with s-FIFO for each configuration defined in Table 2, respectively. Again the figures show median values.

From Figure 1 we can observe that, independently of the configuration, s-LRU outperforms LRU algorithm. In fact, the enhancement, in median, ranges from 1.5% to 9.4% in Configuration #2 and Configuration #5 respectively.

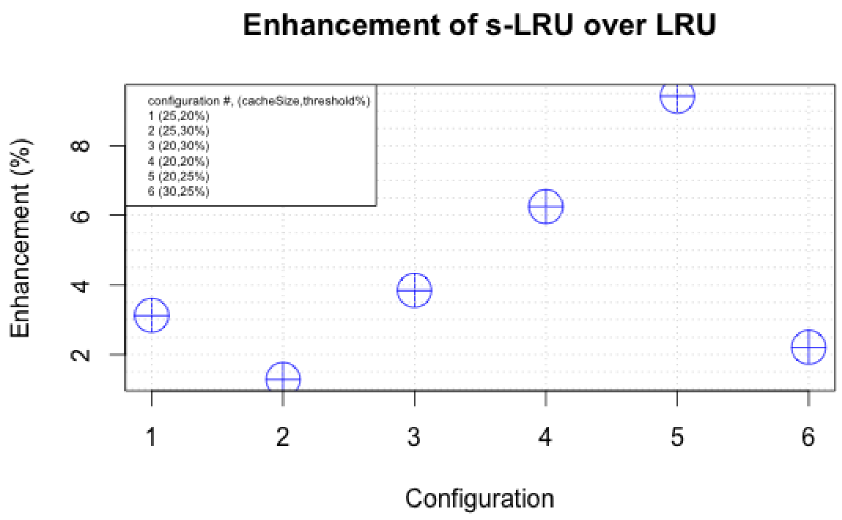


Figure 1 Median enhancements of s-LRU over LRU for six different configurations

On the other hand, Figure 2 shows the results comparing FIFO with s-FIFO for each configuration defined in Table 2 too. It is worth to mention that all the above comments should be also considered in this figure. In this case, the enhancement, in median, ranges from 5.3% to 13.2% in Configuration #4 and Configuration #3 respectively.

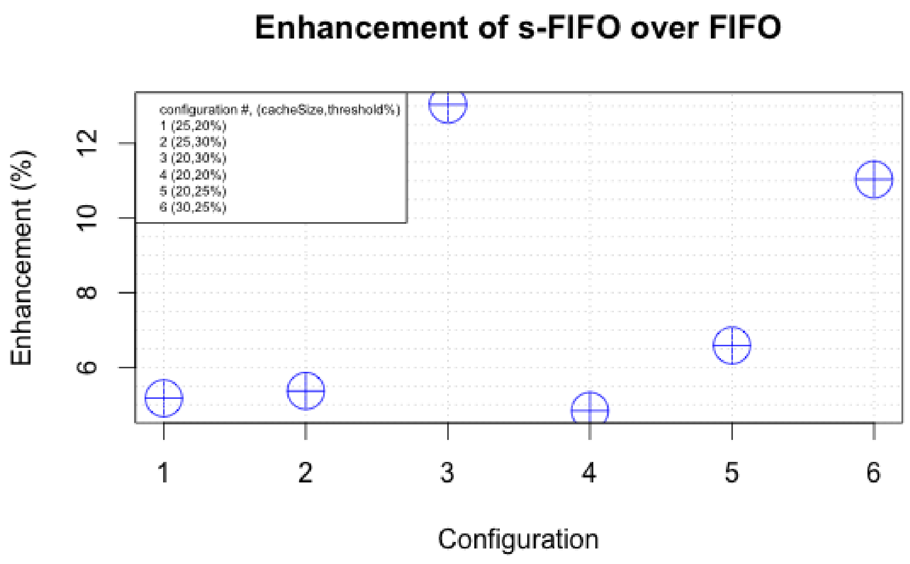


Figure 2 Median enhancements of s-FIFO over FIFO for six different configurations

From these figures we can obtain the following conclusions:

- For these scenarios, s-LRU and s-FIFO outperform traditional algorithms.
- High variability because of the impact of different configuration parameters.
- In terms of HR, the best scenario (i.e. highest HR 39.57%) is obtained using a value of 30 for cache size, 25% for social threshold and s-LRU as content placement algorithm. Based on the enhancement values obtained, s-FIFO offers higher improvement over FIFO comparing to the improvement values of s-LRU over LRU. In terms of the highest enhancement value of a social content algorithm over tradition alone, the configuration number 3 of s-FIFO reflects the best scenario with an improvement of 13%.

2.2 Personal Sharing Clouds

2.2.1 Test PSC01: Friends Discovery

This test had to be considered "passed" if the list of friends having the app is shown on the media centre application home page.

User side, the test performed involves a list of specific actions to do:

1. Connect Via Browser to media centre
2. Connect the media centre to a specific Facebook Account
3. Authorize the Facebook App if necessary
4. Check the list of Facebook Friends who gave their permission to the app.
5. Check their online status.

In order to perform the test, some preliminary operation had to be done to build the network of social relationships to test the spirit, moreover, this network had to evolve during the test in order to have reliable testing results.

Phase 2: Light Testing

The first step was in creating a fake network of social relationships. Three Facebook users (Namely User Alice, Bob and Charles) were created, and friendship relationships were set among them.

After that, Alice and Bob connected to the media centre performing the operations described above, authorizing the app. The expected result (reached) was to have just one friend with the app (at point 4) and see it as “online” (point 5).

Afterwards, Bob was disconnected from the media centre and Charles was connected to it. As expected, the list of friends with app (at point 4) increased of one unit, but only one among two users in the list was shown as “online” (point 5).

Phase 3: Real User Testing

The social network was extended with real Facebook users. This implies that both the initial discovery phase and the phase of “online” friends discovery was doomed to get more complicated. The Facebook users involved in testing had about 300 to 500 friends. We tested the connection phase for about 10 Facebook users, who were already friends among them; moreover, we asked them to include Alice, Bob and Charles into their friend lists.

The number of media centres available for testing was 4, so we did several testing by connecting and disconnecting users, checking coherence between the number of friends reported and the number of friends actually.

In this case the test was passed completely: the Connection phase including authorization of application on the target social network worked correctly during both the light and real user testing phases and at the end of the process a clear indication was given of the fact that a specific friend was online or offline.

2.2.2 Test PSC02: Resource Discovery

Once connected, users should be able to browse remote resources as local. In order to do that the resource discovery phase, leveraging the module called “Resource Locator” or “Content Lookup” in the eCOUSIN architecture, should work well.

The test aims at checking the discovery phase functionalities. User side, the operations to perform are these:

1. Connect to the media centre and connect Facebook account
2. Check the connected device list. Devices should have a local IP
3. Browse remote resources via local IP
4. Open content on remote resources, with local IP+filename naming. (Described in WP5).

The test was performed in both “Light” and “Real users” Scenarios following directly the start-up phase of test PSC01.

In general, the approach we followed, was to connect one media centre to one user, even if theoretically more than one user in order not to have doubt on actual resource location.

From a Networking point of view, we had one media centre standing behind an ADSL connection and all of the others were connected through an access point to an LTE module, this in order to have them standing on different networks.

This test was passed since the expected result was that the list of remote content should be browsed by opening the remote media centre name links and that was achieved. Moreover, no difference between the local and remote links should be shown in presentation. Our tests verified that in all experiments. The only small remark is related to bandwidth availability at both endpoints: a “local-like” user experience is only available when the end-to-end bitrate is high enough to support the streaming.

As a kind of “extra testing”, we also connected a uPnP enabled Smart-TV to the WiFi of the media centre standing behind the ADSL connection. The TV was able to play a remote video like it was a local one, performing this way a kind of “content Centric Streaming”, meaning that once the connection is set, if the connection speed is good, this use case can offer interesting scenarios.

2.2.3 Test PSC03: Operation in Home Environment

Media centres in general does not have a very powerful hardware, but the software should anyway be able to work in a typical home environment.

In this sense, progressive stress testing had to be performed in a different way with respect to the previous PSC01 and PSC02 testing. In this case the number of user is not so relevant, while it is relevant the number of uPnP devices connected to the home network.

So the operation performed in order to grab results from this tests were:

1. Connect two users to their media centres.
2. Connect one more device to the same WiFi network
3. Browse remote content
4. Connect more devices, one by one, and each time try to browse remote content.

Devices used were smart TV, laptops and smartphones. The outcome of the test was that up to 10 devices the media centres could safely manage the filtering process for the uPnP packages. When the number of devices is higher than 10 the system start slowing down, eventually freezing for a while. Much changes also depending on the type of devices which are connected. The Smart TVs for example produce huge amount of traffic, while smartphones tend to create less problems. The test were performed using the QNAPP media centre which is one of the most suitable candidates for industrial exploitation.

In order to test the software part, we decided to run the test also on a laptop, into an office environment. The WiFi Used is normally very polluted with about 30 to 35 devices connected and the laptop (a commercial Sony Vaio with an Intel i7 processor) didn’t have any problem in managing that amount of uPnP traffic, this could be anyway an attention point, considering the increasing number of connected devices that is growing day by day.

2.2.4 Test PSC04: Usability Testing

This was more an evaluation than a typical test. The research group of Telecom Italia dealing with eCOUSIN, can leverage on a pool of expert in cognitive psychology, that are normally involved in designing user interfaces for apps and services by Telecom Italia.

User Centred design, when possible, is a kind of rule for Telecom Italia, which aims at involving final users in the design process since the very start via Focus Groups or service testing.

In the cases in which it is not possible to involve a reasonable number of final users, due to cost or time, there are a number of guidelines (Nielsen Heuristics evaluation is probably the main example), aiming at evaluating the usability of a specific service or object. What we did here is a kind of pre-evaluation, in order to understand, also due to exploitation purposes, the amount of work necessary to turn a prototype into a product.

The Test had as a result that the user interface is fine for a prototype, since it shows clearly the aim of the project. Besides this, if this has to become a sellable product, this interface should be slightly more friendly and understandable. The most important result, from the project perspective, is that bandwidth is critical to have a user interface for whom local resource and remote has the same access mode. The user interface instead should be improved in order to be considered a real product. Such an activity should include target final users and specific indications should be collected but this level of detail for the user interface is outside of the scope of the project.

2.3 Information Centric Networking

2.3.1 Scenarios for tests IN1, IN2, IN3 and IN6

We describe in this section the progressive scenarios that we used to perform the different functional tests IN1, IN2, IN3 and IN6. Figure 3 shows the network topology with some users (and the social graph of user Joe) of our Twitter-like application.

We use an NDN-based architecture to optimize the networking behaviour while better reflecting the local end-users behaviour. Popular end-users, whose content is consumed worldwide, should have a different way of working than non-popular local end-users, whose content will be locally consumed. We then perform social-driven local routing between the end-users who are in the immediate vicinity. Locality is defined by network routing hop, and we consider local network regions of 2 routing hops: two users are local if there are separated by 2 routing hops (or any other value depending on the design configuration).For example, in Figure 3, the users Joe, Alice and Mary are in a same local region. The deliverable D5.3 [D5.3] – Section 4.2.3 provides more details about our proposed local-aware name-based routing scheme and the IRTF draft [TRUO15b]. We enable a centralized (SDN-based) controller to dynamically configure the NDN forwarding tables for local-aware routing, based on the social interactions in the OSN (e.g. add route in the local NDN routers if close friends/followers are on-line, remove it if not).

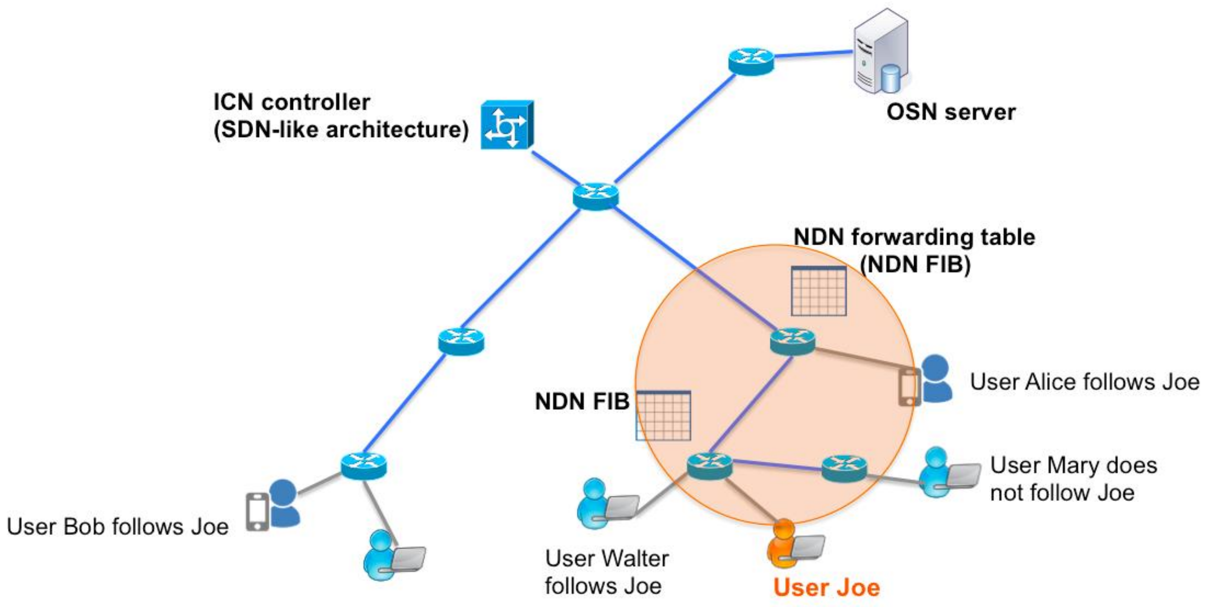


Figure 3: Network topology for functional tests IN1, IN2, IN3 and IN6

For naming end-users and their contents (text messages, videos or photos), we suggest the following hierarchical naming:/OSNapp/userID/contentID

2.3.1.1 *Publication of a Content (text message, photo or video)*

The Figure 4 illustrates the chronological workflow when the user Joe wants to publish a content, let us say for example a video.

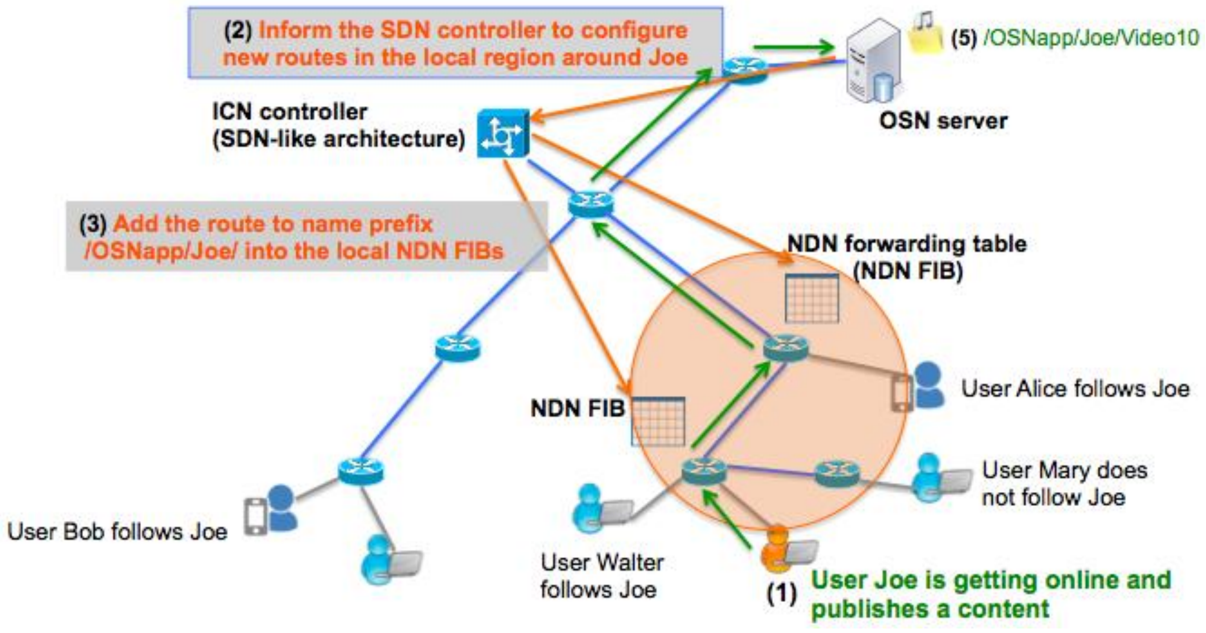

(2) Inform the SDN controller to configure new routes in the local region around Joe
(5) /OSNapp/Joe/Video10
ICN controller (SDN-like architecture)
OSN server
(3) Add the route to name prefix /OSNapp/Joe/ into the local NDN FIBs
NDN forwarding table (NDN FIB)
User Alice follows Joe
NDN FIB
User Mary does not follow Joe
User Bob follows Joe
User Walter follows Joe
(1) User Joe is getting online and publishes a content

Figure 4: Publication of content

1. Joe is getting online: he connects to the OSN server using the application client on his smartphone or computer. Joe wants to publish a content, named /OSNapp/Joe/video10: the content object will be stored in the OSN server.
2. The OSN server informs the ICN controller that Joe has published a content. The server then sends to the controller a notification message containing information related to Joe:
    a. the location (NDN name) of the NDN router to which Joe is connected;
    b. the list of Joe's friends/followers who are currently online, along with their location (i.e. the name of the NDN routers to which Joe's online friends are connected).

(Note that when one user is getting online, the server notifies the controller for routing updates only if the user publishes a content.)

3. The ICN controller configures the local NDN routers (located at most 2 routing hops from Joe) by adding the route "/OSNapp/Joe" for Joe reachability in the related FIBs. This route is added to a local NDN router only if there are Joe's online friends attached to the router.

2.3.1.2 *Local Routing for Retrieving Content from Local Users*

We illustrate here how local users (2 routing hops far away from Joe) can get content directly from Joe, instead from the OSN server, leading to optimized and improved content delivery.

We suppose that the user Alice wants to get the content "/OSNapp/Joe/Video10" that Joe has just published (see Figure 5).

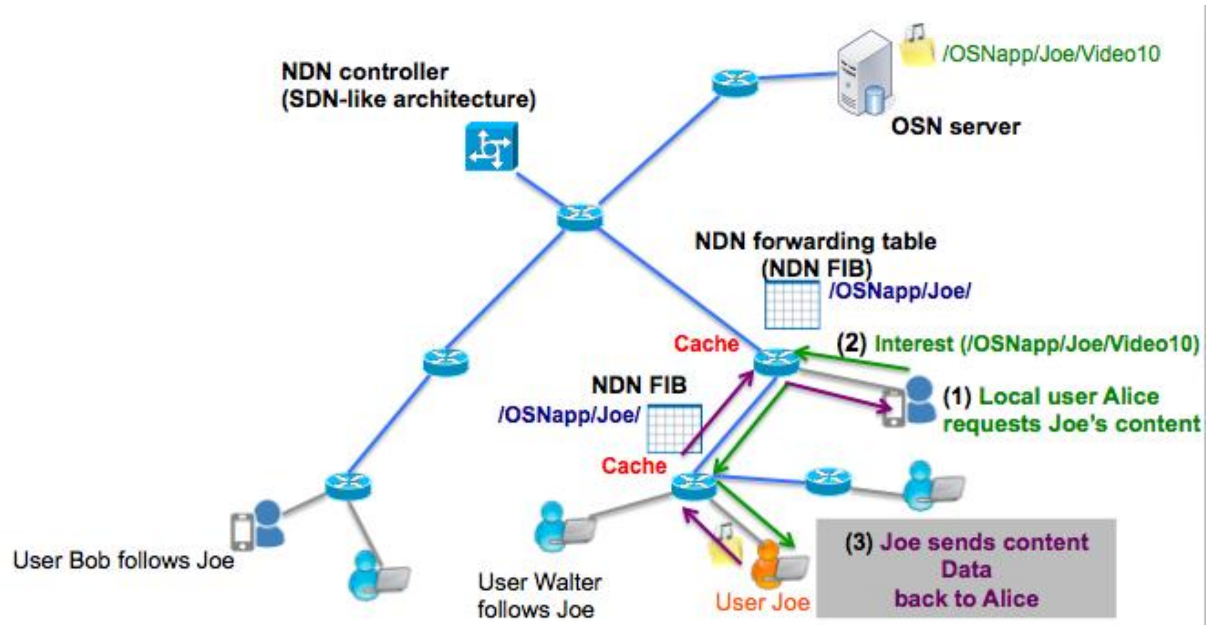


Figure 5: Local routing for retrieving content from local users

1. Local user Alice wants to retrieve Joe's content"/OSNapp/Joe/Video10". The application client on Alice device notifies the OSN server with information (Joe,video10) to let the OSN server know that a content from Bob is about to be consumed. Alice then express an Interest for "/OSNapp/Joe/Video10".
2. Thanks to the previous routing configuration by the ICN controller, the Interest("/OSNapp/Joe/Video10") will be routed to Joe.
3. Joe's OSN app client, receiving the Interest, issues a Data message to Alice for serving the requested content Video10.
4. While forwarding the Data message on the reserve path (taken by the Interest), Joe's content is cached into the Content Store of all the traversed NDN routers. Note that caching is social-aware, meaning that a NDN router keeps the content in the cache only if there are more than N users potentially interested by the content.
5. If now Walter wants the same content, he will get it from the cache

2.3.1.3 *Using OSN Server for Retrieving Content from Non-Local Users*

This scenario, as illustrated by shows that non-local users will be served by the OSN server.

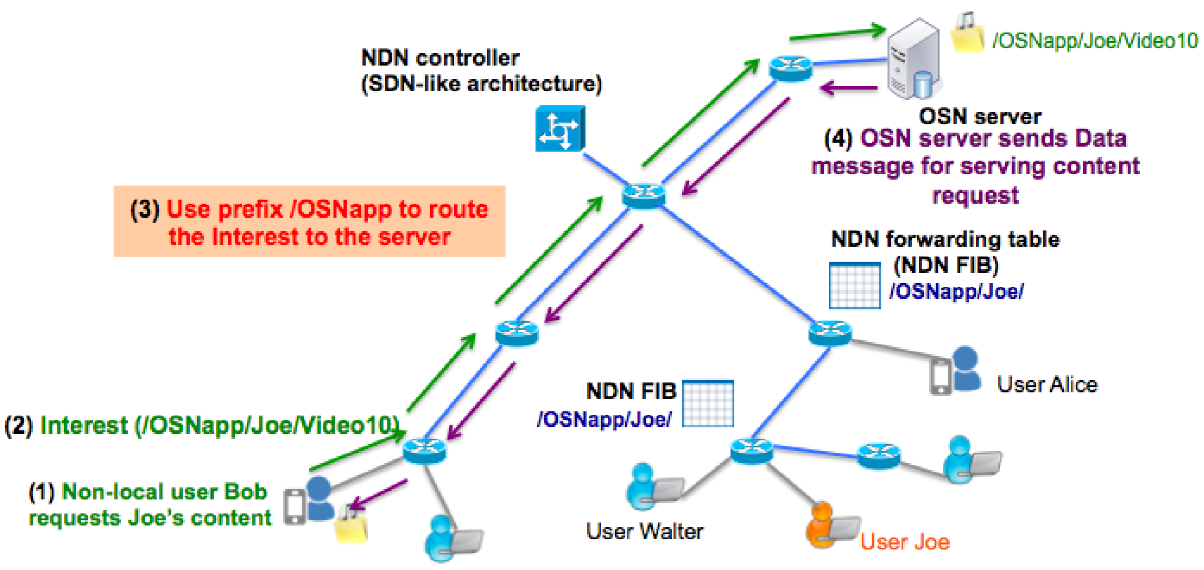

NDN controller
(SDN-like architecture)
/OSNapp/Joe/Video10
OSN server
(4) OSN server sends Data message for serving content request
(3) Use prefix /OSNapp to route the Interest to the server
NDN forwarding table
(NDN FIB)
/OSNapp/Joe/
User Alice
NDN FIB
/OSNapp/Joe/
(2) Interest (/OSNapp/Joe/Video10)
(1) Non-local user Bob requests Joe's content
User Walter
User Joe

Figure 6: Using OSN server for retrieving content from non-local users

1. Non-local user Bob wants to retrieve Joe's content. He sends an Interest for "/OSNapp/Joe/Video10".
2. As Bob is a non-local user, i.e. located too far from Joe, his Interest will be forwarded using the default route "/OSNapp".
3. The NDN routers forward Bob's Interest using the prefix "/OSNapp".
4. The OSN server is responsible for serving the request:
    a. The sent content will be cached in the content store of the different traversed NDN routers on the reverse path, based on our social-aware caching feature.

2.3.2 Analytical Evaluation of the Locality-Aware NDN-Based Content Delivery

We define in this section the model we use for deriving our analytical evaluation for our local-aware NDN-based routing scheme for Twitter-like applications. We consider the network topology for modelling an ISP network, as represented in Figure 7:

3 network levels;

N end-users requesting contents of same size s in bytes;

W denotes the ratio of users' requests that can be served locally in the network. This means that among the N requests, we simulate different proportions W (in percentage) for requests that can be locally served due to our local-aware routing scheme;

Local regions for local routing are set to t routing hops. (we consider $t = 2$) ;

At a same network level i, all the NDN routers have a same cache miss probability p_i and a same cache size C_i. It is common to consider that the cache size is larger in the upper layers which have to convey a larger part of traffic flows downwards, so we will assume that $C_3 > C_2 > C_1$ for the 3-level.

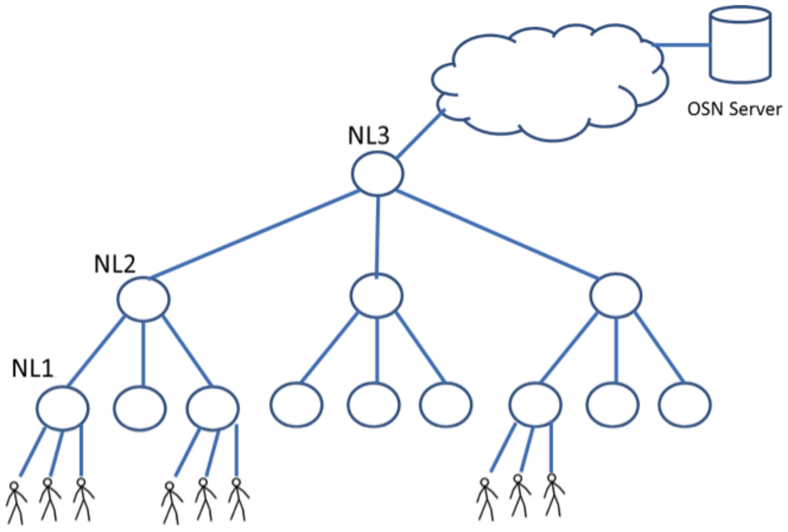


Figure 7: 3-level hierarchical network topology (modelling an ISP network)

In the stationary regime, an Interest for a tweet that cannot be satisfied in the first layer of caches will be forwarded to the upper layer, and if the queried tweet cannot be found in either layer, the request will be directed towards the OSN server located outside the network. Any new tweet retrieved from the OSN server will then be inserted in all the cache levels along the network path at the expense of another tweet, which is thus evicted from those caches.

A detailed analysis of Twitter users performed by [STTW] reveals that an overwhelming part of users have less than 50 followers and only a tiny amount of users have more than 100,000 followers. A Twitter user on average follows 102 people and 10% of users do not follow anyone. The distribution of follower counts is actually a very long tail distribution. Based on those observations, we assume that the popularity of tweets is equivalent to the follower count distribution. Given a tweet m written by some user X, we denote by q_m the ratio of users following this tweet (i.e. following X) to the total number of Twitter users. We then consider the quantity q_m as the popularity of the tweet m. We also extend the notion of tweet to any user-generated content that can be a short text message, a video, a picture, or both texts and multimedia content. It has been observed that the popularity of user-generated content generally follows a Zipf distribution with an exponent $a < 0.8$ ([BRESL99], [MAHA00] and [CHE02]). As a consequence, we suppose in our model that the popularity of tweets (i.e. q_m) satisfies a Zipf law as well.

All tweets are enumerated with decreasing popularity order, so q_1 and q_N indicate the ratio of requests for the tweet with the highest and lowest popularities respectively. We adopt in our model the Independent Reference Model (IRM), which means that all the random variables denoting requests for tweets are independent and identically distributed, or in other words, the probability that a request is for a tweet depends only on the tweet popularity and not on the order of request arrivals.

We aim at evaluating the average load on the OSN origin server, denoted by l, the average routing hop count (referred to as r) for serving a content request, the average volume v_t of traffic transiting in the network, and the average volume v_p of peering traffic (i.e. traffic going outside the ISP network - the ISP would save transit cost if it could keep control of this traffic inside its network).

The following table lists the different notations used in our model.

Notation	Meaning
N	Total amount of OSN end-users requesting tweets of same size s in bytes

s	Average size of tweets in bytes. (The notion of tweet is extended to any user-generated content that can be a short text message, a video, a picture, or both texts and multimedia content)
q_m	The probability of tweet m to be requested, or equivalently the popularity count of tweet m (with respect to the IRM framework)
$0 < W < 1$	Ratio of end-users' requests that can be locally served due to our local-aware routing scheme.
t=1	Local regions for our local name-based routing scheme are delimited by t routing hops. (we will set t=1)
C_i	Cache size of a router at the network level i
p_i	The miss probability of a router cache at the network level i
l	Average load on the OSN server
r	Average routing hop count for satisfying a request
v_t	Average volume of traffic transited in the network
v_p	Average volume of peering traffic

2.3.2.1 *Average Load on the OSN Server*

The average load on the origin server is measured by the ratio of requests served directly by the OSN server. A tweet is retrieved from the OSN origin server only if the request cannot be satisfied in either cache levels of the network, i.e. there is a cache miss in each traversed NDN router at all the network levels. Therefore the average load on the server is defined by the following formula:

$$l = p_1 p_2 p_3 \qquad (1)$$

When considering our locality-aware routing proposal, we can improve the load on the origin server since requests from local users can be locally satisfied within the different network levels thanks to local prefix name announcements from end-users. So the average load on the OSN server with our proposed local routing scheme is then:

$$l = p_1 p_2 p_3 (1 - W) \qquad (2)$$

2.3.2.2 *Average Routing Hop Count*

The average routing hop count denotes the average number of NDN routers an interest for a tweet needs to traverse before getting the requested tweet (either with a cache hit in the content store of a NDN router, or at the OSN origin server for the worse case).

The formula to calculate the average routing hop count is straightforward:

$$r = 3p_1 p_2 p_3 + 2p_1 p_2 (1 - p_3) + p_1 (1 - p_2) \qquad (3)$$

This formula actually reflects the effect of caching in NDN within the different network levels. The first term $3p_1p_2p_3$ represents the ratio of all requests served by the OSN server after getting a cache miss at all the cache levels: the routing hop count for those requests is then the maximum value 3.

The second term $2p_1p_2(1-p_3)$ (respectively $p_1(1-p_2)$) represents the ratio of all requests that get a cache hit in one NDN router located at the third (respectively second) network level, so that the routing hop count for those requests is 2 (respectively 1). Note that the requests that get a cache hit in the first network level have no routing hop, so they are not counted into the formula (3).

After simplifying the formula (3), the average routing hop count when using NDN is finally expressed as follows:

$$r = p_1 + p_1p_2 + p_1p_2p_3 \tag{4}$$

This alternative formula for the routing hop count means that each cache miss at a network level contributes to increment the hop count by one. In particular, when using IP as the content delivery network, the routing hop count is 3, which corresponds to the case when all the miss probabilities p_i are equal to 1.

The formula (4) can be refined with our proposed locality-aware routing strategy. We consider local network regions of $t = 1$ routing hop, which means that the ICN controller will only configure the FIBs of first-level NDN routers when advertising the name prefixes of end-users. Followers under the same first-level router can then benefit from those local prefix advertisements.

When there is a cache miss at the first network level, we make a distinction between local and non-local users. Requests from local users thanks to local name prefix advertisements will be forwarded downwards the publisher under the same first-level router (and never get the second network level). Only non-local users, which amounts to the quantity (1 – W), need to traverse the upper levels. Our average routing hop count formula with local routing then becomes:

$$\begin{aligned} r &= p_1(1-W) + p_1p_2(1-W) + p_1p_2p_3\,(1-W) \\ &= (p_1 + p_1p_2 + p_1p_2p_3)(1-W) \end{aligned} \tag{5}$$

2.3.2.3 *Average Traffic Volume*

For evaluating the traffic generated by our Twitter-like application, we separate the peering traffic from the local traffic.

The peering traffic is the traffic going outside the ISP network, generated by the requests that cannot be satisfied neither by the caches in the NDN network nor by end-users themselves using our local routing scheme (therefore the server can only serve those requests). The peering traffic, expressed in terms of bytes, is then defined by:

$$v_p = p_1p_2p_3(1-W)sN \tag{6}$$

The local traffic corresponds to the requests that are served either by the NDN caches or directly by users thanks to our local routing scheme. With local end-user name prefix announcements at the first network level, the local traffic is given by:

$$v_t = rsN - 3p_1p_2p_3(1-W)sN \tag{7}$$

$$= (p_1 + p_1p_2 - 2p_1p_2p_3)(1-W)sN \tag{8}$$

Eq. (7) describing the local traffic is derived from the following observation. The first term, rsN, corresponds to the total overall traffic related to the Twitter application, from which we need to remove the part of the traffic that correspond to the peering traffic traversing the 3 levels of the network.

2.3.2.4 *Numerical Results from the Analytical Evaluation*

We present in this section the main results of our evaluation, based on the model presented just before.

We consider the example of Twitter traffic in France: 8 millions tweets are published every day; the popular users worldwide are also popular in France; the average tweet size is 30 bytes (except when we made this parameter varying), the content popularity follows a Zipf law with alpha=0,8 (except when we made this parameter varying), the locality ratio W of users' requests that can be locally served is 0 (except when we made this parameter varying), some specific caches size for the different levels (for small size to larger size).

For all the tests, our objective was to evaluate the number of hops (for the data to be transmitted toward the requesting client) and the peering traffic (traffic coming from the Twitter Server) with varying parameters such as the size of the tweet, the cache size at the 3 levels, the content popularity (parameter α of the Zipf law), the locality of end-users (end-users being in the same/close location than the publisher of the tweet - parameter W in our analytical model).

2.3.2.4.1 Impact of the Size of a Tweet

We performed two sets of tests: One for tweet size in the range of 20B to 140B, which is representative of the current Twitter behaviour, and one for tweet size from 30B to 1 MB, which can represent an evolution of Twitter, exchanging not just short text messages, but also pictures (such as Snapchat). For each, we performed with 4 different configurations of the cache size: from small-size caches to large-size-caches.

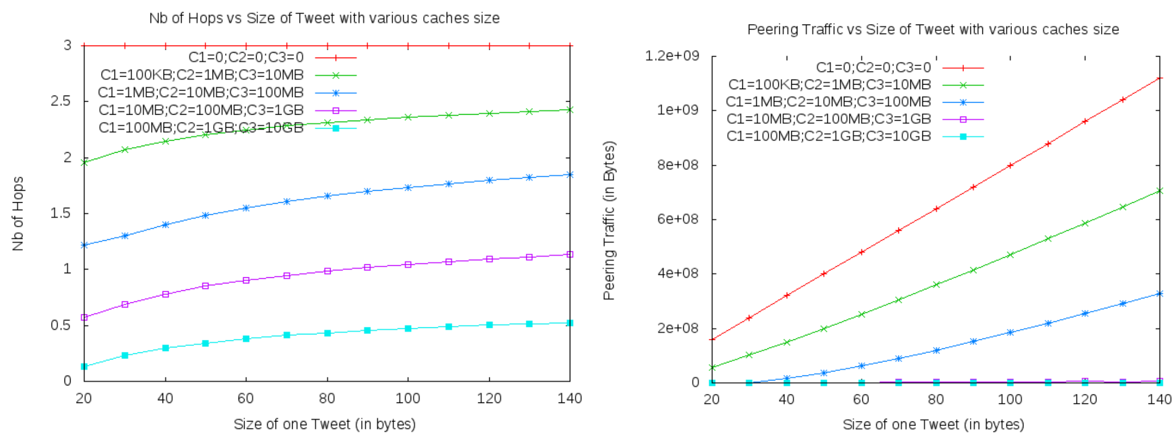


Figure 8: Impact of tweet size (20-140 bytes)

From the size of tweet from 20 to 140 bytes (Figure 8), we can see that the size of the caches largely impacts the number of hops and the peering traffic (more data can be cached, less to retrieve from the Twitter Server). We can also see that with the 2 configurations with large caches, all data can be cached and then no need to get data from the server.

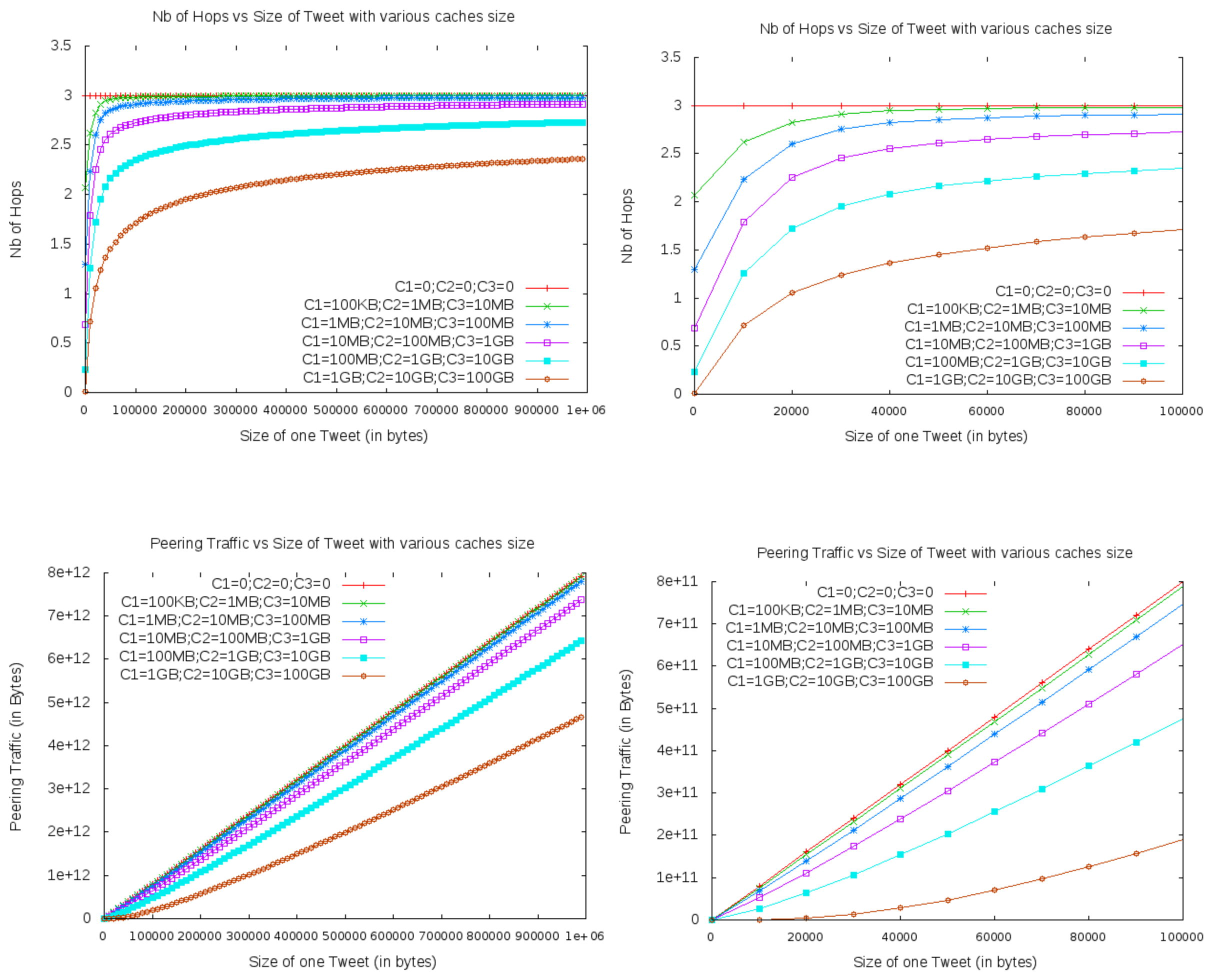


Figure 9: Impact of tweet size (30B - 1MB)

For the size of tweet growing up to 1MB (Figure 9), we can see that only large cache configurations can reduce the number of hops, but only tweets having a size less than 200KB can result in hops lower than 2. If we consider all tweets with 1 MB (each tweet is a photo), the cache configuration is not enough to reduce the number of hops (except the biggest one which can provide better results). It means that if a very huge amount of exchanged tweets are large (photo or snapchat), the caches of the NDN nodes should be very large in order to be efficient. Or if not possible, we have to define an alternative, such as a cooperation between NDN and CDN-like network.

2.3.2.4.2 Impact of Locality

We performed several tests to evaluate the impact of the locality of end-users (followers end-users being in the same region than the publisher of the tweet) and with 2 cache size configurations (the 2 lower ones), for the 2 tweet sizes (30B or 10KB). We also evaluated the scenario when there is no cache (current IP-based delivery) for measure the gain we can obtain via a local-aware delivery of data.

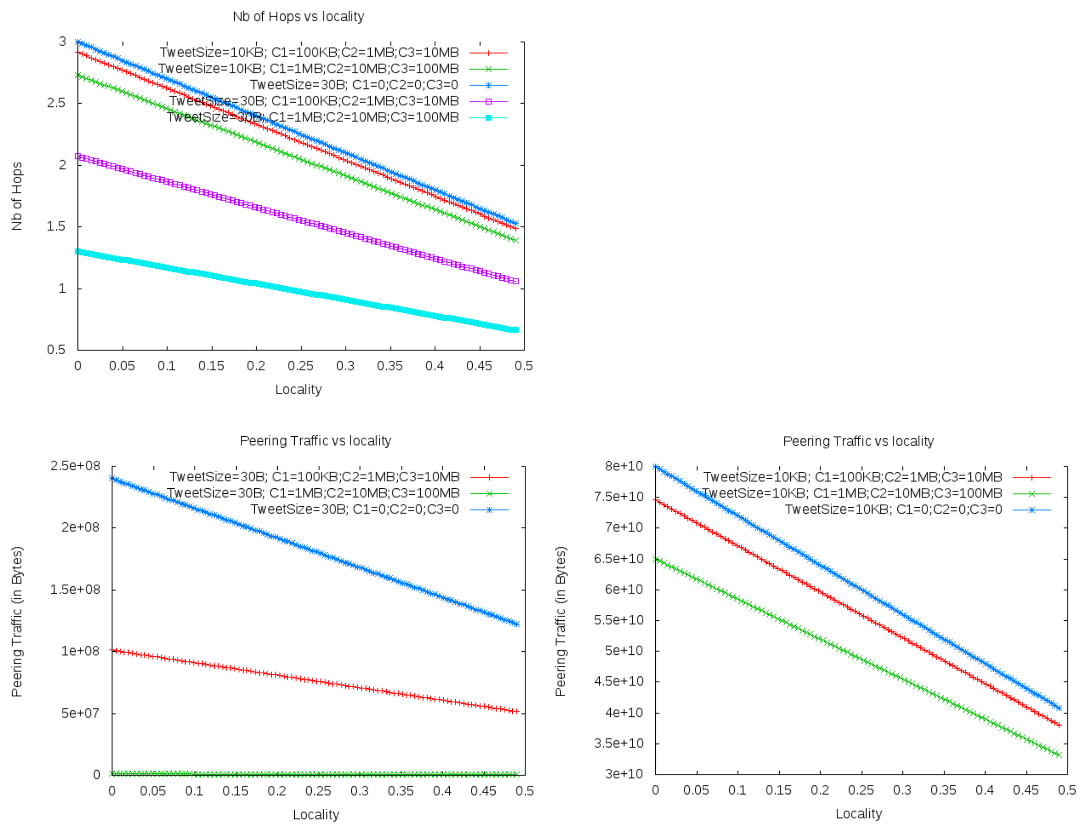
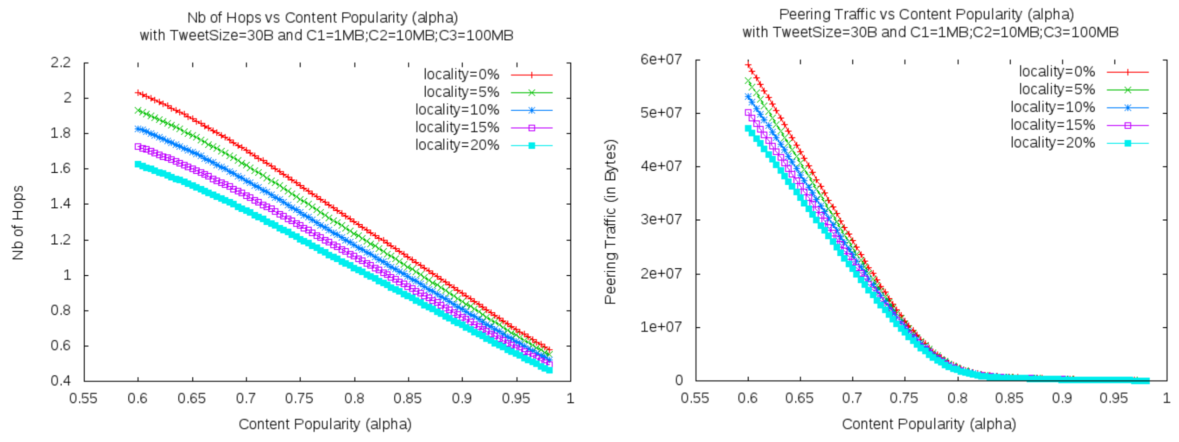


Figure 10: Impact of Locality

On Figure 10, we can detect that the locality has a huge impact on the delivery since the number of hops can be largely reduced (up to 60%), also when no cache is used. But the cache function is very important since it allows to deliver tweets with a 10KB size with a better quality then the current Twitter like system for only 30B-size tweet. The peering traffic also logically reduces as the locality increases (more data are retrieved for the local publisher instead of being retrieved from the remote server).

2.3.2.4.3 Impact of the Content Popularity and Locality

We performed several tests to evaluate the impact of the content popularity (parameter α of the Zipf law varying from 0.6 to 0.99), with 5 different configurations for the locality (0%, 5%, 10%, 15%, and 20% to end-users following the publisher being in the same region), with 2 different sizes for the tweets (30B or 10KB), and 3 different caches size configurations.



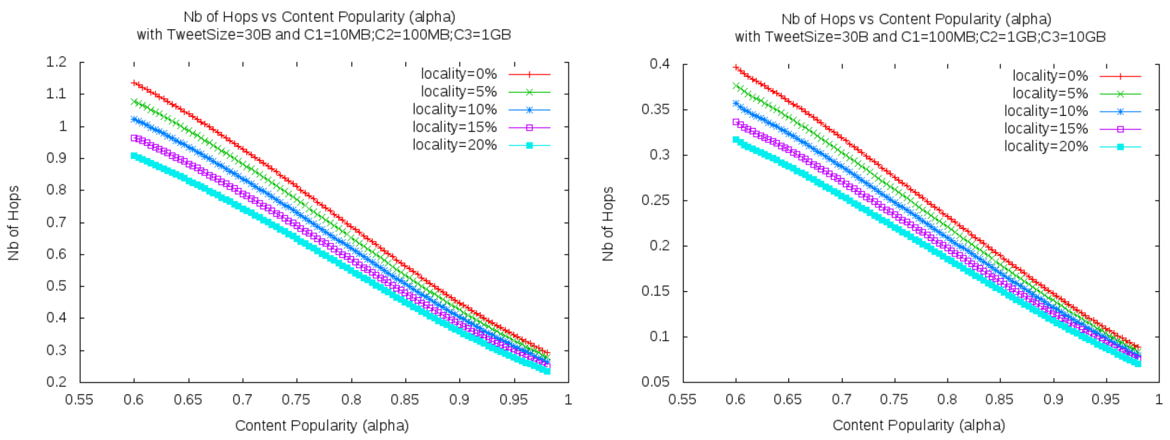
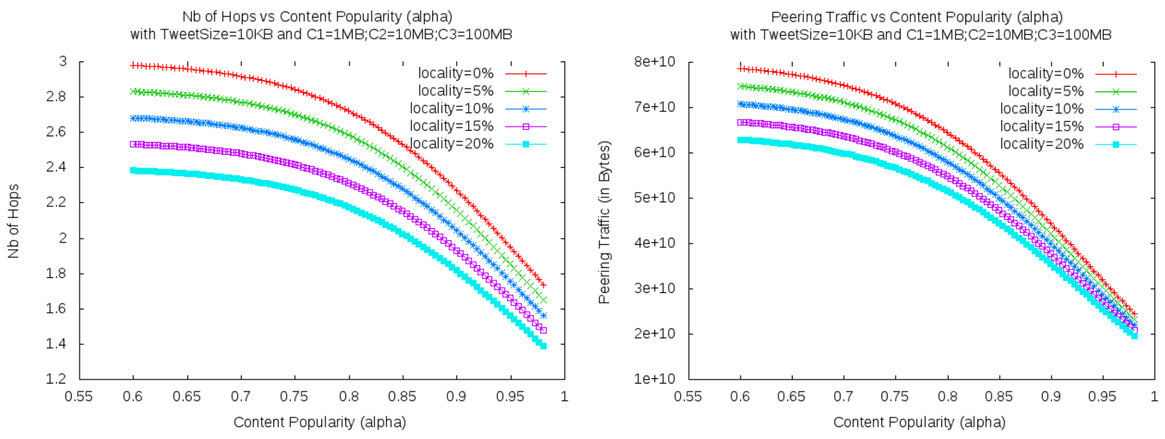


Figure 11: Impact of content popularity and locality (tweet size = 30B)

We can see that the locality aspect (Figure 11) has a big impact for the delivery of data since the number of hops can be reduced up to 25% (in case of 20% locality) for non very popular contents (α =0.6). If the content is very popular (α close to 1), the caches are more efficient (cached data provided to much more end-users) and thus the locality parameter has a less important impact. We can also see that the popularity of content largely affects the delivery since the number of hops is largely reduced for popular contents, compared to less popular ones. Also, the number of hops largely reduces when the cache size increases (not shown here due to space limitations).

Concerning the peering traffic, we can observe the peering traffic drastically reduces with the popularity increase (between α =0.6 and α =0.8) to go down close to 0, meaning that almost all the data can be cached in the nodes. For the larger cache size configurations, this traffic is always close to 0, since the cache sizes allow to cache almost all the data.



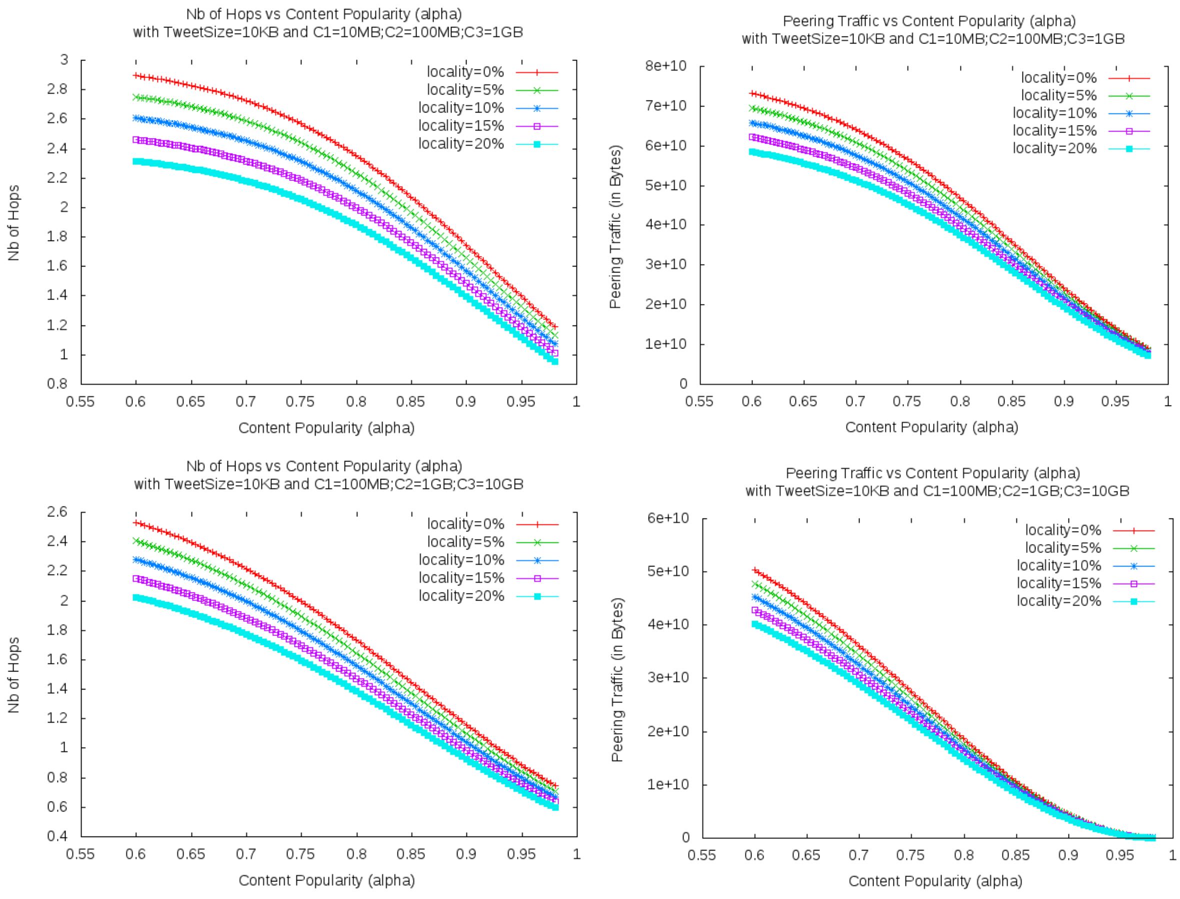


Figure 12: Impact of content popularity and locality (tweet size = 10 KB)

Similarly to the previous case, we can observe in Figure 12 that for tweets with a larger size (10 KB), the locality also largely impacts the delivery quality (number of hops reduced by 30% for α =0.6) and its impact is more limited as the content popularity increases.

The peering traffic also reduces as the locality increases but its impact becomes limited for very popular contents (even if large cache sizes are deployed).

2.3.2.4.4 Impact of the Cache Size at Each Level

In this set of tests, we wanted to evaluate the impact of the cache size and mainly if it would be better to increase the cache size for low level caches (Cache Level 1) or higher level caches (Cache Level 2 or Cache Level 3). To this end, we made the cache size parameter for each level varying, whereas the other cache sizes have the same value; the default values for the cache size are: C1=10MB; C2=100MB; C3=1GB, and the medium configuration for our tests.

Tests were performed for a tweet size of 30B and 10KB.

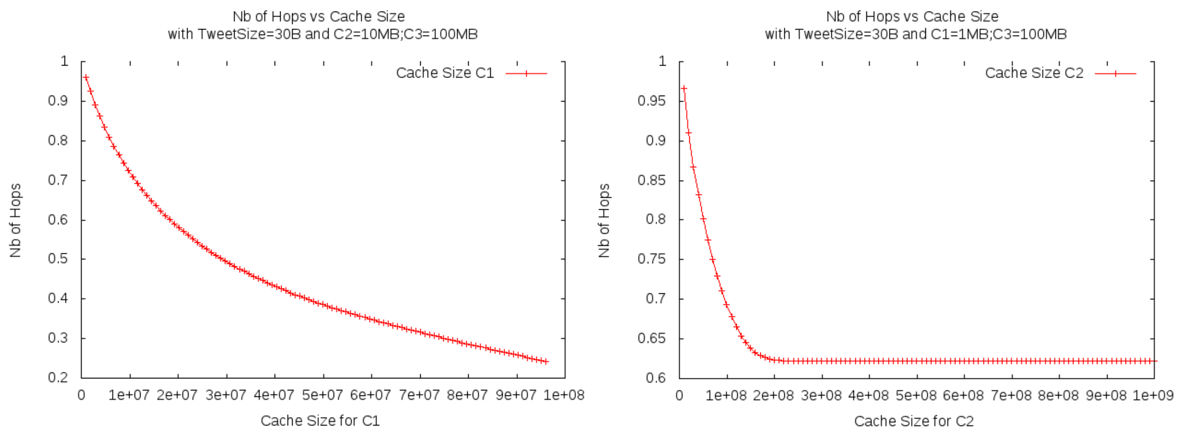


Figure 13: Impact of the cache size

On Figure 13, we can see that it is largely preferable to increase the size of caches at low level (Cache 1), than the Cache 2. Indeed, the number of hops can be reduced down to 0.25 hops when the cache size for C1 is 100MB. When the cache size for C2 increases, the number of hops quickly reduces, but it reaches a minimum at 0.62 and cannot further decrease (because caches from level 1 are full and remaining data should be cached at level 2).

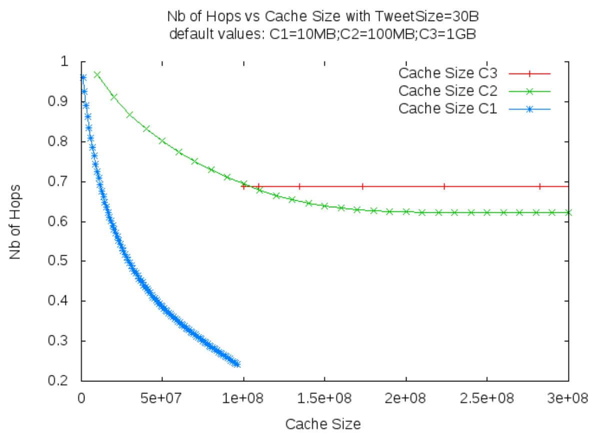


Figure 14: Increasing cache sizes with tweet size=30B (e.g. we fix C2=100MB and C3=1GB to default values, and we make C1 varying)

In Figure 14, we traced the increase of each cache size at different levels (C1, C2 and C3) on the same figure to evaluate the impact of the increase of each level of cache. As seen previously, it is preferable to increase the size of cache at lower levels.

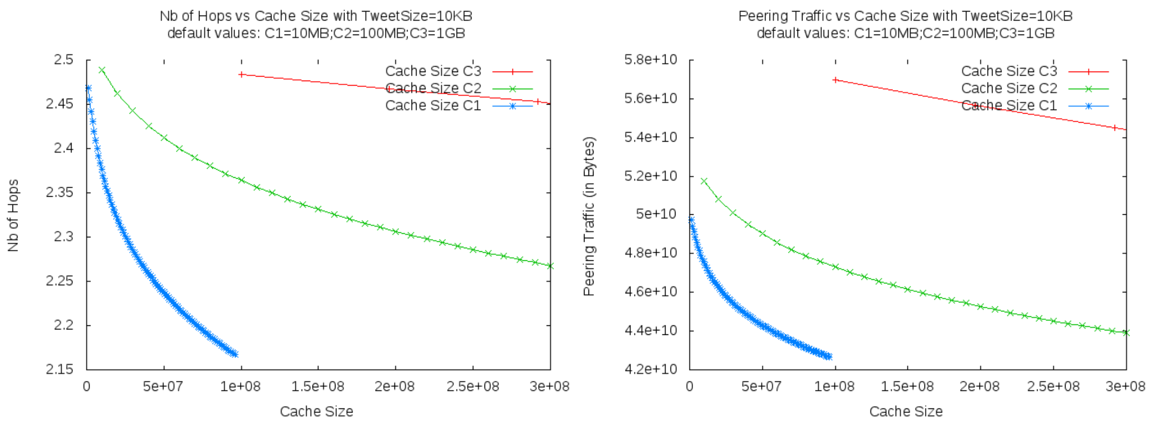


Figure 15: Increasing cache sizes with tweet size=10KB

We performed the same tests for a tweet size of 10KB (Figure 15). In this case, the caches are not able to cache all the data, thus a large part of them are retrieved from the OSN server: this is why we have so much peering traffic and a high value for the number of hops. But as for the previous test, we can see that it is better to increase first the size of caches at low levels.

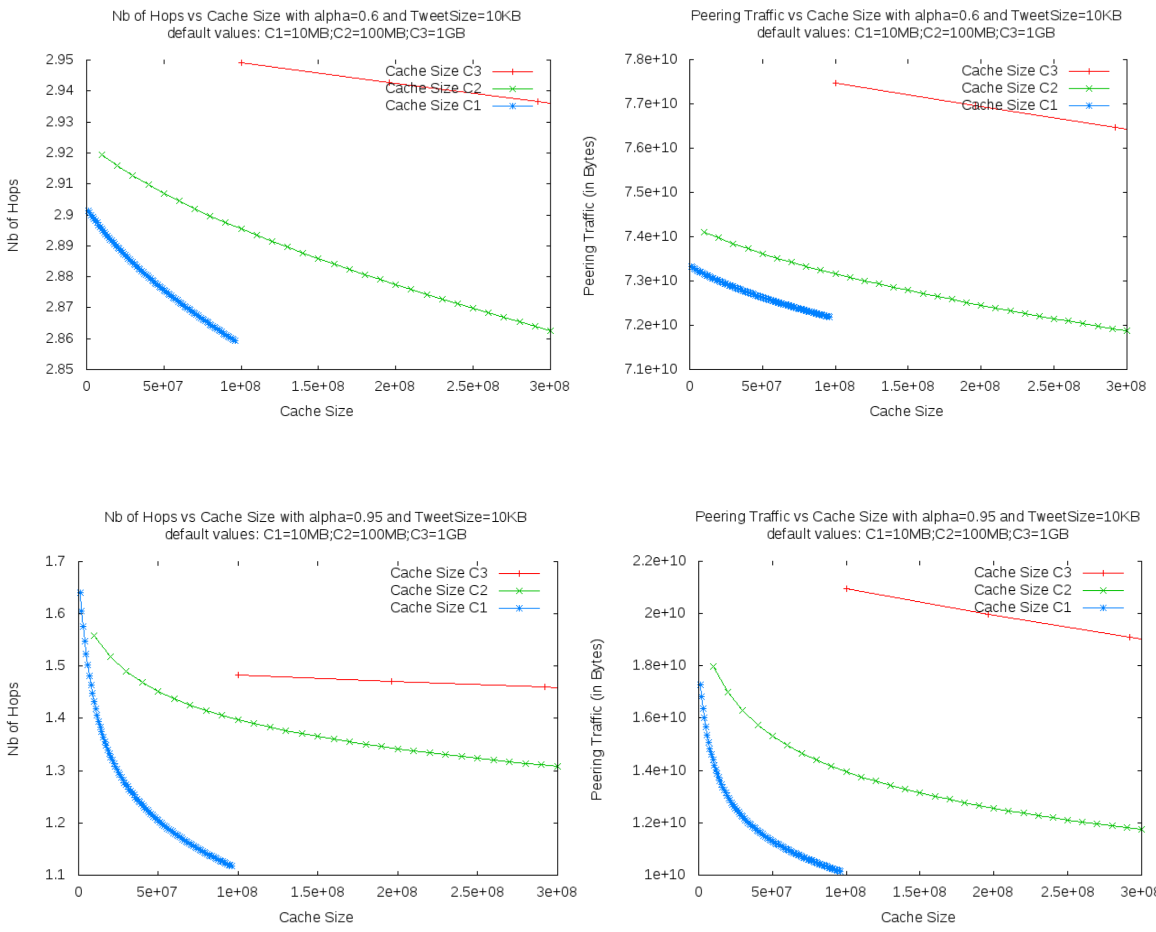


Figure 16: Impact of cache sizes with regard to content popularity

We also wanted to identify if the content popularity has an impact on this conclusion. We then did the same test with α=0.6 and α=0.95 (Figure 16).

We observed that the curves of the figures are different: the reduction is more important as the contents are popular and we can save much more peering traffic (reduction of 70% of the peering traffic between α =0.6 and α=0.95). But the same conclusion can be applied: it is better to increase the caches at low levels before increasing caches at higher levels.

2.3.2.4.5 Conclusions of the Evaluation

The set of tests we have performed allows us to say that locality plays an important role in the quality for delivering the tweets, with a large reduction of the number of hops, as well as the peering traffic, which is very important for a network operator to reduce transit costs. This is less important for very popular contents but since Twitter includes many various end-users, we still have a diverse variety of contents. We also analysed that the size of caches is important and it is better to increase first the size of caches being lower in the network (closer to end-users) instead of increasing upper caches (even if a higher caching capacity). This is particularly true for the current Twitter behaviour, where end-users send tweets with an average length of 30 bytes. But if the Twitter behaviour changes and allows to exchange richer messages (e.g. photos) with a larger size, the network caching capacities will need to be increased so as to be able to deliver the content with the same quality. But the locality can still provide a good alternative at low cost for a network operator, since it allows to improve the delivered quality with no network investment.

2.3.3 Scenarios for tests IN4 and IN5

2.3.3.1 *IN4: Route Calculations on Rocketfuel-based Topologies*

The aim of the performed test was to evaluate the potential benefit provided through the ICN controller based information-centric networking concept in the scenarios described in section 2.3.1. In the following, the underlying assumption for our simulative evaluations and route calculation used by the ICN controller is that the OSN server provides complete information about all available prefixes and related user locations to the ICN controller employing the OSN configuration interface described in section 2.3.3.2.

2.3.3.1.1 Test topology construction

For our simulative evaluation of the concept have created an analytical model to investigate the influence of network topology properties and parameters on the route properties to be calculated by the ICN controller we used a simple analytical model. For our evaluation we chose modified versions of different PoP level Rocketfuel–mapped topologies:

a) AT&T network topology based on AS7018, consisting of 115 nodes and 148 edges
b) Sprint network topology based on AS1239, consisting of 52 nodes and 84 edges

We applied simple heuristics to separate the Rocketfuel topology nodes into different categories: clients, gateways, and backbone nodes:

- All nodes with a degree of one were assigned to the “client” category
- All nodes with a degree of two were assigned to the “gateway” category
- The rest of the nodes were assigned to the “backbone” node category.

After this node categorisation, a number of user nodes are attached to a subset of client nodes.

A sample Rocketfuel-based topology which has been used as starting point in our simulative evaluation of the concept is shown in Figure 17.

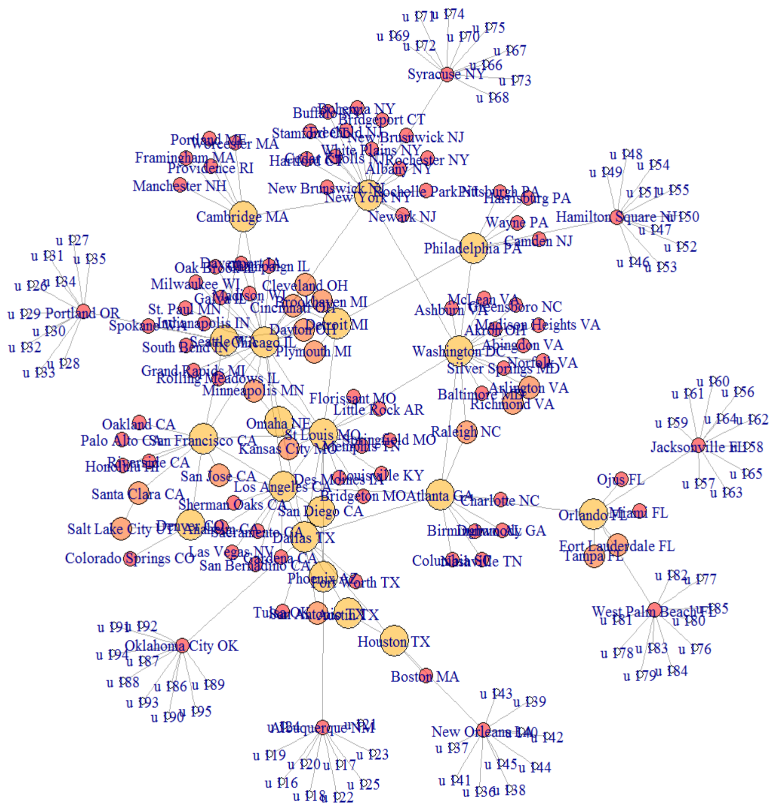


Figure 17: Sample Rocketfuel-based topology.

2.3.3.1.2 Evaluation of path lengths distribution

As a first step we randomly assigned the OSN server location to one of nodes of the backbone category. Next we calculated the set of paths originating from each of the user nodes to the OSN server node to define the baseline scenario, and calculated the statistics such as the overall path lengths distribution from the set. We made use of the igraph package [IGRA15] to perform shortest path calculations between the nodes in the on our test topology graphs using the same algorithm we use for the shortest path calculations on the internal topology graph structure of the ICN controller.

In a second step, a number of client users were randomly selected and assigned to a subset of “local” users which can serve as alternative source of the content which is normally served by the OSN server.

Subsequently we calculated shortest path between all user nodes and their closest source for the content, be it either the OSN server or one of the “local”-type user nodes. From the set of calculated set of shortest paths we finally derived the path length distribution, and the portion of paths still containing the OSN server as endpoint.

When increasing the portion of “local” users we observe a reduction of the measured mean path length distribution towards shorter lengths, meaning that a reduced number of hops is required to reach the content because our algorithm now can find alternative sources with a reduced hop count.

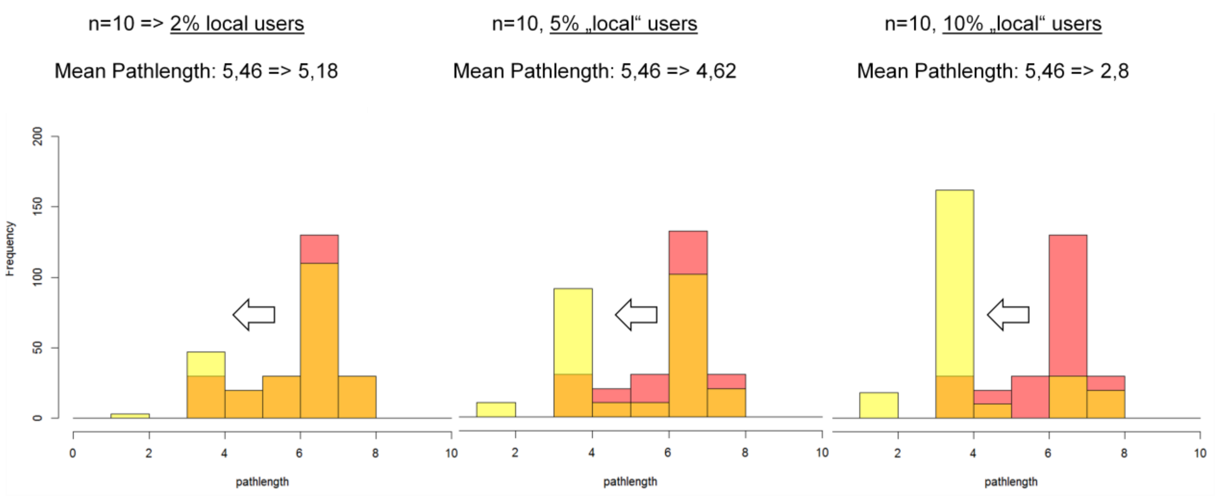


Figure 18: Calculated path length distribution for AS7018 (AT&T) Rocketfuel-based topology with 115 nodes and 148 edges, yellow distribution: with "local" users in the range of 2-10 %, red: no local users.

We have evaluated the mean path length over a range of 0-50% portion of "local"-type users on different Rocketfuel-based topologies and with a varying number of users attached to selected client nodes in the topology (Figure 19).

The mean path length reduction rapidly decreases with the number of attached user nodes, and with the local user ratio. A ratio of 20% is sufficient to reach a saturation level for the investigated parameter range. Further increasing the Local User Ratio does not result in a further decrease of the mean path length. Also the observed scaling behaviour of the mean path length appears to be independent of the specifics of the network topology.

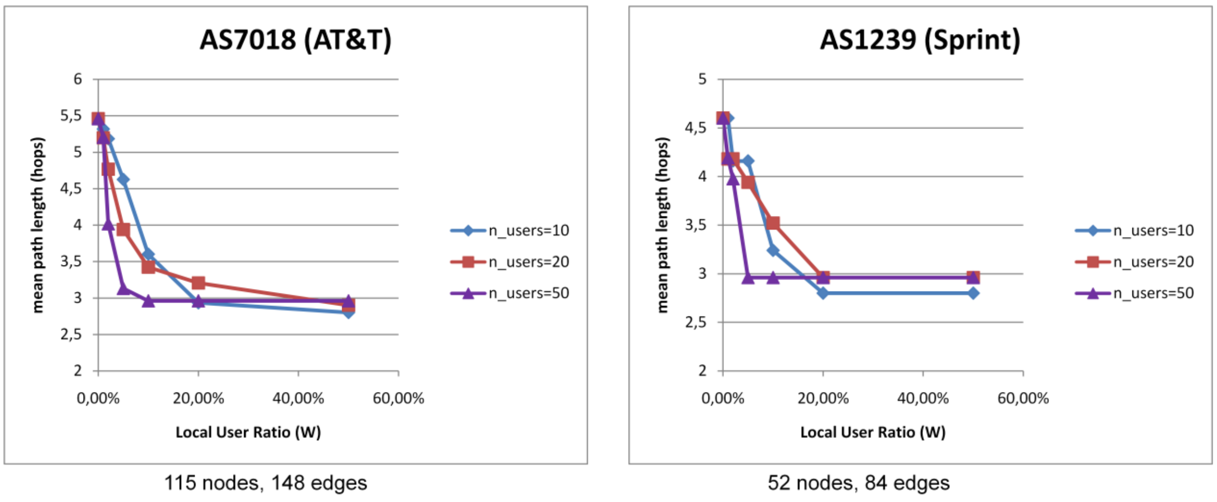
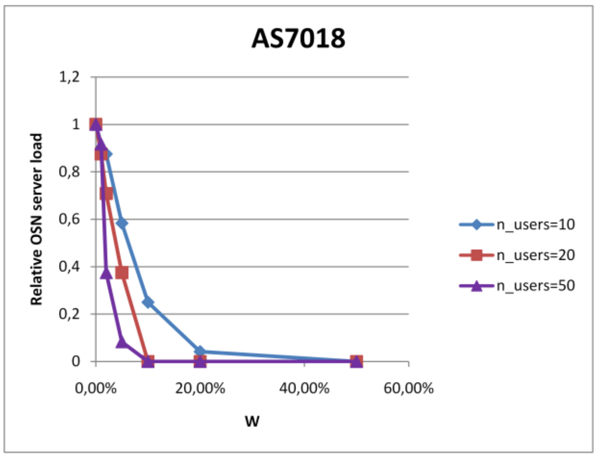


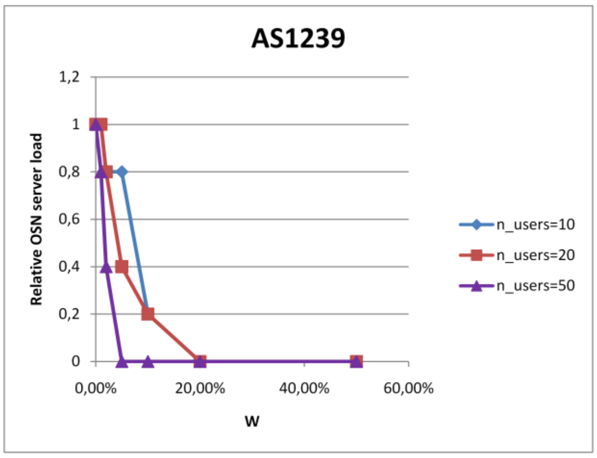
Figure 19: Calculated dependency of the path length distribution for AS7018 (AT&T) and AS1239 (Sprint) Rocketfuel-based topology on Local User Ratio.

2.3.3.1.3 Evaluation of OSN server load

Finally, the relative load on the OSN server with increasing Local User Ratio has been evaluated for both Rocketfuel-based topologies (Figure 20). The relative OSN server load is estimated from the portion of calculated paths with the OSN server as end point.

A similar saturation and scaling behaviour as with the mean path length is observed. For the topology parameters evaluated a ratio of 20 % of Local User Ratio is sufficient to completely offload the OSN server.





115 nodes, 148 edges

52 nodes, 84 edges

Figure 20: Calculated dependency of the Relative OSN server load for AS7018 (AT&T) and AS1239 (Sprint) Rocketfuel-based topologies on Local User Ratio.

2.3.3.1.4 Conclusions of the Evaluation on Rocketfuel-based topologies

The set of simulative evaluations we have performed on Rocketfuel-based topologies allows us to confirm that the portion of local users plays in important role. We could derive the dependency of the mean path length reduction on one hand, and residual OSN server load on the portion of local users on the other. We observe clear saturation effects both in mean path and relative OSN server load calculations.

In contrast to the dependency on Local User Ratio, in our evaluations the specific of the topology did not play an important role, which could indicate that the achieved benefits are largely topology-independent.

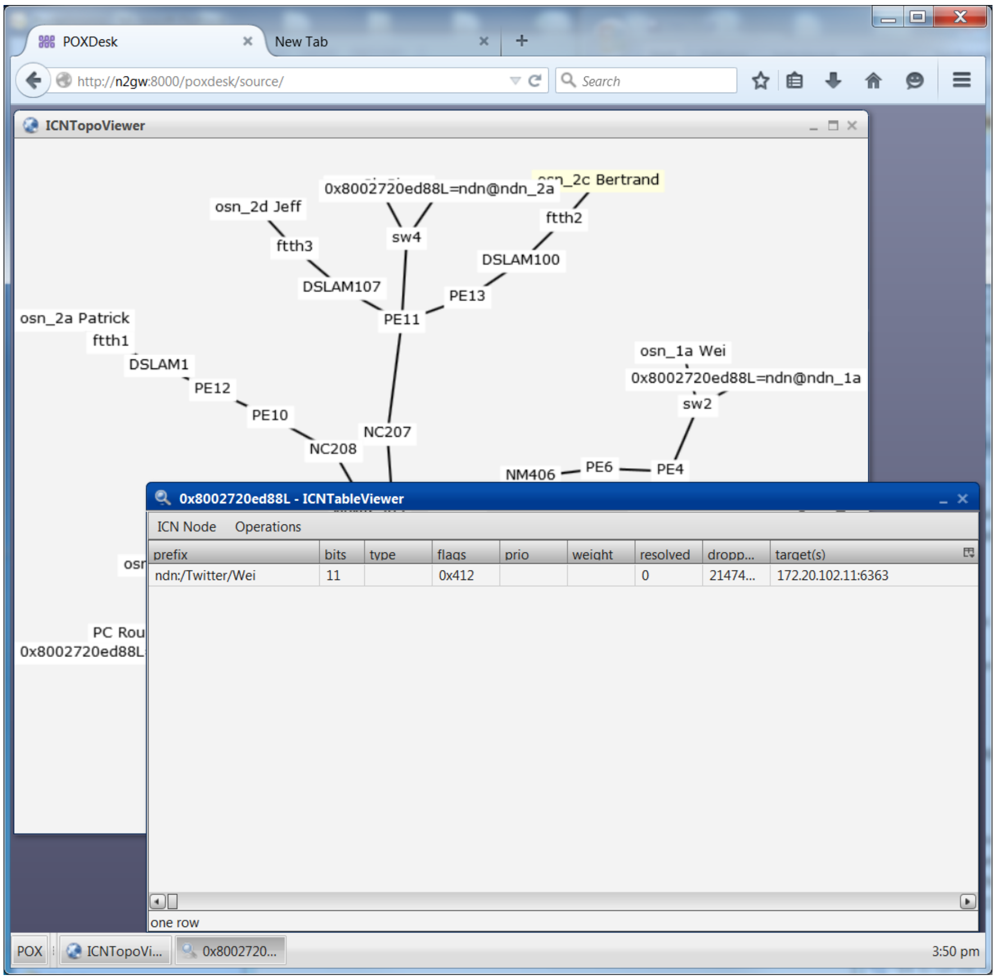
2.3.3.2 *IN5: Programming of forwarding elements*

2.3.3.2.1 Extended ICN controller test case

The ICN controller provides a graphical user interface which allows monitoring the topology and the content of the service tables (FIBs) of the registered forwarding nodes (see Figure 21). The GUI is provided at the local host and uses port 8000, so that the browser has to be pointed at:

```
http://localhost:8000/poxdesk/source/
```

The GUI also allows for monitoring of the content of the Forwarding Information Base (FIB) of registered NDN nodes as can be seen in Figure 21, showing in the ICNTableViewer that routing information for prefix “ndn:/Twitter/Wei” has been pushed from the ICN Controller to the registered NDN node with ID=0x8002720ed88L, which is also appearing on top of the ICNTopoViewer window.



prefix	bits	type	flags	prio	weight	resolved	dropp...	target(s)
ndn:/Twitter/Wei	11		0x412			0	21474...	172.20.102.11:6363

Figure 21: ICN controller GUI showing the ICNTopoViewer (showing the topology representation of the FT testbed) and ICNTableViewer windows.

2.3.3.2.2 ICN client test case

After GUI becoming available the controller now is ready to receive registrations from forwarding nodes. The client scripts realize a wrapper for the NDN ndndc configuration client which as to be installed an running on the forwarding node.

The ndn packet and python has to be installed and configured on the client together with some python packets such as socket, threading, json, time, uuid, sys, subprocess, urllib, xml, re, itertools, HTMLParser and netifaces (see python script for details). The following script will launch the NDN daemon ndndc on the client which will connect to the controller:

```
$ndndstart
$python ICN_NDN_client_v2.5.py &
```

The client can be launched with parameters e.g. giving the IP address and port to reach the ICN controller:

```
$python ICN_NDN_client_v2.5.py 10.0.2.3:7790 # modify IP address and port
of the ICN controller
```

2.3.3.2.3 Programming the ICN controller: the ICN REST interface

The existing ICN Controller REST interface has been extended so that it is possible to additionally configure the behaviour of the ICN controller with externally provided parameters. The respective modifications have been done in pox/openflow/webservice.py. The ICN rest interface also uses port 8000 and can be reached at the /ICN/ instead of /OF/.

In the following we describe the commands we have used to test the ICN Controller REST interface. For each of the commands we have checked the results returned by the ICN controller, either in form of the returned textual representation of the related dictionary structure, or via the resulting modifications in the ICNTopoViewer or ICNTableViewer windows of the ICN controller GUI.

Sample curl statement format:

```
$curl -I -X POST -d '{"method":"get_icn_nodes", "id":"1"}'
```

Returns the list of registered ICN nodes

```
$curl -i -X POST -d '{"method":"get_ICN_node_desc", "dpid":ICNID}}'
http://127.0.0.1:8000/ICN/
```

gets a number of ICN node parameters such as its ICNID, VendorID, ICN_FW_PROTOCOL, DPIP, and ICN_FW_PROTOCOL. Please note the used URL used to address the interface is

`http://127.0.0.1:8000/ICN/` instead of `http://127.0.0.1:8000/OF/`

```
curl -i -X POST -d '{"method":"get_icn_stats", "id":"1",
"params":{"dpid":ICNID}}' http://127.0.0.1:8000/ICN/
```

gets the stats (i.e., the ICN service table contents) of the ICN node ICNID (unique ICN identifier of the node).

```
curl -i -X POST -d '{"method":"set_icn_lb_ratio","id":"1" ,
"params":{"dpid":"1", "ratio":"1"}}' http://127.0.0.1:8000/ICN/
```

sets the load balancing ratio of for service router, tbd: add the service IDs:

```
curl -i -X POST -d '{"method":"set_icn_popularity_threshold", "id":"1"
"params":{"threshold":"999"}}' http://127.0.0.1:8000/ICN/
```

sets the popularity threshold for controller. If the follower number in a subsequent add_icn_meta_info request is below this threshold, it will be ignored.

```
curl -i -X POST -d '{"method":"add_icn_meta_info", "id":"1",
"params":{"contentid":"ndn:/Twitter/Bertrand", "followers":"1999",
"location":"region1"}}' http://127.0.0.1:8000/ICN/
```

adds meta info for a contentid, such as number of followers and location (IP) information for a specific content prefix/contented.

```
curl -i -X POST -d '{"method":"del_icn_meta_info", "id":"1",
"params":{"contentid":"ndn:/Twitter/Bertrand", "followers":"1999",
"location":"region1"}}' http://127.0.0.1:8000/ICN/
```

deletes the meta info for a contentid, such as number of followers and location (IP) information for a specific content prefix/contentid. There also are implemented some commands to query the ICN controller for routes on the ICN topology for debugging or evaluation purposes.

The ICN controller also implements a command to calculate routes between OSN user nodes and the OSN server:

```
curl -i -X POST -d '{"method":"calc_route", "id":"1",
"params":{"contentid":"ndn:/Twitter/Jeff", "start_node":"Bertrand"}}'
http://127.0.0.1:8000/ICN/
```

returns two routes: from the node start_node holding name prefix "Bertrand", to a) the registered OSN server, and b) the route to from start node to the node named "Jeff" to evaluate the gain by local-enhanced routing.

2.4 Content Offloading for Mobile Networks

2.4.1 Introduction to Test CO1-CO7

All components are integrated into one single software architecture, the Social Prefetching App ([D6.4], Section 2.4.4). The following tests assess different key components of this software architecture with programmatically executable tests written in Java with help of JUnit. Additionally, the interplay of the different components as well as the communication with Facebook is assessed by a scenario test. This scenario test allows us to identify failures and errors that are not only inherent to our software but also caused by changes Facebook or the Android system have made. This helps us to identify failures and errors which are raised due to a changing environment, e.g. by a new Android version or because Facebook changed their API or has a bug on their side. This allowed us, e.g. to quickly identify that the "Video Player" component was not ready for Android 5 previously. Furthermore, Facebook had a bug on their side during the user study, which was recently taken care of (https://developers.facebook.com/bugs/1540632892856427/, https://developers.facebook.com/bugs/449167695235902/https://developers.facebook.com/bugs/449167695235902/). The tests allows for identification of such bugs.

The test code can be found as source code in /app/src/androidTest/java/de.tudarmstadt.ps.mobilesocialprefetcher/tests/C0…. The following tests C01-C07 are software tests for our main contribution, the "Prefetching App". The tests R1 (Analysis of the Mobile Prefetching Study), R2(Analysis of D2D Traces), and R3 (Analysis of the Bandwidth Prediction Study) are performance tests for the conducted studies and give insights of the collected data and user contribution.

For the test involving Facebook, a set of Facebook test users is created automatically shown in Figure 22. First, the test users are created by using the Facebook Graph API. Second, the test users befriend each other so they can see content posted from their befriended test users. Third, a set of video posts is posted by test user 1. Now, the Social Monitor integration test is executed. More details are described in the corresponding test sections.

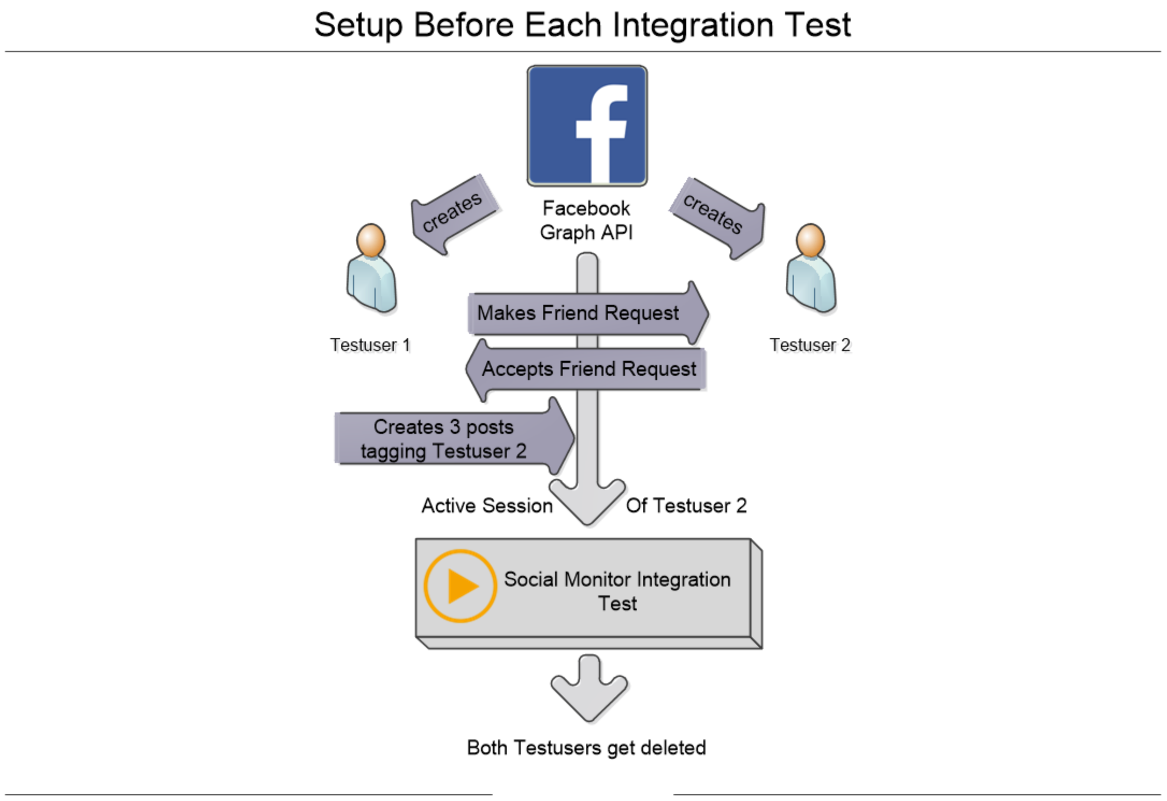

Setup Before Each Integration Test
Facebook Graph API
creates
creates
Testuser 1
Testuser 2
Makes Friend Request
Accepts Friend Request
Creates 3 posts tagging Testuser 2
Active Session
Of Testuser 2
Social Monitor Integration Test
Both Testusers get deleted

Figure 22: Life cycle of test users

2.4.2 Test CO1: Social Data Collection and Aggregation

The purpose of this test is to test the data collection and aggregation implementation functionality. Accessing, processing, and aggregating data from Facebook using the Graph API is a rather complex process due to the large number of different content item types. Especially nested items that refer to, e.g. items that were re-shared and not directly posted to the feed, require a recursive processing. Test C01, therefore, aims at testing the correctness of this process.

The test proceeds in the following manner:

1. The data bases are cleared.
2. A new session to Facebook is established by the Facebook API and OAuth2. After the session is established successfully, the API returns a corresponding success message and the Facebook user ID. The test compares if the session establishment is successful and the user ID is not null and not empty.
3. Next, information about the video posts on the user's Facebook feed is retrieved. The test succeeds if the number of retrieved video posts is greater than 0 and if for each metadata a value is given in the data base. The following values are checked for each video post in the database if they are not null and not empty. If one of the following variables has been assigned those values, the test will abort:
   a. ID
   b. VideoID
   c. Created Time
   d. URL of the Video
   e. Actor ID
   f. Number of Likes

g. Number of Comments

2.4.3 Test CO2: Social Aggregator

Initially, the “Social Aggregator” has been designed as a browser-based tool which shows the friendship connections of a Facebook user to other Facebook users. The tie strength of this friendship connection was shown by a line whose thickness was proportional to the number of posts seen from this friend on a user’s Facebook feed([D6.2], Section 2.4.3).

It is no longer possible to request the friends of a user since Facebook updated its API in a way that only friends which also use the same app can be seen. Therefore, and because of the extension of the app with respect to mobile video playback monitoring, we decided to modify the app. We integrated the new “Social Aggregator” in the “Social Prefetcher App” which has the advantage that everybody who has installed the app can use this tool without additional installation and configuration overhead. By doing so, we provide the users an additional incentive to use the app and follow a strong trend called “self-tracking”. This satisfies the need of users to analyze and learn about their own behaviour.

The two views provided by the “Social Aggregator” are integrated within the Facebook view of the app. Here the user can select between two views:

1. “Number of Videos Watched”, see Figure 23: The number of video posts shown on a user’s Facebook feed and the number of those videos watched are shown, per day for the last 7 days.

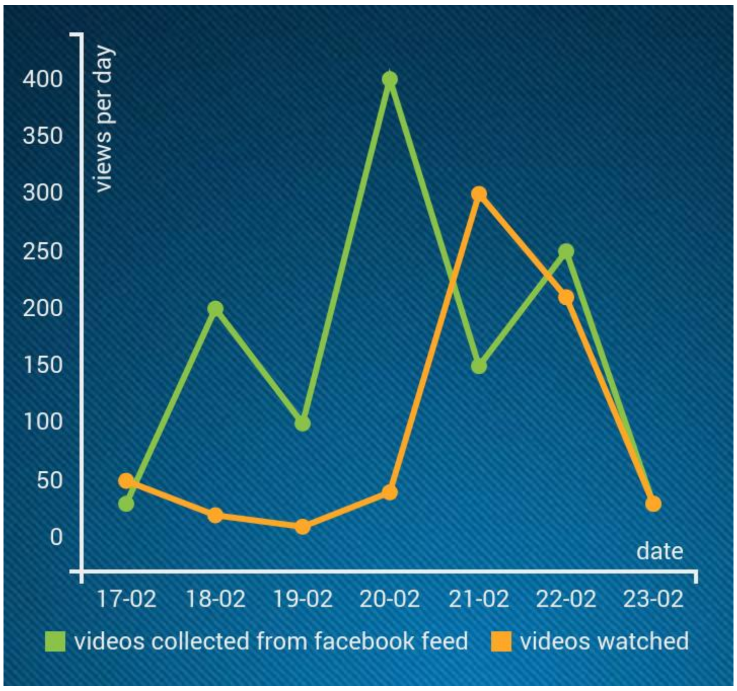
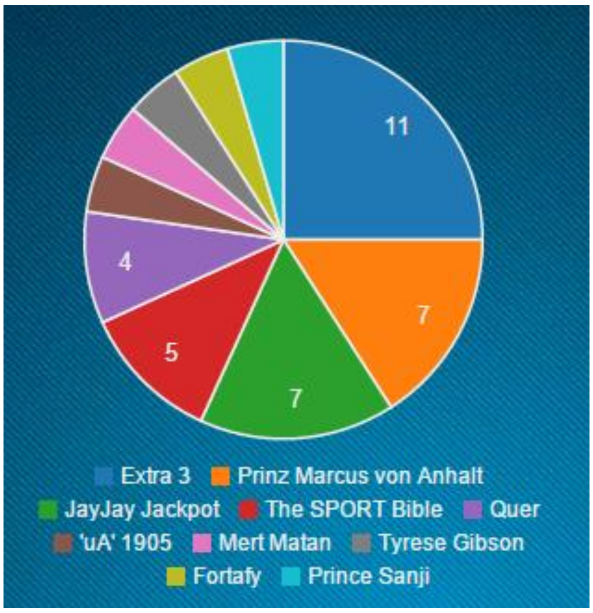


Figure 23: Number of video posts on a users’ Facebook feed (green line) and the number of video watched from the Facebook feed (orange line)

2. “Videos Watched From”, see Figure 24: The 10 most requested video sources from a user’s Facebook feed are shown. The pie chart visualization helps to get a clue on the distribution of the user’s video requests. For this visualization, all data collected since the app’s installation is used.

Figure 24: The ten most watched video posters on Facebook for a certain user. The pie chart shows the percentage distribution of videos watched.



The testing of the Social Aggregator is performed by the following steps:

1. For both views, the pie chart and the line graph, dummy data is inserted in the corresponding database automatically.
2. The JSON object is created by the software and handed over to the visualization library (C3.js which is based on D3.js) is compared with the expected result. The inserted data which is also expected by the JSON object created is shown in Table 4. The corresponding data for the pie chart is shown in Table 5. The JSON object which is created by the Social Aggregator based on this data is shown in Table 6 which is passed to C3.js. This JSON object is used as data source for the visualization graph in C3.js.
3. If the JSON object matches the expected JSON object result, the test succeeds. Otherwise it fails.

Table 4: Dummy data used for testing the line chart of the Social Aggregator

Date	Posts	Video sessions
Two days ago	2	0
Yesterday	3	0
Today	1	1

Table 5: Dummy data used for testing the pie chart of the Social Aggregator

Date	Actor
Two days ago	YouTube
Today	YouTube
Today	YouTube
Yesterday	Facebook

Today	Facebook

Table 6: Expected information in the JSON object created by the Social Aggregator for the visualization.

Actor	Number Of Posts
YouTube	3
Facebook	2

2.4.4 Test CO3: Social Tracer

The purpose of this test is to check the synchronization of the data collected on the client device, e.g. smartphone, and the server which is collecting the data. The upload functionality for each table in the SQLite database is going to be validated. Each test case performs the same steps. The source set is created dynamically.

Preparation:

1. Local/remote databases are recreated/truncated
2. Some content is inserted into the table. Each column is provided with a value if possible

Execution:

3. Before uploading the data, the data which is sent gets referenced in a variable (formatted as JSON).
4. Data is uploaded to the server.
5. A representation of the uploaded data is fetched via a GET-Request. The result is formatted in JSON

Assertion:

6. The JSON arrays are compared in detail.

These tests use a distinct test database. Therefore, it is not possible to request real user data by the method used by this test. However, the tables in this database are the same as for the productivity version. In the end, there is a scenario text where the lifecycle of the service is executed once. Test users are created like the introduction has shown.

2.4.5 TestCO4: Social Predictor

In this test an instance of SocialPredictor is instantiated and the result of the method „getPriority“ is compared to the expected priority. This validates the functionality of ranking videos which is used by the Social Data Collector Module. The priority is assigned based on the posts like count as described in the following. The videos with the lowest priority value (1) are prefetched first. Videos with a like count smaller than 21 get assigned the priority of 2. Videos with a like count greater than 499 get assigned 1 and videos with a like count between get assigned 3.

2.4.6 Test C05: Video Player

The purpose of C05 is to test the functionalities of the video player under different circumstances. Therefore a set of videos from different video platforms is used for this test. The corresponding videos are shown in Table 7. Videos from Facebook, YouTube, Vimeo and Dailymotion are tested.

Table 7: Videos used for the video player tests

URL	Facebook Video ID	Playable
http://www.dailymotion.com/video/x2kdub9_les-effets-de-base-au-tennis-de-table-partie-1-au-service_sport	-	Yes
-	10152810128498918	Yes
https://vimeo.com/120343236	-	Yes
youtu.be/Tqa8xcpBvA8	-	Yes

The performed test cases are executed as follows:

1. Playing valid Dailymotion video
2. Playing valid Facebook video
3. Playing valid Vimeo video
4. Playing valid YouTube video

Each test-case performs the same execution steps. It's necessary that the video under test has a minimum video length around 3 minutes.

Preparation:

The video link is passed to an instance of a LinkResolverTask which creates a streamable URL of the video. The video intent with the necessary information like stream URL, video title etc. is build. The video intent is passed to the VideoPlayerActivity. If Wi-Fi is disabled while a test-case is being setup, then Wi-Fi gets enabled.

Execution:

When the video starts playing, a callback is invoked to perform specified actions automatically. These actions are shown in Table 8.

Table 8: Actions performed during the test of each video playback

Action	Triggers
Skipping to position 1:30	Skip Event, (Possibly Buffering Events)
Disabling Wi-Fi	Connection Change Event
Pausing video	Pause Event
Enabling Wi-Fi	Connection Change Event
Changing orientation to Landscape	Orientation Change Event
Playing video	Play Event

Skipping to positions 2 seconds before video ends	Skip Event
Change orientation to portrait	Orientation Change Event

These tests result in database entries for each of the action, if successful. An example output of the database's table VIDEO_SESSIONS and VIDEO_TRACE_EVENTS is shown in Figure 25 and Figure 26.

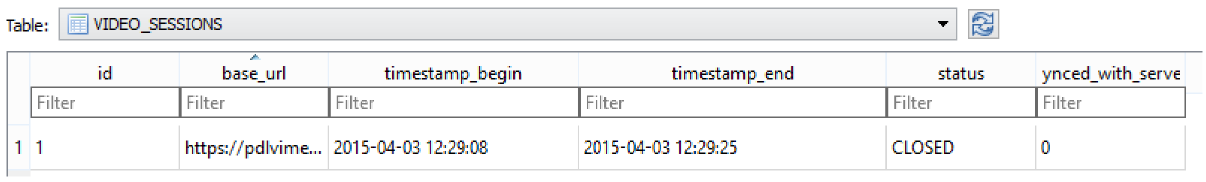
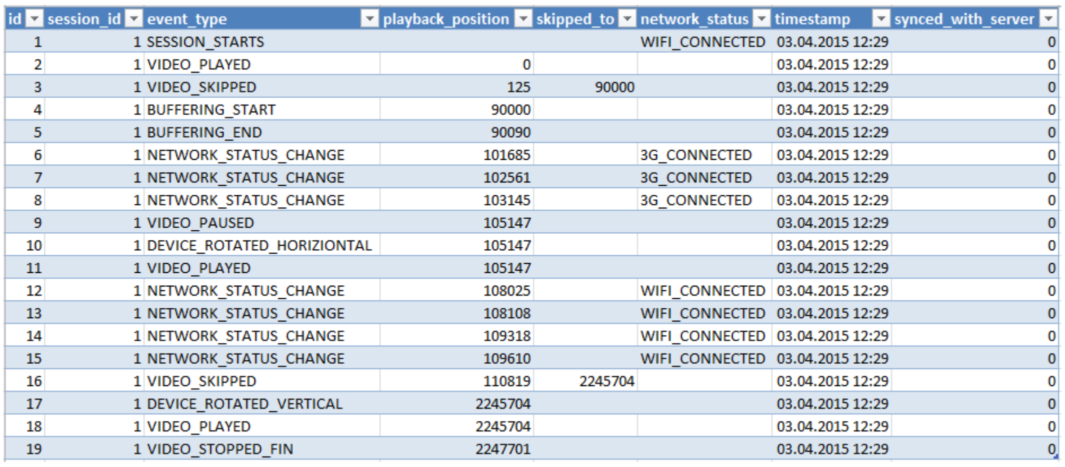


Table: VIDEO_SESSIONS

	id	base_url	timestamp_begin	timestamp_end	status	ynced_with_serve
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	https://pdlvime...	2015-04-03 12:29:08	2015-04-03 12:29:25	CLOSED	0

Figure 25: Example of a database entry in the table VIDEO_SESSIONS



id	session_id	event_type	playback_position	skipped_to	network_status	timestamp	synced_with_server
1	1	SESSION_STARTS			WIFI_CONNECTED	03.04.2015 12:29	0
2	1	VIDEO_PLAYED	0			03.04.2015 12:29	0
3	1	VIDEO_SKIPPED	125	90000		03.04.2015 12:29	0
4	1	BUFFERING_START	90000			03.04.2015 12:29	0
5	1	BUFFERING_END	90090			03.04.2015 12:29	0
6	1	NETWORK_STATUS_CHANGE	101685		3G_CONNECTED	03.04.2015 12:29	0
7	1	NETWORK_STATUS_CHANGE	102561		3G_CONNECTED	03.04.2015 12:29	0
8	1	NETWORK_STATUS_CHANGE	103145		3G_CONNECTED	03.04.2015 12:29	0
9	1	VIDEO_PAUSED	105147			03.04.2015 12:29	0
10	1	DEVICE_ROTATED_HORIZIONTAL	105147			03.04.2015 12:29	0
11	1	VIDEO_PLAYED	105147			03.04.2015 12:29	0
12	1	NETWORK_STATUS_CHANGE	108025		WIFI_CONNECTED	03.04.2015 12:29	0
13	1	NETWORK_STATUS_CHANGE	108108		WIFI_CONNECTED	03.04.2015 12:29	0
14	1	NETWORK_STATUS_CHANGE	109318		WIFI_CONNECTED	03.04.2015 12:29	0
15	1	NETWORK_STATUS_CHANGE	109610		WIFI_CONNECTED	03.04.2015 12:29	0
16	1	VIDEO_SKIPPED	110819	2245704		03.04.2015 12:29	0
17	1	DEVICE_ROTATED_VERTICAL	2245704			03.04.2015 12:29	0
18	1	VIDEO_PLAYED	2245704			03.04.2015 12:29	0
19	1	VIDEO_STOPPED_FIN	2247701			03.04.2015 12:29	0

Figure 26: Example of the database entries in the table VIDEO_TRACE_EVENTS

Assertions:

Finally, there must be an entry in the VIDEO_SESSIONS table with the base URL of the video. The value of the column „status" has to be „CLOSED". It's verified that all available trace events must exist as an entry in the VIDEO_TRACE_EVENTS table under the created video session (session_id). If any of the above mentioned conditions does not hold, the test fails.

2.4.7 Test CO6/CO7: Content Prefetcher, Download Scheduler, Download Client

The purpose of this test is to assess the video download capabilities of the mobile prefetching application. To this end, the videos stated in Table 9 are used. Please note there is one video (#3) which cannot be downloaded. The reason for this is because the video is blocked in Germany, where the tests are performed. The performed tests are comprehensive tests for the whole download cycle using the modules: content prefetcher, download scheduler, and download clients.

Table 9: Videos used for the Content Prefetcher assessment

Row ID	Name	URL	Facebook Video ID	Priority	Downloadable

1	South Park - Casa Bonita - Cartman vs. Kyle (german)	youtu.be/LS5mCg-mHFY	-	3	Yes
2	WebFilings runs on Google Cloud Platform	youtu.be/n3CqrGtsuGk	-	2	yes
3	Rihanna, Kanye West, Paul McCartney - FourFiveSeconds (57th GRAMMYs)	youtu.be/swRgOrKhP8M	-	1	no
4	Safer Internet Day 2015	https://fbcdn-video-g-a.akamaihd.net/hvideo-ak-xpf1/v/t42.1790.. (Optional)	10152989445275932	1	yes
5	SPORTS CONTENT	https://vimeo.com/120343237	-	1	yes
6	Gruselige Mutation: Schaf sieht aus wie ein Mensch	http://www.dailymotion.com/video/x2l0lyi_gruselige-mutation-schaf-sieht-aus-wie-ein-mensch_sport	Not needed	3	Yes

The following test procedure is performed for each of the test videos.

Test cases:

1. Videos are marked to be downloaded while the device is connected to the Internet by Wi-Fi
2. Videos are marked to be downloaded while Wi-Fi is deactivated and while the device is connected to the Internet by 3G/4G

Preparation:

- Local SQLite test database gets recreated
- Videos are inserted in the PREFETCHED_VIDEOS table (hoster: YouTube,Facebook, Vimeo,Dailymotion)

Execution:

- ContentPrefetcher gets instantiated to read videos from local test database and downloading the videos into the app's cache directory. Prefetching is started.

Assertions:

Test case 1: Videos downloaded over Wi-Fi

- Last modified time of the downloaded videos are compared to determine that the videos were downloaded in the correct order specified by the videos priorities specified by the Social Predictor module

- Entries in the table PREFETCHED_VIDEOS must have status „SUCCESSFUL“ which are defined as downloadable in the source set. If a video is not downloadable, it must have status „FAILED“.

Test case 2: 2. Videos downloaded with Wi-Fi deactivated

- All videos must have the status „NOT_PREFETCHED“ which means the Content Prefetcher did not run because no Wi-Fi connection was available.

2.4.8 Test CO8: Evaluation of Bandwidth Availability Prediction

During the overall span of eCOUSIN prediction techniques have been proposed with different degrees of reliability and confidence. While in the beginning we started assuming bandwidth prediction was feasible, we later moved to reviewing the state of the art on prediction techniques and mobile networks measurements [D5.2]. Then we developed a general forecasting solution spanning short, medium and long term prediction [D4.3,D5.3].

In particular, we verified that following categories of prediction are feasible and can be used to develop optimization techniques for mobile networks [BUI14]:

- Short term prediction: by means of filtering techniques (e.g. autoregressive and moving average, ARMA) we showed that the evolution of the achievable rate time series can be predicted and a bound on the prediction error can be computed. This allows us to compute useful quantities such as the prediction time reliability.
- Medium-long term prediction: when users dynamics make the use of filtering techniques unfeasible, we resorted on using statistical information and larger scale dynamics, such as cell transition probabilities to provide worst case approximations of the rate evolution.

More information about test results can be found in D4.3 Section 3.2.4, D5.2 Sections 3.1.2 and 5.3.4 and D5.3 Sections 3.1 and 4.3.3.1.1. The most relevant parts are also provided in the appendix of this document for the ease of reference.

Additionally, we studied a simple model that links prediction error to Gaussian random walks [BUI14b], which we summarize in what follows.

2.4.8.1 *Modelling throughput prediction error as Gaussian random walks*

This section focuses on the prediction of downlink rate between base station (eNodeB) and user equipment (EU). User throughput in mobile networks depends on several aspects and it is most always modelled as a function of the signal to interference plus noise ratio (SINR) γ and the number of active users in the cell *K*. The SINR is usually modelled as a function of the distance *d* between eNodeB and UE, the *K* active users and the scheduler type.

In what follows we specifically address LTE technology for the case of proportionally fair scheduling. The throughput *g* can be expressed as:

$$g = \frac{\eta\left(\gamma_0 d^{-\alpha} r(K)\right)}{K},$$

where $\eta(x)$ is a piece-wise constant function associating throughput to SINR ranges, γ_0 is a constant scaling factor related to environmental and system parameters (e.g., transmit power, antenna gains, etc.), $\alpha \in [2,4]$ is the exponent of the pathloss law and $r(K)$ is the fast fading gain and depends on *K* to model opportunistic gain achieved by the scheduler.

To study prediction error we synthetically created user data rate traces, $G(s, T_s, K)$, characterized by the sampling time T_s, the user speed *s* and the number of users *K*.

For what concerns prediction itself, we limited our focus on autoregressive and moving average (ARMA) filters, but extending the model to other filters is straightforward. The basic ARMA model is as follows:

$$X_i = c + \varepsilon_i + \sum_{j=1}^{p} \varphi_j X_{i-j} + \sum_{k=1}^{q} \vartheta_k \varepsilon_{i-k},$$

where *c* is a constant, ε_i are white noise error terms, φ_j, ϑ_k, *p* and *q* are the autoregressive and the moving average coefficients and their respective orders and X_i is the reference signal. This model is referred to as an ARMA(*p,q*) with reference to the order of the two parts of the filter. To determine the order to be used, we followed the Box-Jenkins method using automatic inspection of autocorrelation and sampled partial autocorrelation functions.

To generate error sequences and their statistics we operate as follows: first we obtain the optimal order of the ARMA filters to be used and, for each couple of speed and sampling period, we generate a training sequence from which we tune the filter coefficients; subsequently, we generate a set of tests sequences to compute the prediction error. Finally, we compute

$$\sigma_k^2(s, T_s, K) = \frac{\mathrm{E}\left[\left(e_{ijk} - \mu\right)^2\right]}{\sigma_G^2},$$

which represent the variance of the *k*-th prediction error normalized to the variance of the original training signal.

Using Gaussian random walks as a model for prediction error is interesting, because their total variance after *k* steps is proportional to the number of steps and they can be expressed as a sum of i.i.d Gaussian random variables.

We verified that assuming the error sequences to be drawn from zero mean normal distribution was a valid hypothesis. To do so, we perform the Kolmogorov-Smirnov test between the generated error sequences and theoretical normal distributions with zero mean and the same variance as the error sequences. All the tests performed rejected the null hypothesis according to which the error and the normal distributions are not equal.

Subsequently, by visual inspection of $\sigma_k^2(s, T_s, K)$ we noticed that:

1. it increases with the prediction distance
2. the steepness is increasing with increasing user speed and sampling time
3. the minimum error is decreasing with both user speed and sampling time
4. when $\sigma_k^2(s, T_s, K) > 1$ does not improve over randomly guessing the next sample of the sequence from the distribution and we define reliability time $T_c = \mathrm{argmin}_k \sigma_k^2 > 1$
5. the number of active users *K* has a negligible impact on the prediction error.

Thus we are looking for a family of linear equations that approximates the variance sequence:

$$\sigma_k^2(s, T_s) = \begin{cases} A(s, T_s)k + B(s, T_s) & k \le T_c/T_s \\ 1 & \text{otherwise} \end{cases},$$

where $A(s, T_s)$ represent the steepness and $B(s, T_s)$ the offset of the process or, in other words, how fast the prediction reliability decreases and how large is the intrinsic randomness of the process respectively. We fit two linear functions on the sT_s product to approximate $A = A_1 sT_s + A_2$ and $B = B_1 sT_s + B_2$ respectively.

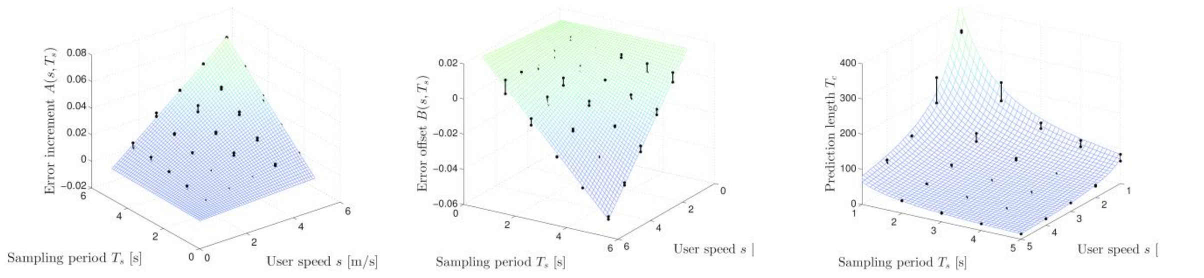
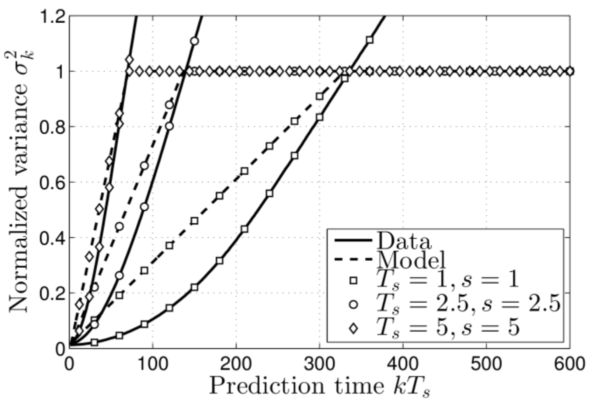


Figure 27 - Comparison between the collected data and the fitted model varying user speed and sampling time. Starting from the left, the figures show the approximation of *A* (left), *B* (center) and the reliability time T_c (right) as surfaces and the approximation errors as lines.

Figure 27 shows how close the model fits the data. The model coefficients have been obtained from the original sequences by imposing a perfect match for $s = 1$ and $T_s = 1$ and minimizing the least square error in the other points. In particular, the first and the second plots of the figure show the surfaces obtained from the linear approximation and the distance from the actual data and the surfaces. Also, the third plot shows the prediction validity length derived from the model (surface) compared to the same obtained from the data.



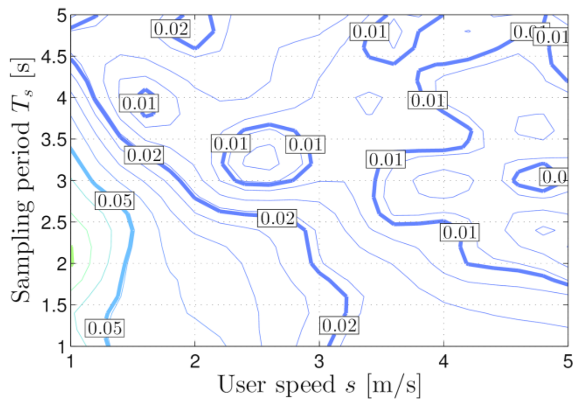


Figure 28 - Comparison between model and data for different s, T_s couples.

Figure 29 - Contour plots of the average distance between the model and the data.

Figure 28 visualizes how the model fits the data for a few speed-sampling time couples: solid and dashed lines represent the normalized variance σ_k^2 obtained from the data and from the model respectively; square, diamond and circle markers identify the (s, T_s) couple as (1,1), (2.5,2.5) and (5,5) respectively. In all the three cases the model fits the curves reasonably well and is always providing a conservative approximation: the predicted error is always larger than obtained from actual data.

Figure 29 shows contour plots of the average approximation error. Bold lines are marked with the actual error value, which is most always smaller than 2%, but for very small sampling time and user speed where it is slightly larger than 5%.

Finally, we conclude that Gaussian random walks can be used as a valid model for short term prediction errors since they can reproduce the main characteristics of the original random process. Also, random walks allow for an easier analysis of prediction based optimization problems: in fact, it is possible to approximate the distribution of the prediction error as a sum zero mean Gaussian variables: one of variance $B(s, T_s)$ accounting for the sequence inherent randomness and k with variance $A(s, T_s)$ each to account for decreasing reliability of the prediction after k steps.

2.4.8.2 *Analysis of the Bandwidth Prediction Study*

The QoS of mobile applications such as mobile cloud gaming or live video streaming rely on certain bandwidth and RTT guarantees [CLAY10, FIED10]. Also the profit of commercial web services depends on low network latency [KHIR02]. To be able to optimize the QoS according to the available network capacity and to avoid failing connections such as dropped VoIP calls, mobile applications need to rely on information about the performance of the underlying mobile network. This depends on a number of aspects, including the number of active users, their location, the time of the day, and likely also the day of the week, amongst other factors. One approach to improve mobile traffic scheduling is to use the information provided by mobile operators. However, the performance figures (e.g. data rates) advertised by mobile operators in many cases report primarily maximum values that often do not match the actual performance [HUAN13]. Moreover, detailed information about average number of users per cell, backbone capacity and also expected delays and available throughput are often not available at the required granularity for confidentiality reasons.

An alternative approach is collecting network performance data on end-user devices in a crowd-sourcing based approach [HUAN13, SONN13, YAO08]. This allows estimating and predicting from those the cellular network quality at certain times and locations, as experienced by the end-user. Still, active measurements are expensive in terms of energy and may even cause costs at the user's end. Knowing the network and cell utilization, a number of optimizations can be derived for the handset. For example background traffic may be delayed until the congestion is over or the user has moved to a less congested cell. This allows App developers to optimize their application to adjust the quality of the consumed content a-priori, also improving the user experience.

This work is focused on analysing the parameters affecting the mobile network performance. Special emphasis was put on the RTT, as an increasing number of services are delay sensitive (i.e. mobile cloud gaming, or health care). Hence, this work aims to address the following questions:

1. How does the network configuration and management influence the cellular network performance?
2. Can the performance of the cellular network be predicted based on the available measurements?
3. What is the optimization potential in the given network configuration, and how can this be exploited?

The analysis is based on an results of a crowd sourced measurement campaign, and an extensive stationary network measurement study. The evaluated subset is detailed in the respective sections.

The findings in this work differ from related studies in that the analysis performed also includes the paths taken through the network to derive their influence on the end-to-end performance. From this it is derived that network configuration and management considerably influence the received performance. The work contributes a detailed analysis of the mobile end-to-end measurements considering the RTT in LTE, the analysis of the cell density for one cellular network provider, and a model to predict the experienced RTT depending on past observations, and suggests improvements in the network configuration and management to considerably increase the end-to-end performance.

2.4.8.2.1 Background and Related Work

A number of publications deal with the measurement of cellular network performance [HUAN13, SONN13, YAO08], its prediction [YAO08, BUI14b], energy consumption [BALA09, VERG13], or mechanisms improving the mobile QoS, while keeping the QoE high [ICKI13]. A large number of studies focus on 3G networks [YAO08, VERG13, BALA10], while others [HUAN13, REZA11, GUO13, HUAN12] evaluate the low-level network-based performance or model the energy efficiency.

At the same time, a number of publications cover the cellular performance based on samples collected at the mobile backbone. Gerber et al. [GERB10] describe an approach to measure the maximum mobile throughput based on packet traces. This approach is extended by Huang et al. [HUAN12], evaluating the throughput of a 4G network and individual flows in a similar manner. Both studies observe the traffic generated by a large number of devices over the area of several base stations. This allows deriving data rates for single devices as well as the overall throughput from within the network. The approach presented in this work differs, as it allows drawing conclusions on the coverage of the individual cells, and their number at given locations. Furthermore, the RTT is analysed on a per-cell basis, which was not conducted in these studies.

Wylie-Green et al. [WYLI10] analyse the throughput and RTT performance of an unloaded and loaded LTE network from different distances using multiple devices. The described measurements were conducted on newly built hardware and hence analyse the optimum network performance. The purposely generated traffic does not affect the system performance, as it is below the system capacity.

A detailed analysis of the one-way delay of different parts of the network can be found in [LANE12]. The authors measured the timing between a local machine connected to a LTE and HSPA network, a time synchronized server, and additional vantage points within the mobile network. This work differs, as it uses end-to-end measurements from the mobile device and evaluates the performance depending on the cell.

A different approach, based on measuring the network performance from handsets only, is published by Sonntag et al. [SONN13]. They published an application measuring a number of network parameters like throughput and RTT, and collect the data on their server. The analysis of the collected data is limited to a few general metrics and the creation of bandwidth maps. Contrary, this work analyses the measured RTT in detail to derive the performance based on the path taken through the mobile network.

The accuracy of RTT measurements on Android handsets was evaluated by Li et al. [LI15]. From this the overhead for the case of a native ping on the Nexus 5 was derived. Here, the maximum RTT increase is 7.7ms with a variance of 2.3ms.

The prediction of the availability of Wi-Fi networks is used to improve mobile connectivity by Nicholson et al. [NICH08]. This approach is extended by Bui et al. [BUI14b], who propose a stochastic mobile bandwidth prediction model. A different approach is used by Wac [WAC09], applying machine learning techniques to predict mobile QoS. In this work, this approach is extended to forecast the RTT based on the time of the connection.

Contrary to [HUAN13, SONN13, YAO08] this work focuses on the root cause analysis of performance variations in the cellular network. Two different data sets were collected, complementing each other. The crowd sourcing based data set is similar to [SONN13], but also includes fine granular location and cell information. The complementary data set extends the first one by RTT measurements to the DNS server and trace-routes between the mobile device and measurement server. Based on the combination of both, a detailed analysis of the cellular network performance becomes possible. No such data set or analysis is available to the best of the author's knowledge.

2.4.8.2.2 Measurement Methodology

The mobile network measurements have been performed with an Android application, the *NetworkCoverage App*[1]. This application was developed to gather measurements from various

[1] `https://play.google.com/store/apps/details?id=de.tudarmstadt.networkcoverage` [Last access: 2015-03-11]

devices, users, and locations. It consists of a service that passively monitors system resources like GPS and the cell interface and a component executing active measurements probing the network performance by generating traffic on the cell or Wi-Fi interface.

The application passively monitors the local signal strength, the network provider, and the used technology. Those measurements are connected to a time stamp and location derived from the Android framework. The location is either GPS-based, or based on information from available networks using Google's location API. Active measurements include RTT and throughput measurements. The RTT measurements are executed periodically in the background, resulting in a large number of samples. The throughput measurements are generally only collected upon user request, as the data transfer might cause additional cost at the user's end. The measurement values are stored in a local *SQLite* database and transmitted to a data collection server for later analysis.

Both the RTT and throughput measurements are run against a server at TU Darmstadt, allowing the maximum flexibility in the measurement setup. The server and the university network are well connected to the public Internet using a lightly loaded network. The mobile devices used in the measurement studies are all Nexus 5 devices running Android 4.4.4, while the devices used by the users of the public application vary. Hence, thorough filtering was applied to assure high quality of the measurements.

When measuring throughput and RTT, the cellular network poses additional challenges compared to wired networks, requiring adaptation of the measurement procedure. Due to the mobility of the user, the duration of the measurements must be kept to a minimum. This is particularly important for the throughput measurements, as the available data rates may change within seconds.

The RTT measurements use the native ping command on the Android device to determine the latency to a server located at TU Darmstadt. The minimum, average, and maximum RTT out of five consecutive ping measurements were recorded. These measurements were run once every minute to gain a fine-granular view on the network performance.

Network operations on mobile devices cause an additional delay, which is caused by the kernel and operating system processing the outgoing and incoming packets [LI15]. This delay is different depending on the chosen implementation. For our measurements, the RTT is measured using the native ping command, eliminating the influence of the Android virtual machine and its scheduling on the accuracy of the measurements. The ping command directly accesses the low level timing functions. According to [18] the additional delay of the sent and received packets on the Nexus 5 for a measured RTT of 20ms is 6.028ms with a 95% confidence interval of ±0.811ms. The TU Darmstadt server used for the RTT measurements is also used to measure the throughput between the end-user device and the server. The measurements are run using a patched version of *iperf*, allowing to measure the down-link bandwidth (conventional *iperf* always measures the up-link from the *iperf* client). The generated traffic is reduced by limiting the time of the measurement, which by default is set to run for 10 s. In lab trials the minimum measurement duration on Wi-Fi was determined to be 3 s for the connection to exit the slow start phase and remain within a limit of ±10% compared to the full measurement. Repeating the same measurements on a 3G network lead to a minimum measurement interval of 5s due to the higher RTT. The accuracy of this setting was also verified on LTE and hence set as the default interval for all measurements.

2.4.8.2.3 Description of the data sets

The data sets used for the analysis of the network performance are split into two sets as the focus of the analysis and the collected data are slightly different. Furthermore, this simplifies referring to the respective subset of measurements. The first data set consists of the crowd-sourcing based data used for the estimation of the location-dependent network performance, while the second measurement study is focused on the time-based analysis of network performance.

2.4.8.2.3.1 Description of the crowd-sourcing based cellular network measurements

The first data set, targeted at analysing the location-based network performance, was collected by users of the NetworkCoverage Application. The measurements go back to December 2013, where the first version of the application was released. The full data set consists of 7.1M coverage samples, 395k RTT samples, and 6.8k throughput measurements. The measurements cover mainly Darmstadt and the surrounding areas, although a few measurements are collected in other cities. Table 10 lists the break down for the different technologies and measurement types.

The full data set covers 6596 cells operated by 64 network service providers. An interesting effect is that single cell IDs of one operator have been observed serving multiple generation networks. According to the mobility of the users, some cells observe a high sample count (>10k), while others are only observed a small number of times. The median number of measurements per cell is 12, while the mean is 636 observations.

During this time, also a measurement study collecting mobile traces for the analysis of the RTT and its relation to the signal strength and the network throughput was run, which began on June 2nd 2014 and ran for two weeks. For this, five students were recruited to carry one smartphone periodically probing the network performance during their daily routine. The phone model used by all participants in the study was the Nexus 5. Most of the time, these were connected to the LTE network, providing a basis for the detailed analysis of its performance. Over the course of the measurement study, the RTT measurements were executed once every minute, and the throughput measurements were run every 25 minutes.

Table 10: Overview of all collected measurements

Type	4G	3G	2G	all
Signal Strength	6.5M	538k	64k	7.1M
Cells	1748	14k	2k	17,9k
RTT	344k	45k	5k	395k
Throughput	5.5k	1.3k	36	6.8k

2.4.8.2.3.2 Description of the data set collected by the stationary measurement study

To assess the time-based performance of the cellular network, additional measurements are required. Hence, a second study targeted on the time-variability of the network was executed. Contrary to the first study the location of the devices was kept constant to keep as many parameters stable as possible. The target was to reduce the effect of handovers, different network technologies, and changing signal strength. The measurement study was run between December 19, 2014 and January 19, 2014, and hence includes measurements of two regular weeks as well as measurements during the Christmas and New Year’s celebrations. On overview of the measurements recorded during the study is given in Table 11.

The second data set consists of measurements collected by devices located at three different locations in the city, allowing assessing the time-based network performance without the influence of changing signal strengths, cell IDs, network technologies and handovers. Two devices were positioned in different residential areas approximately 1km from the city centre, while a third device was located in an office building adjacent to the central park in Darmstadt. These locations were chosen to represent different loads in different cells of the mobile network. These measurements were run for 4 weeks to assess the time-based network performance. For reference, two devices were taken on the usual trips between the residential areas and the office building for the course of one week.

Additionally to the previously measured metrics, also the trace-route between the mobile device and the server was recorded on the mobile devices. This will be referenced as the forward trace-routes in the remainder of this publication. As the forward trace-routes proved problematic (due to errors or traffic engineering in the cellular network), the measurement server was extended to also run a trace-route to connecting mobile hosts. In the following, this is referenced as reverse trace-route. The goal of this setup is to defer the routes taken through the core network, and such derive possible causes of the observed RTT deviations. The recorded information on the server consists of an identifier of the mobile device, the timestamp, and the complete output of the trace-route. Such, the RTT to the different hops can be measured.

Here, care must be taken when interpreting these results, as ICMP unreachable messages, as provoked by the trace-route measurements, are generated with lowest priority by intermediate routers, resulting in a high variance in the measurements. Further, it is possible that intermediate hops show an increased RTT, while the next hop shows a better performance. Each hop was probed with three packets. Due to the low priority of these packets, only the minimum of one probe was considered in the later evaluation.

These reference measurements were exclusively conducted on the 4G network. The end-to-end RTT as already sampled in the last studies was recorded once every minute. The forward and reverse trace-routes as well as the RTT to the DNS server were measured once every 15 minutes to limit the load on the network. The large changes in RTT as observed during the stationary measurement campaign were in the range of days; hence a resolution of 15 minutes for the evaluation of changes in the network is sufficient. The overview of the additional measurements conducted during the reference study is given in Table 12.

Table 11: Overview of the measurements collected during the stationary measurement study

Type	4G	3G	2G	all
Signal Strength	3M	11.2k	0	3M
RTT	155k	1.3k	0	156k
Throughput	2.3k	15	0	2.3k

Table 12: Overview of the measurements collected during the reference measurement study

Type	Number
Signal Strength	755k
RTT	38k
trace-routes	5k
reverse trace-routes	7.4k
DNS RTT	6.4k

2.4.8.2.4 General observations of the Cellular Network Performance (3G/4G)

From the full data set as described in Section Description of the data sets a number of conclusions can be drawn. First, the observations regarding single parameters are described. Then, these measurements are combined to analyse correlations between the individual performance measurements.

2.4.8.2.4.1 Cell coverage

The collected data allows estimating the signal strength at a given location. But as also the cell ID, lac, and mobile operator are stored with each sample, also the size of cells can be derived. The measurements resembling the extent of a cell are determined by grouping all samples with the same operator, lac, and cell ID. The accuracy of the estimate is achieved by limiting the location accuracy of the selected samples to better than 15 m, and removing points where the signal strength is invalid. To discard the influence of sparsely sampled cells, only cell-IDs with more than 100 measurements are considered. From the remaining samples, the extent of the cell is estimated by calculating a convex hull covering all points. The mean size of the remaining 480 cells is 5.3 km^2, while the median is only 0.56 km^2, indicating a small number of large, and a large number of small cells. The CDF of the number of observed cells at each location is given in Figure 30.

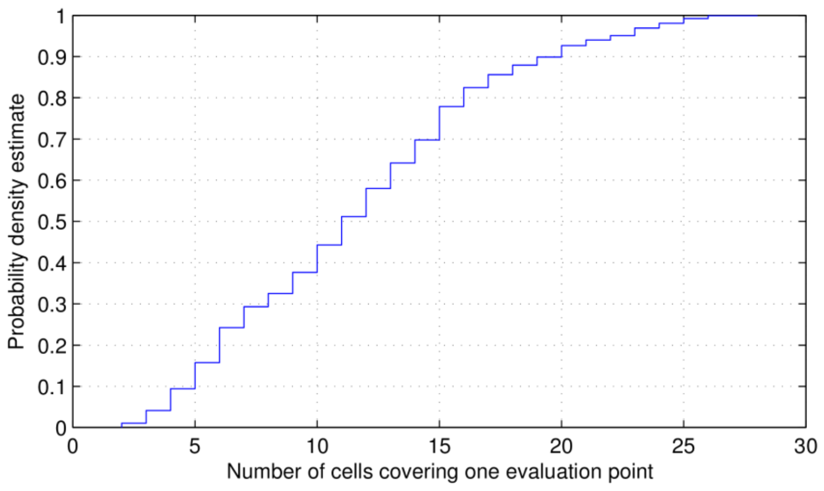


Figure 30: CDF of the number of cells covering one evaluation point

The cell density for the city of Darmstadt is evaluated by creating a grid of sample points covering the inner part of the city. The cell density was sampled at the centres of each circle in Figure 31. For each sample point, the number of cells observed at this location is calculated and a heat map generated. The map shows that a larger number of unique cell IDs is observed in the city centre, while the number is smaller in the outer areas. As each cell provides bandwidth at a different frequency range, this is expected to be closely related to the mobile data traffic generated in the respective area. Furthermore, smaller cells provide a better signal-to-noise ratio, and such, higher bandwidths to the individual user. Additionally, scheduling decisions can be adapted by calculating the probability of leaving a cell within a given time.

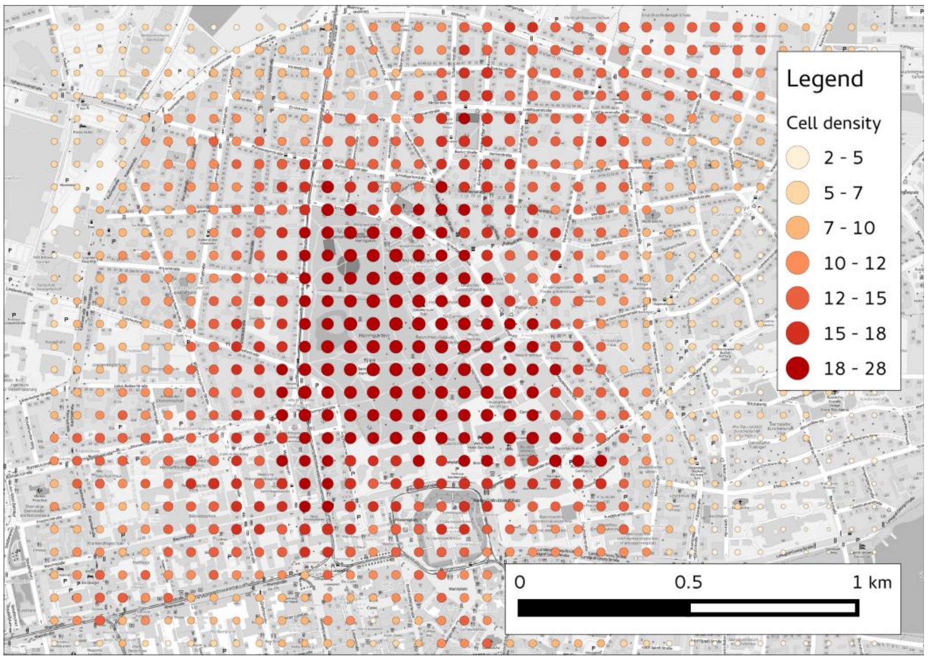


Figure 31: Heatmap of the number of cells of a single network provider covering each of the evaluated locations

2.4.8.2.4.2 Relation between RTT and Throughput

In order to determine inexpensive measures from which the mobile network quality can be derived, correlations between different network performance parameters, in particular available throughput and RTT, have been analysed. Figure 32 shows the relation between the RTT and the down-link rate. The 3G measurements show a consistent maximum throughput rate, while the RTT is uniformly distributed in the range of 30ms to 80ms. Also for an increasing RTT a decreasing data rate is visible. An interesting observation is that the 4G measurements show RTT clusters at around 20ms and 30ms. This will be analysed in detail in the Section 2.4.8.2.5.

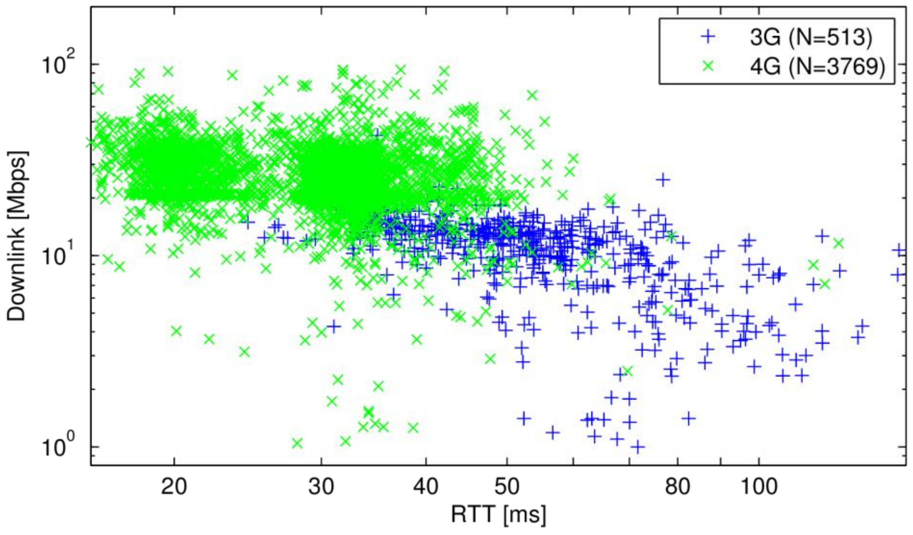


Figure 32: Relation between down-link bandwidth and RTT

2.4.8.2.4.3 Relation between ASU and Throughput

The signal strength is given as ASU, which is a linear mapping from the signal strength in dB to integer values. The influence of the signal strength on the maximum throughput of wireless networks is well known [PROA09]. This effect is visible in the throughput measurements conducted during this study. Figure 33 shows the measured down-link rates for each signal strength value. The second row of values indicates the number of samples contributing to each group. The median of the measurements is plotted in the centre of the box, while the box itself represents the inner quartiles above and below the mean. The whiskers extend to the maximum value within 150% of the height of the inner quartiles. All other values are marked by a plus, denoting outliers. As expected, the throughput increases with increasing signal strength, but only up to medium values (~18). Above, other factors appear to be limiting the maximum throughput.

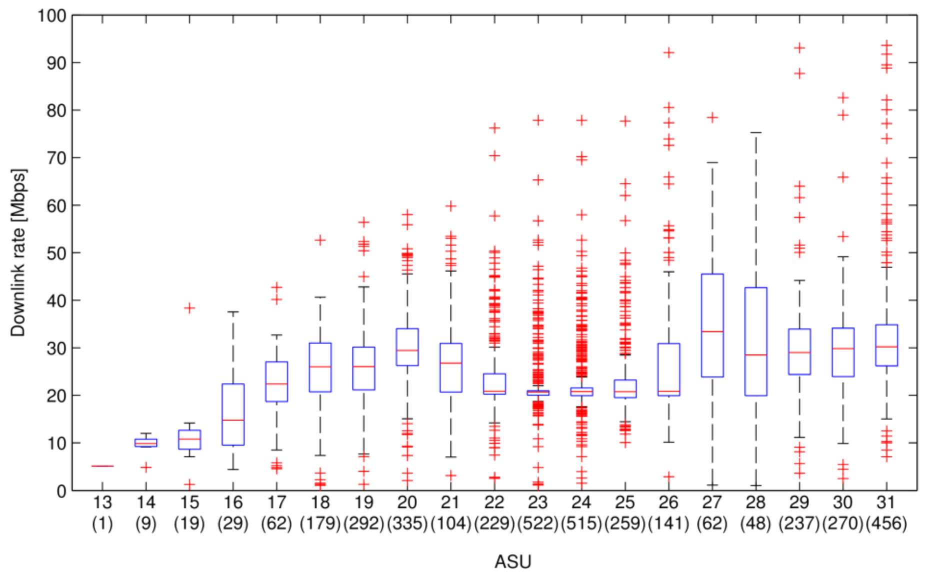


Figure 33: Distribution of the down-link rates in a 4G network. The numbers in braces denote the number of samples

2.4.8.2.5 Detailed analysis of the 4G network performance

This section covers the detailed analysis of the variances in RTT as discovered in Section2.4.8.2.4. A special focus is put on the influence of network management and configuration on the achieved network performance.

2.4.8.2.5.1 RTTs

Figure 34 shows the histogram of the measured RTT. The graph shows two distinct peaks, one at 20ms and two merged ones at around 28ms and 31ms. It is astonishing that no samples have been observed in the in the gap between. As the scheduling of packets in LTE is conducted once per ms, samples between the peaks should be present, if that was the cause. As no simple explanation for this behaviour could be found in literature, additional measurements were conducted to analyse the effect in detail.

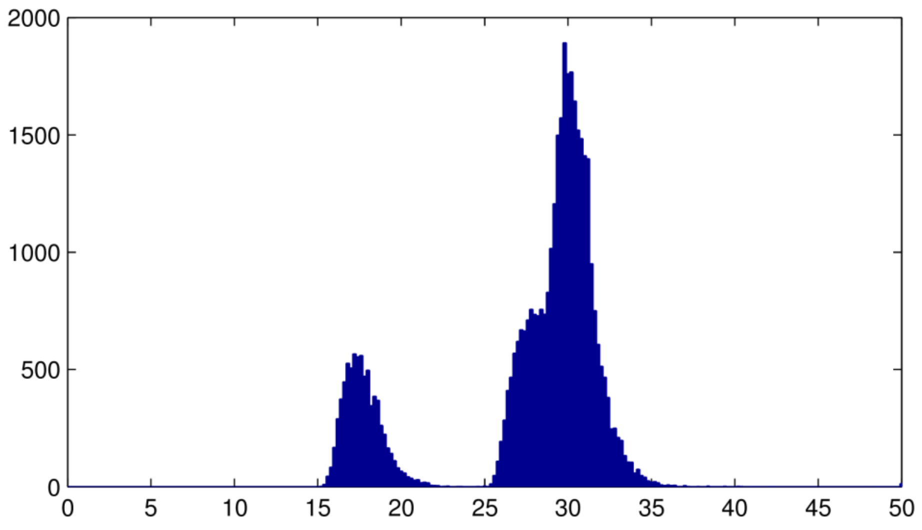


Figure 34: Histogram of the minimum 4G end-to-end RTT (bin width: 1ms)

2.4.8.2.5.2 Analysis of the network topology

The performance of the cellular network is analysed in detail by running trace-route measurements on the mobile devices, and measuring the same path from the server in the direction of the mobile device. In the trace-route measurements, the mobile network is opaque, i.e. no ICMP unreachable messages are generated by routers within the mobile operator's domain. From the host names it can be derived that the last visible hop corresponds to a router in the Telekom Network, which is likely the gateway to the mobile part of the network. This is in the following taken as the discrimination between the core network and the mobile domain (T-Mobile network).

To compare these measurements, the trace-route measurements on both sides and the end-to-end RTT are correlated using the device identifier stored with every measurement and the time. Analysing the time series of each device shows jumps in the RTT in the range of days. Hence, a sampling interval of 15 minutes for the trace-routes is sufficient. The data is then grouped into one hour wide bins. For each measurement (E2E, device to server, server to NAT), the minimum RTT of each series is recorded.

Figure 35 shows the measured RTT for an exemplary device in different parts of the network. The red line indicates the minimum RTT as already discussed in the previous sections. The blue line gives the minimum RTT to the DNS server configured by the operator. The green line corresponds to the minimum RTT as measured between the server at TU Darmstadt and the last hop visible in the trace-routes. This is indicated as *MobEdge to Srv*. Subtracting the minimum RTT between server and mobile edge from the end-to-end RTT allows estimating the influence of the mobile operator's network (wireless network and cellular backbone) on the overall RTT as measured between the mobile device and the server. These fit quite well with the RTT to the DNS.

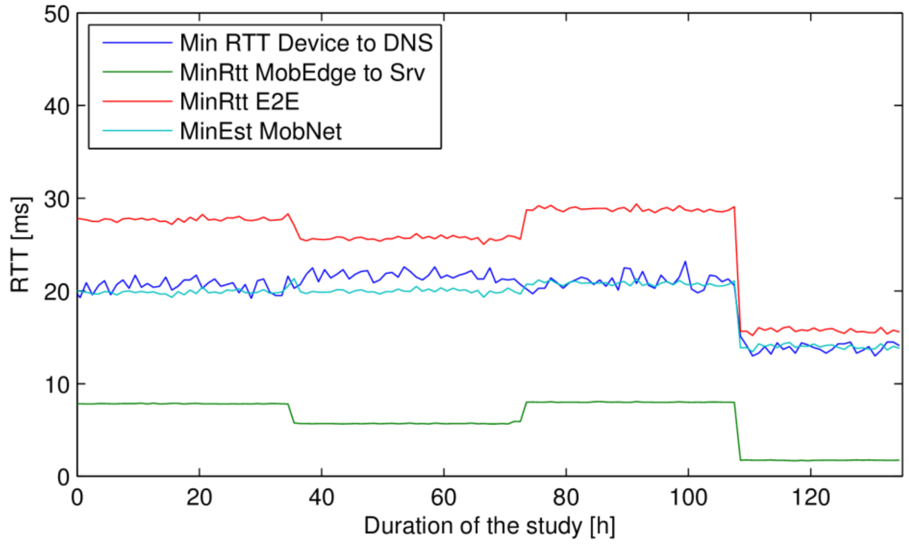


Figure 35: Exemplary RTT measurements for one device with the estimated RTT within the mobile operator's domain in turquoise

Analysing the address of the DNS over the course of the measurements shows that the same address is configured on all devices at all times. Still, comparing the RTT to the DNS in Figure 36 shows differences in the performance over time. From this it follows that either DNS requests are always rerouted to the closest DNS server, or different subnets are used at different PoPs.

Another important observation in Figure 35 is the changing performance over time. This is consistent with the observations in Figure 34, but contrary to the aggregated measurements over the full duration of the study, three different performance levels are visible. Furthermore, the performance is constant over the course of days, after which it changes to stay constant for another time frame of similar length.

From the host names of the traversed routers groups of the PoP can be derived. From here on, these will be referred to as PoP 1 to 3. The unique part of the host names are used to indicate the performance of each geographical location in Figure 36. These routers also show similar performance, leading to the conclusion that the performance of the cellular network is mainly related to the routes taken through the network. By clustering these by location the cellular operator's PoP can be derived.

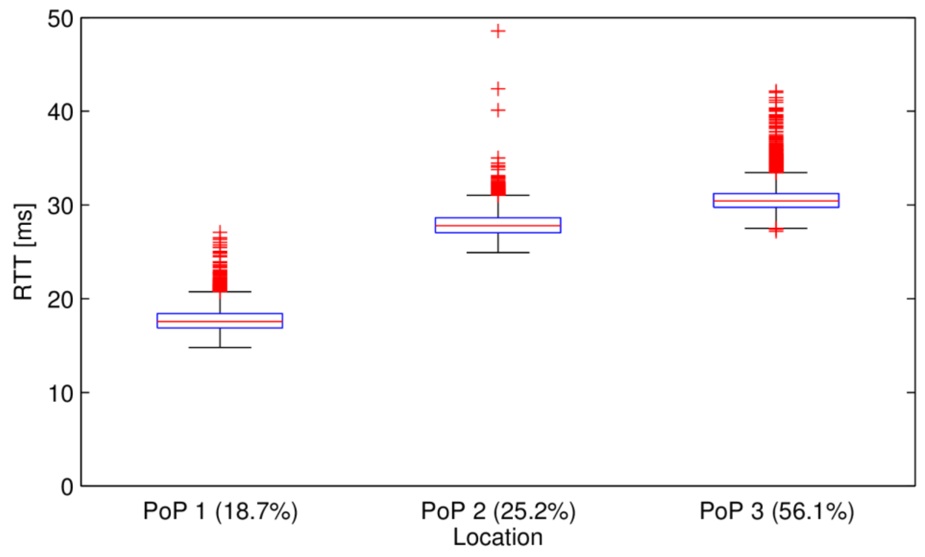


Figure 36: Boxplot of the minimum RTT and the fraction of measurements between mobile device and measurement server, grouped by the derived PoP of the operator

Table 13: Mean minimum end-to-end RTT and 95th percentiles of the minimum RTT measurements for all connections mapped to the different PoP

	Median	**Mean**	**95% confidence**		**95th percentiles**	
	RTT	**RTT**	**min**	**Max**	**min**	**max**
PoP 1	17.5720	17.7395	17.7124	17.7667	16.1510	19.8740
PoP 2	27.7930	32.7421	29.4212	36.0631	26.2100	30.1820
PoP 3	30.4350	30.5994	30.5802	30.6186	28.8890	32.8996

These groups can then be used to map the end-to-end measurements collected on the mobile device to specific PoP, from which their performance can be derived. This is done by identifying the intervals with constant PoP from the reverse trace-routes and then collecting the end-to-end measurements for the identified interval. The resulting distribution of the minimum RTT observed at these locations is plotted in Figure 36 and the measurements are also detailed in Table 13. As is already visible in Figure 34, the measurements for PoP 2 and Pop 3 overlap. This is caused by the noise on the measurements introduced by the mobile network and the measurement accuracy of the mobile devices.

Contrary to other measurement studies, and common assumptions of the workings of cellular networks, these measurements show that the performance of the mobile network to a considerable amount depends on the routing and management decisions of the mobile operator. The penalty in the minimum end-to-end RTT is identified in Table 14. Taking the best observed performance as reference, the penalty when being assigned to other PoP in the mobile network is between 58% and 73%. The RTT increase between the server and the mobile edge is between 245 and 365% while the increase in the mobile operator's domain is only between 41% and 47%. Considering that only 19% of the traffic is routed via the optimal path, a potential for improvement is apparent. Deriving the optimization potential from this, the E2E RTT may be reduced by 37% to 42%, while the reduction in the Internet path is between 70% and 78%.

Compared to these numbers, the optimization potential in the operator's domain with 29% to 32% may seem quite low and hence, may have been accepted by the mobile operator to increase the utilization of their existing hardware at other PoP. Still, this decision has an impact on other domains of the network, and consequently on the end-to-end RTT as experienced by the end-users.

Considering the effort by commercial web-site operators in reducing the latency by a few milliseconds to increase the profit, huge improvements are possible in the mobile network management by optimizing the PoP assignment for mobile devices depending on the destination.

Table 14: Possible reduction in RTT in the different domains when selecting the optimal PoP for the selected server. Here, the column I-Net denotes the RTT between server and mobile edge, MobOp the estimated RTT within the mobile operator's network, and E2E the

	Median RTT overhead			Percentage
	I-net	**MobOp**	**E2E**	**of packets**
PoP 1	0	0	0	18.7%
PoP 2	247%	41%	58%	25.2%
PoP 3	365%	47%	73%	56.1%

2.4.8.2.5.3 Comparison of the observed variances on-device overhead

Comparing the measurements with the results by Li et al. [LI15], the variances are similar. For the Nexus 5 ±2.331ms for an RTT of 20ms and ±0.811ms for an RTT of 50ms were determined. The measurements of the minimum RTT as reported here (cf. Table 13) show a width of less than 1ms in the case of the PoP 1 and PoP 3. Only for PoP 2 the confidence interval is larger. Still, as the RTT measurements show no normal distribution (cf. Figure 36), the 95*th* percentiles give a more accurate representation of the underlying data.

The measurements reported here are observed by the native ping in the user space. Hence, if the RTT excluding the processing on the local device is required (i.e. between network interface and server), the measured RTT as presented here may be reduced by 6.028 to 7.700ms.

2.4.8.2.5.4 Analysis of the periodicity of the cellular network performance

The performance of the cellular network, as is already visible in Figure 35, is time dependent. To derive the pattern, the begin and end of each PoP assignment as derived from the reverse trace-routes is analysed.

Calculating the duration of each PoP assignment interval and aggregating the resulting values allows deriving the periodicity of changes in network performance. These measurements are given in Figure 37 in form of the red bars. This histogram shows the number of occurrences of the specific connection duration. For visualization purposes 6 hour wide bins were chosen, although the data is much more accurate. Our findings show that a major peak is visible at 36 hours, and additional peaks at multiples of 36 hours (72, 108, 144). From this it is concluded that a re-assignment to the PoP is executed every 36 hours.

Cross correlating the PoP re-assignment intervals with the duration of use of a single public IP by the individual devices shows a similar behaviour. The calculated intervals are given in Figure 37 as blue bars. Also here, the observed intervals are a multiple of 36 hours. As the number of observed public IPs is larger than the number of available PoP, the probability to detect a change is higher. Concluding, the interval of 36 hours is a configuration parameter. Its purpose is not fully clear, but it can be expected that it allows cleaning buffers and re-adjust the load on the respective PoP.

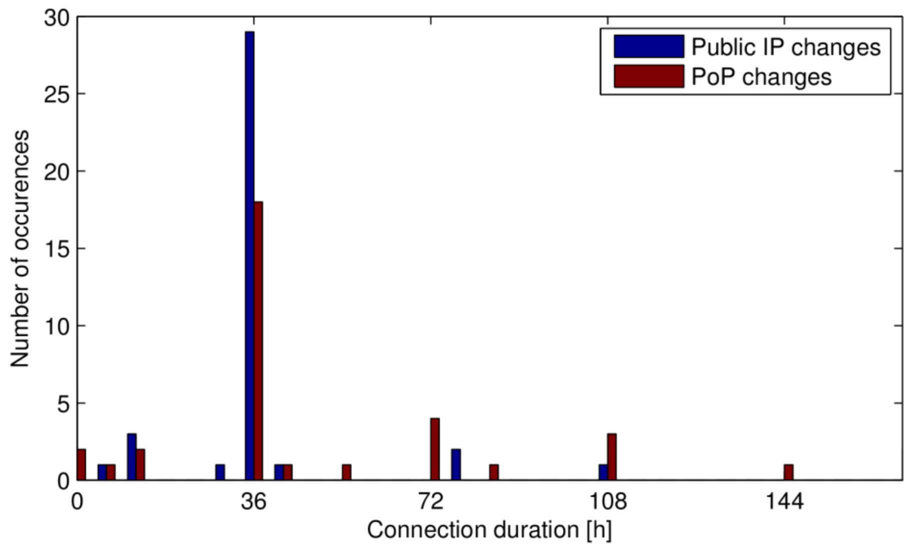


Figure 37: Histogram of the duration of IP and PoP assignments

A re-assignment to a different PoP can also be triggered by disabling mobile data and re-enabling it as soon as the connection was fully shut down. This leads to a random association to one of the available PoP. From the frequency of assignments to the respective PoP, the mobile device is assigned to the optimal location for the measured connection with a chance of 19%. Considering the placement of data centres within Germany, where a large number is built close to the German Internet exchange DE-CIX, the structure of the cellular network, as observed in these measurements is even worse.

2.4.8.2.5.5 Comparison to other networks

To assess the validity of the observations, also reference measurements were conducted using a Motorola Moto G 4G (1st gen) on a Spanish cellular network". These measurements were run for ten days and show effects similar to the German network in that the public IP visible to the server is stable for a number of hours. The observed interval for the IP re-assignment is 12 hours, indicating that this is an operator configurable parameter for network maintenance. This observation is consistent over the first seven days, and a later verification period of three days. Considering that mobile devices sometimes keep a large number of connections open enabling bi-directional communication and notification of changes on the server, a large number of ports on the NAT gateway needs to be kept open and mapped to the mobile device. Re-assigning the devices allows the operator to clean the state kept on the NAT gateways.

Due to the limited number of measurements, only one PoP was visible in the Spanish data set. Contrary to the measurements in the German network, the last hop in the mobile network is visible in the forward trace-routes. This allows directly measuring the influence of the cellular network from the mobile device. Two devices where visible, acting as NAT gateways, giving a coherent RTT of 30ms. From this it can be concluded that only one PoP was measured.

2.4.8.2.6 Predicting the cellular network performance

Based on the observations derived from the measurements, first a Markov Chain modelling the assignment to the PoP is derived. Secondly, a machine learning approach is used to predict the PoP of a mobile user when connecting to the mobile network, and consequently the RTT performance for specific services.

2.4.8.2.6.1 Analysis of the transition probabilities using a Markov Chain

To understand the possible transitions between the PoPs, first a Markov Chain is calculated, modelling the observed data. The data set used for the analysis is the second data set as described in Section 2.4.8.2.3. Within the data set 44 different states can be observed, from which 38 transitions are derived. These transitions contain changes between the different PoP, or no change after the re-assignment interval of 36 hours. No transition between PoP 1 and PoP 2 has been observed, but frequent changes between other PoP.

Whenever the device connects to the cellular network it is associated with one of the three PoPs (with probability of 56% for PoP 3, 25% for PoP 2 and 18% for PoP 1). The chain of dependencies between states is presented in Figure 38. The transition probabilities between the PoP states represent the change of the PoP after the re-assignment interval. The probabilities are not equal, and some nodes and transitions are more likely to be executed, than others.

The Markov Chain can be used for any new device attempting to attach to a network to predict its most likely attachment state and subsequent transitions. In a given PoP state, the distribution function for the next transition is triangular *T*(0,0,36), representing the deterministic change of PoP for a device after 36h of being attached to a given PoP.

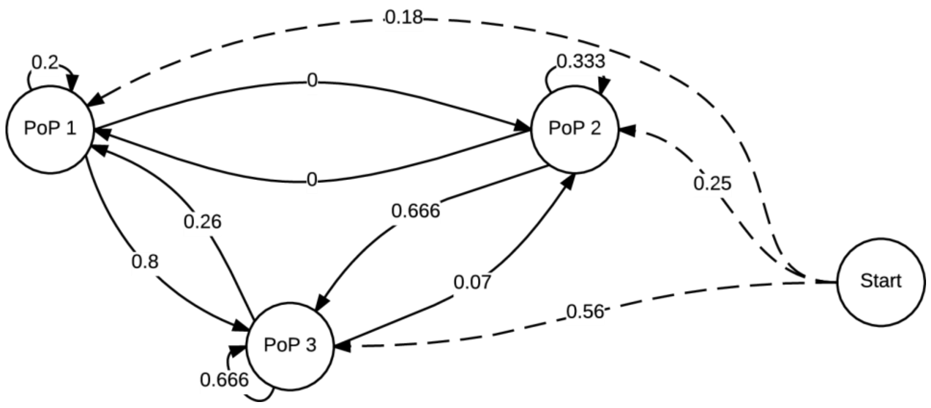


Figure 38: Transition probabilities between the PoPs as derived from the observed state transitions

2.4.8.2.6.2 Prediction of point-of-presence (PoP)

This subsection presents the results for machine learning employed for predicting the PoP value when the user is connected with the 4G network. The measurement set includes 40267 measurements for LTE for 8 users. The features selected for machine learning are day of the week, hour, and user ID value. The "InfoGain" algorithm indicated that day of the week and hour are amongst the most predictive values for PoP. There were 3 different PoPs with distribution of 56% (PoP 3, highest delay), 25% (Pop 2) and 18% (PoP 1). There was no missing data in this dataset.

A prediction task has been defined as a binary classification task, i.e., towards a prediction of the PoP. The machine learning method selected for this task is a classification tree (J48 in WEKA suite) and the analysis has been executed such that a stratified 60% of the data have been used for a training of a model, while the remaining 40% of the data for the testing. This has been repeated 10 times to get statistically significant results.

The “educated guess” accuracy value, i.e., when the prediction is made based only on the historical values of the PoP observed in the training dataset is 56%. The prediction accuracy when decision tree is employed and evaluated on the testing dataset is 97.13% (±1.24%). The confusion matrix indicates that the errors originate in miss-classification of the PoP 2 being classified as PoP 3. Such a misclassification would induce only the 1ms difference in delay, which may be negligible for most of nowadays applications. The advantage of this prediction method is that it is possible to predict, solely based on time features, the PoP the user will be attached to.

2.4.8.2.7 Conclusions

The aim of this work was to analyse the mobile network performance based on an extensive measurement study. In particular the analysis of performance limiting factors and their root cause was targeted. The data was collected during an extensive crowd-sourcing study and an additional targeted measurement campaign, generating reference measurements using a more extensive test set. From this data, a number of observations regarding the cellular network and its performance can be derived. The crowd-sourcing based measurements are well suited to derive general, location based network parameters like the cell size and density, as well as the available technologies and the respective signal strength. The network performance related parameters can mainly be derived using dedicated measurements and more complex measurement procedure.

The main findings are summarized along the research questions posed in the beginning for which the answers are outlined in the following:

How does the network configuration and management influence the cellular network performance? The analysis of the second data set shows that the network performance (i.e. RTT) is highly dependent on network configuration and management decisions. In the data sets, three different PoP were discovered. One of these is selected for each device connecting to the mobile network. The assignment was observed to be random but with unequal probabilities. The resulting RTT between the mobile devices and the server at TU Darmstadt also strengthens this observation. Due to the random assignment of the mobile devices to the PoP the performance experienced on the mobile devices is often sub-optimal. This performance also depends on the location of the server. Considering that a large number of servers and services are located near PoP 1, and only 19% of the traffic are routed using this PoP, the resulting performance shows a clear optimization potential.

Can the performance of the cellular network be predicted based on the available measurements? Analyzing the assignment of the mobile devices to the PoP provides a clear indication of periodicity. Given this a Markov Chain was derived to model the starting state probability and transition probabilities for the PoPs. This is also exploited to predict the assignment of the devices to the PoP. As time patterns were also visible in the Spanish measurements, discovering similar effects is probable. Applying a binary classification tree, a 97% accuracy predicting the correct PoP has been achieved.

What is the optimization potential in the given network configuration, and how can this be exploited? We verified a large additional RTT when using a sub-optimal PoP. An overhead of 5873% was visible in more than 57% of the measurements. This overhead is nearly constant for the full time of the PoP association. Automatic re-assignments are only carried out after a connection duration of 36 hours. Hence, the performance of the current connection can be considered stable. The performance does neither change with handovers in between cells. Still, forcing a re-assignment to another PoP is possible by disabling and re-enabling cellular data. Such, the performance penalty imposed by the sub-optimal PoP assignment can be mitigated.

Summarizing the above findings, the performance of a cellular network depends more on the network configuration and management than on the signal strength, time, cell ID or other parameters. From the measurements it was derived that the current configuration and PoP selection of the network is often sub-optimal. Still, a large optimization potential exists. Hence, it is suggested to modify the cellular operator's network to assign the devices to an advantageous PoP. As this also depends on the location of the remote server, a network with higher flexibility is required, allowing to dynamically selecting the optimum PoP for individual connections. An intermediate step reducing the latency of the cellular network for a large number of connections would be extending the infrastructure according to the most likely server locations. Then assigning the devices to the optimum PoP based on their usage patterns promises to considerably improve the network performance for a large number of users.

Future work will focus on increasing the number of measurements available for analysis and the simultaneous throughput measurement in different cells, to further detail the performance of the cellular network and its backbone. Furthermore, increasing the efficiency of the mobile throughput measurements with its particular requirements (fast, minimal traffic generated) will enable the wide-spread measurement of the cellular throughput and as such, increase the availability of data to allow an even more fine-grained analysis of the mobile network.

2.4.9 Test CO9: Evaluation of Bandwidth Allocation Optimization

The objective of this test has been investigating the improvements that prediction-based optimization could bring to mobile networks.

Within eCOUSIN we studied the limits of an omniscient system that could use the exact future information. Results of this first series of test have been discussed in D4.2 Sections 2.2.2 and 2.3.3.2.

Subsequently, in D4.3 Section 3.2.4, we proposed a resource allocation technique leveraging imperfect prediction. We showed that with our solution we could achieve almost optimal buffer under-run time and up to 25% resource savings. This translates in the possibility of increasing the network capacity by the same percentage with as few information as that described in [BUI14]. More recently, we extend the work of D4.3 in [BUI15b] including a thorough evaluation of the technique on trace obtained from real data.

Finally, we extended our analysis to the multi-user, multi-quality case [BUI15]. In this paper we showed that the average multimedia bitrate could be increased by up to 50% by maintaining an optimal buffer under-run time (no interruption in most of the studied scenarios). In what follows, we summarize the result section of the aforementioned paper.

2.4.9.1 *Anticipatory Quality-Resource Allocation for Multi-User Mobile Video Streaming*

Split, Sort & Swap (SS&S) has been developed to account for multi-user optimization and it is a direct extension of ICARO [D4.3]. The algorithm uses, first, a greedy approach to obtain a fast and feasible allocation and, then, iteratively improves the results by swapping resources among users and slots until no further improvements can be achieved.

This section evaluates the performance of SS&S and its result after *x* iterations (SS&S*(x)*) against the optimal solution (Optimum) and the unoptimized performance (Baseline). The baseline is computed assuming proportionally fair scheduling is providing each users *1/K*-th of the time. While the optimal performance is obtained with a standard numerical solver.

In each simulation, all traces are normalized so that the average capacity is $C = 1$ and, thus, equal for all users. In all simulation the buffer size is set to last at least 50 seconds at the maximum quality, so that a full buffer allow a user to playing the video without interruptions even if no download is made between two capacity maxima. Finally, each parameter combination is averaged over 50 runs and error bars are plotted for this averages at 95% confidence.

In order to systematically study various rate requirements for video streaming, we define two additional parameters: $\alpha \in [0, \infty]$ and $\beta \in [0,1]$. The parameter α is the maximum quality any user can obtain (e.g. $\alpha = 1$ means every user can obtain a bitrate the same size of the average capacity she can achieve). The parameter β instead is the fraction of bitrate needed to deliver the minimum quality (e.g. a user who fails in obtaining the minimum quality will experience streaming interruptions).

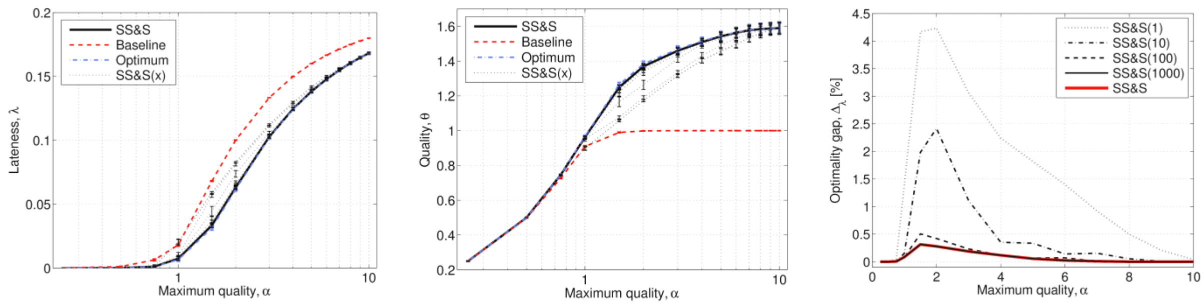


Figure 39 - Performance comparison among SS&S, optimal and baseline.

Figure 39 shows the first set of plots, that illustrate, from the left to the right, the average lateness λ, the average total video quality θ normalized over the average capacity and the gap between the results obtained by SS&S and the optimal.

The first plot is obtained with $\alpha \in [0.25,10]$ in logarithmic steps and $\beta = 1$. This setup is meant to study λ as a function of the ratio between demand and offer, thus no extra quality is considered. The results obtained by SS&S and the optimal are plotted as black solid and blue dash-dotted lines respectively and they are very close to each other confirming that SS&S obtains almost optimal performance. SS&S*(x)* performance are plotted for $x \in [1, 1000]$ as dotted black lines and they are increasingly close to the optimal performance with increasing *x*. Finally, the baseline performance is shown as a red dashed line. Notably, the baseline performance is always worse than the others and it is obtaining an average lateness more than twice (2.45) as long as SS&S when $\alpha = 1$.

The second plot, in the centre, is obtained for $\beta = 0$ with everything else unchanged. This set of experiments is meant to study the maximum average bitrate achievable varying α. Again SS&S reaches almost optimal performance and both solutions outperform the baseline by up to 60% and, starting from $\alpha \leq 1.5$ of as much as 25%. As in the previous graph, increasing the number of iterations reduces the distance from SS&S*(x)* to the optimum. Notably, both here and in the previous graph, a larger number of iterations is needed for $\alpha \in [1,3]$: in fact, out of this region a completely greedy solution is already good enough as the minimum requirements are either very low ($\alpha < 1$) and they can be greedily allocated to all the users or very high ($\alpha > 3$) so that only the user with the highest capacity is being allocated.

The third plot of Figure 39 shows the difference between the lateness obtained by SS&S*(x)* and the optimal $\Delta_\lambda = \lambda_{\text{SS\&S(x)}} - \lambda_{\text{Opt}}$ varying $x \in \{1,10,100,1000\}$. Again the gap is larger for fewer iterations and in the region $1 < \alpha < 3$.. Also, $x \geq 1000$ iterations are sufficient to achieve the best performance of SS&S, which, in turn, is very close to the optimal ($\Delta_\lambda < 5 \cdot 10^{-3}$). Finally, even with a single iteration the gap is smaller than 5% ($\Delta_\lambda \approx 0.043$).

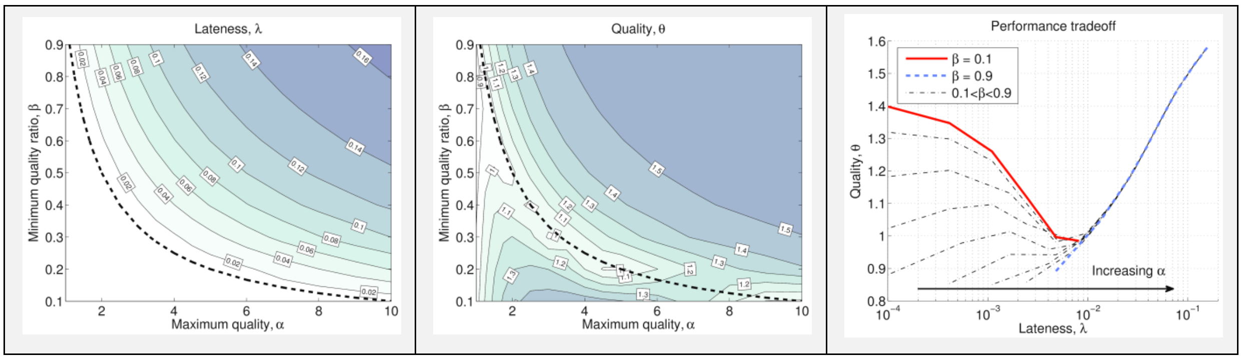


Figure 40 - Contour plots of the average lateness (left), average total quality (center) and trade off plot (right) between lateness and quality varying α and β.

The second series of plot in Figure 40 represents from left to right: contour plots of the average lateness, contour plots of the total average quality and the trade off between lateness and quality varying both $\alpha \in [1,10]$ and $\beta \in [0.1,0.9]$. In the first two plots a black dashed contour is plotted to mark the boundary between the region where minimum requirements for an uninterrupted streaming are lower (bottom-left part) or larger (upper-right part) than the average capacity.

The lateness results (left) are quite intuitive as below the dashed border SS&S mostly streams the video uninterrupted at the desired minimum quality. However, crossing this border causes an increasingly higher lateness up to 20% of the video duration. Conversely, the quality contours (center) are slightly more complex: in fact the quality increases both above and below the dashed border. The quality increase when the resources are scarcer (top-right part) is justified by the fact that the system is trading lateness for quality allocating only users that can obtain higher quality. Conversely, the quality increase in the lower-left part of the figure is obtained without almost no lateness (compare to the left figure): in fact, in this region the minimum bitrate is small compared to the average capacity, thus allocation computed by SS&S allow all the users to receive an uninterrupted stream at the required quality.

The trade off between lateness and quality is clearly illustrated in the right plot if Figure 40, where λ and θ are plotted on abscissa and ordinate respectively. The different curves are plotted for different β and each curve is obtained for varying α. Notably, the system is bound between $\beta = 0.1$ on the top-left part and $\beta = 0.9$ on the right. All the curves are quite close when $\alpha\beta = 1$. This plot can be used to estimate the expected system performance when adopting SS&S: for instance, in a system where the minimum requirements are 20% of the maximum quality (e.g.: 400 kbps and 2 Mbps) the second curve from the top can be used to decide how many users to allow in the system as a function of the desired average video bitrate; as an example, if C = 20 Mbps and the desired average video bitrate should not be lower than 1.5 Mbps, the user number should be $K \approx 16$, obtaining an average lateness lower than $\lambda < 10^{-4}$; more users can be traded for lower average quality, lower minimum quality or higher lateness.

2.4.10 Test CO10: Evaluation of the Network Visualization

Network visualization has been addressed over the whole duration of eCOUSIN and while ad hoc tools have been developed for very specific demonstrators, the most common tool used to visualize project results over maps has been the standard visualization tool provided with the ns-3 simulator. The main reason for such a choice has been that the tool was already offering trace oriented interface for creating animations and map overlays to illustrate any specific results or dynamic variation of metrics in the network.

Ad hoc solutions can be found in D4.2 Section 2.3.3.2 where a geographic map oriented tool has been used, D5.3 Section 3.1.3 , where a web application has been used to allow for dynamic data visualization over the web and D5.3 Section 4.2.2.5 where the ICN controller tool has been designed to offer a graphic interface to set up, control and monitor the demonstrator.

2.4.11 Test CO11: Evaluation of the Passive Measurement Module

The passive measurement module has been used in several tests over the whole project duration and eventually has been provided with lightweight active measurement functionalities to be able to compare active and passive measurement campaigns.

In our most recent paper [MICH15] we showed that passive and lightweight measurement technique can provide very good estimate of both users' achievable data rate in a mobile network cell and users' end-to-end throughput. In particular, we developed a technique based on packet train dispersion time, that is able to obtain an error lower than the 20% by using as few as 5% of the information required by traditional techniques.

In the technical work packages we addressed this topic in D5.3 Section 3.1.2 where we discuss the main results about the module, D5.2 Section 5.3.4 where we discussed how dense measurement techniques are paramount to obtain reliable network predictions and D4.3 Section 3.2.4 where we discussed how to move from frequent measurement to general prediction and, finally, resource allocation optimization [BUI15b].

2.4.12 Test CO12: Analysis of the Mobile Prefetching Study

2.4.12.1 *Introduction*

Prefetching is an efficient way to offload mobile networks by downloading content a user is likely to request in the future in advance. Thereby, load on the network operators can be shifted away from peak times to off-peak times and from cellular networks to Wi-Fi networks. The benefit of an efficient mobile prefetching system is two-fold. First, network operators can save costs as a consequence of less peak-hour traffic. ISP agreements, like burstable billing are raising extra costs for peak-hour traffic. Second, the mobile device user gets a high playback quality if a video has been prefetched. Furthermore, the device's battery and the mobile data volume are exceeded slower by performing prefetching over Wi-Fi. In this work, the analysis of the participation and behaviour of the Social Monitor user study, a joint work with Orange Labs, is conducted.

2.4.12.2 *Related Work*

There has been done work in the area of OSN-based prefetching before. However, to the best of our knowledge this is the first work investigating YouTube and Facebook with a specific focus on video prefetching so far. Additionally, the collected data not only contains social network information but also specific measurements of mobile device usage behaviour of individual users.

Paul et al. investigate the feasibility and performance of prefetching for Facebook [PAUL15]. The described but not yet implemented approach leverages the time a user spends looking on a video posted on Facebook as an indication for it being watched soon. This forecast of the user's intention to start the video can be used to set up the connection to the content server and download, e.g. the first chunk. This approach has the goal to minimize the initial start-up delay of a video playback as this is especially unpleasant for users. It has been shown that 1 second of start-up delay increases the abandonment rate by 5.8 according to a simple regression model. However, the avoidance of the start-up delay has been investigated in previous work, e.g. by [DIMA11, LI12, CHEN09, WANG11, KHEM12].The presented work differs by considering videos as a whole and expanding the time-frame for prefetching to multiple hours. Additionally, not only the first segment(s) of a video are considered but as much as the user might be interested in of the video.

Internet Service Provider (ISPs), and network operators have the possibility to collect information about the video consumption behaviour of a huge number of users. Guillemin et al., for example, analyze large datasets of YouTube requests observed in the mobile network of Orange, one of the largest mobile network operators in Europe. In [GUIL13a, GUIL13b] the cacheability of YouTube traffic for ISPs and network backbones is investigated. However, the network view is limited to unencrypted traffic. This work's dataset is not limited in this way as the information collected is retrieved from a customized video player on mobile devices and from the user feed leveraging the content provider APIs.

Khemmarat et al. investigate two prefetching approaches [KHEM12]. First, they use the related video list of YouTube to determine videos where a prefix should be prefetched. Videos contained in the related video list of a currently watched video are likely to be watched, as about 30 of the observed requests come from a video's related video list. Second, the search-results of YouTube were evaluated as a possible resource for prefetching candidates. Both approaches can only be applied in short-time as they require a user to search or watch a video on YouTube. One important evaluation performed in their work is the comparison of prefetching alone and in conjunction with a caching mechanism, locally on a mobile device as well as placed at a network proxy. As a result of their evaluation, the proposed prefetching mechanisms applied to a network proxy is more efficient than on a mobile device and caching significantly lowers the overhead of a prefetching scheme on the network. The work presented focuses on mobile offloading over Wi-Fi and therefore cannot rely on short-term prefetching as described above. This would increase the load on the network and raise battery consumption and costs for mobile device users because not every predicted video will be watched.

Ngoc Do, Ye Zhao, et al., have developed a prefetching scheme, implemented it in an app, performed a user study, and refined their scheme [ZHAO13, DO14]. They consider many metrics in their prefetching decision process, e.g. the social closeness of people in Facebook, the battery status, network connectivity, and the like count of a video. Nevertheless, their results suffer from a low number of study participants, namely 10. A further drawback is that they provided their study participants with an app which mimics the Facebook feed, but has a different look and feel as well as a different post order.

Sedhain et al. evaluate recommendations for video content on Facebook [SEDH13]. They investigate metrics like the number of exchanges messages, likes, comments, and group memberships. According to the authors, especially video can be precisely recommend based on the metrics they use. The work presented can be considered as an analysis that goes even beyond metrics from social networks and also considers the users daily routine, battery status, and his mobile network connectivity.

2.4.12.3 *App-based User Study*

An Android app named Social Monitor (available in the Google Play: http://tinyurl.com/socialmonitor) has been developed at the Technische Universität Darmstadt with the goal to distribute it in scope of a user study. To be compliant with the German privacy and data protection laws, potential participants were asked to accept the terms of service which explained in detail which information is collected by the app.

Three main functionalities are provided by the app to collect information from the participants' personal Facebook and YouTube feeds and monitoring the users' playback behaviour.

First, a crawler for the participants Facebook and YouTube feed was developed. This component allows monitoring changes regarding video posts which are presented to the user on both OSNs' news feed. This includes the appearance of new video posts as well as changes regarding the number of likes or comments of known video posts for 3 consecutive days, at maximum. Three days are a sufficiently large time span to observe the near-time increase of likes and comments of social video sharing, as about 34% of the social video views happen within the first day after publishing [BROX10]. At the same time, three days limit the amount of data which is collected at our participants' smartphones. By doing so, we spare the participants battery and mobile data cap.

Second, the app allows the participants to access their Facebook and YouTube feed by a customized Android WebView (http://developer.android.com/reference/android/webkit/WebView.html). This WebView was modified in a way that it looks like the native web page of the corresponding OSN. However, if a video playback is started, a special media player is invoked. Basically, the presented website is the native OSN's webpage modified in a way that videos are not played with HMTL5 or any other player but with our player, described in the following.

Third, a customized Android media player (http://developer.android.com/reference/android/media/MediaPlayer.html) has been developed to keep track of the duration a user watches a video. Furthermore, playback interactions like pause, stop, fast forward and skipping are tracked. Also buffering events occurring during the playback are collected. More information on the Social Monitor's software architecture can be found in [Koch15].

User Study Participants

The participants of the user study were recruited in two steps. In a first step, students and colleagues participating in our lectures, labs, seminars, and similar events were recruited during November 2014. In a second step, we collaborated with Orange, a large European telecommunication company located in France. They recruited about 20 additional participants amongst their employees. The participation duration of the users varies between a couple of weeks and a few months.

In Figure 41, the number of days for users participating at least for 7 days in the study is shown in conjunction with the number of users which participated for this time. CDFs for all participants, independent of their participation duration are shown for Facebook and YouTube separately in Figure 42 and Figure 43. It is remarkable that more than 50% of the users participated in the study for 2 weeks or more.

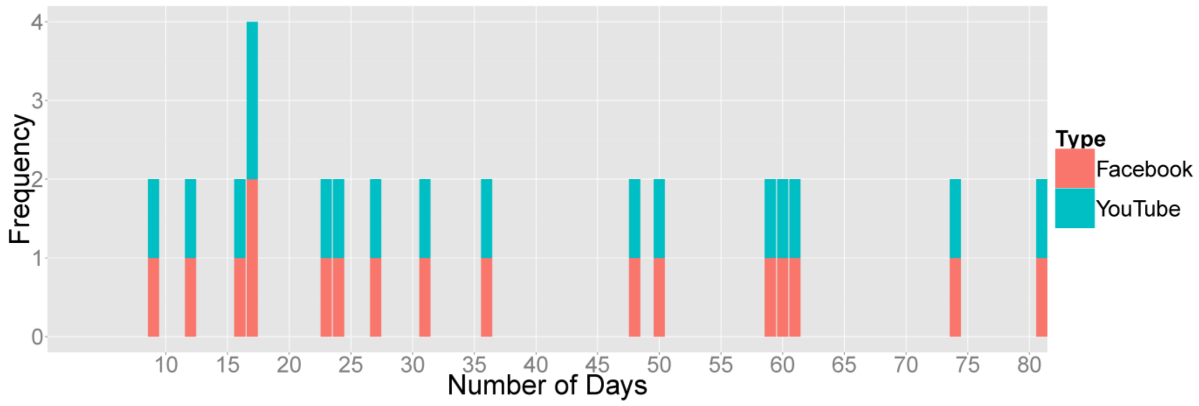


Figure 41: Overview of the time distinct users participated in the study

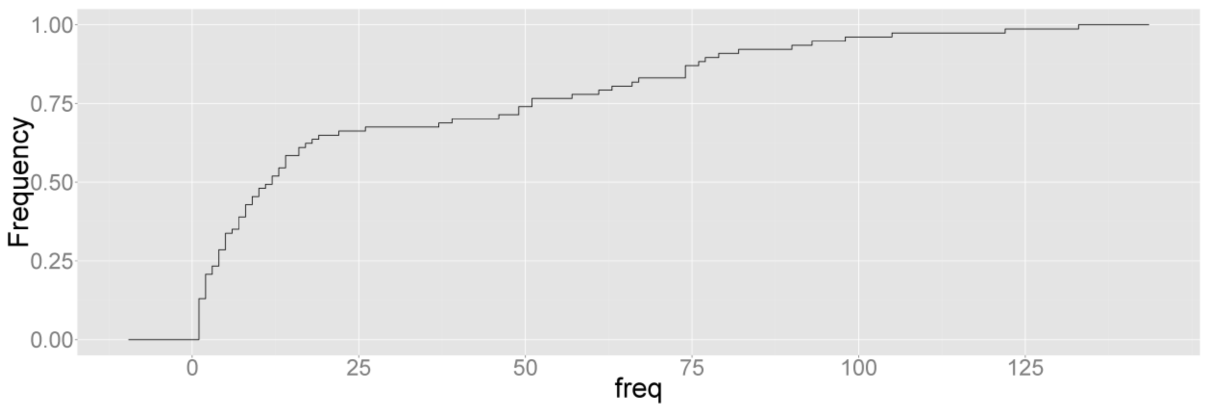


Figure 42: Number of days unique Facebook users contributed to the user study

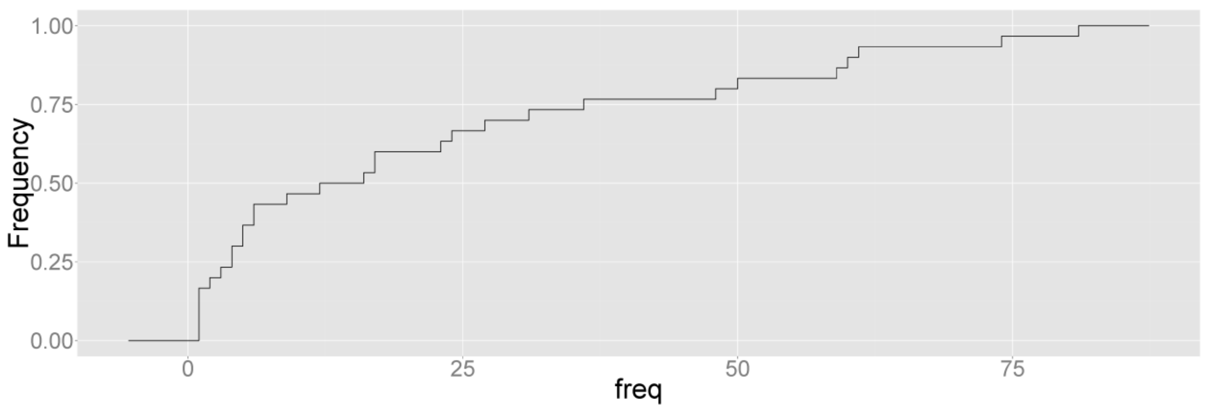


Figure 43: Number of days unique YouTube users contributed to the user study

2.4.12.4 *Dataset Description*

In the following, the different information collected during the user study is described. Five datasets can be distinguished. First, the Demographic Information dataset gives insights about demographic information of the study participants. Second, the Facebook dataset includes video posts which occurred on the participant's Facebook feed and if this video was watched. Third, the YouTube dataset contains information about the videos appearing on the participants YouTube feed, the videos watched mobile as well as on other devices. Fourth, the Media Player dataset describes how the users watched videos, containing different playback events and durations for all videos watched with the Social Monitor's video player. Fifth, the System Events dataset including information about the user’s connectivity and battery status during the time of their participation.

Demographic Information

Some users contributed to the study only for a short amount of time. Figure 41 gives an overview of the time spans of all users' participation time. The days of contribution per participant for all 77 Facebook participants is shown in Figure 42, whereas Figure 43 shows the same information for all 30 YouTube participants. For evaluations, a minimum of data points is needed to draw conclusions with a certain level of confidence. To this end, the data described in the following only applies for users which had installed the app for at least one week if not described differently.

Each participant was asked to fill out a survey at the first start of the app. Information gathered by this way are: place of residence, age, gender, and the data plan which the participant is subscribed to. During the study, 44 participants filled the survey, 6 of them were female and 38 were male. The following statistics have been calculated for these 44 users. The youngest participant is 10, while the oldest is 62. The mean age of the participants is 30.9 and the median age is 26.5. Considering the female participants only, the median is 43.0, and the mean is 43.2. For male participants only, the median is 26.5 and the mean is 31.8. Regarding the residence, most participants come from Germany (26), France (9), and India (6). Not every participant did fill the form as it was voluntary to avoid false information entered by annoyed participants. The mobile data plans used by the participants are shown in Table 15. Most of the participants are from Germany or France, as shown in Figure 44.

Table 15: Data plan subscriptions of all participants

Mobile Data Plan	#Participants
0 to 100 MB	**6**
101 to 500 MB	**13**
501 to 1 GB	**7**
1GB to 5 GB	**13**
More than 5 GB	**4**
Pay-per-use	**0**
Don’t know	**1**

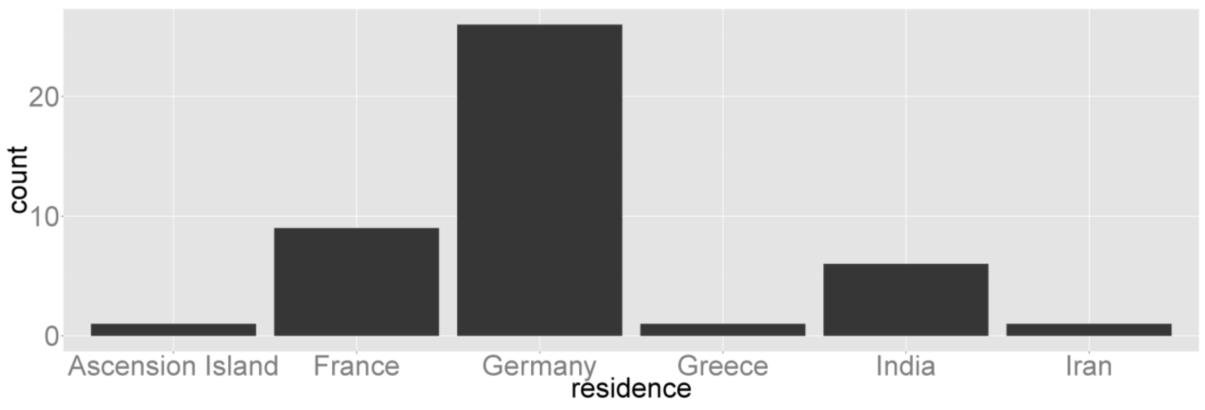


Figure 44: Distribution of the participant’s residence

Facebook

By monitoring video posts on the participants' Facebook feeds, the videos presented to the users are retrieved. From those videos, typically only a subset is watched. The order of Facebook posts presented to the user, however, is not deterministic and cannot be reconstructed from the data retrieved by Facebook's Graph API (https://developers.facebook.com/docs/graph-api). For each video post, the URL of the contained video, as well as the number of likes and comments are collected. Furthermore, each video post was monitored to track the increase of the number of likes and comments for 3 consecutive days. The app is able to retrieve all videos on a user's Facebook feed. However, only the videos which have been played back with the Social Monitor's video player are known to be watched by the user. As a result, nothing about the videos watched on devices where the app has not been installed is known, e.g. the user's PC. Also video playbacks performed with another video player or the native Android Facebook app cannot be captured. To this end, we informed our users to use the app's media player only during the user study.

YouTube

Compared to Facebook, every post on YouTube contains a video. The monitoring of those posts is performed by periodically crawling the YouTube channels to which the user is subscribed to. These videos build the YouTube feed of the participants, where new videos are presented. Beyond the channel on which the video was published, also the number of likes, comments and views are collected. Additionally, the subscribe events to channels and unsubscribe events from channels are tracked.

A huge benefit of YouTube in comparison to Facebook is the accessibility of the users' watch history, showing which videos have been watched in the past. This allows getting insights about videos watched on the mobile device and those watched on other devices, e.g. a PC. Videos which have been watched with the app's video player are not included in the YouTube history. A minor drawback of a user's YouTube history is that a video can only occur once at the YouTube history. If it is watched again the entry of the previous view gets a new date assigned and, as a consequence, represents only the latest view. As our crawler crawls the YouTube history periodically, e.g. every 15 minutes, a video which has been watched repeatedly will show only one entry in the history. However, if the video is watched multiple times, the date when it was watched changes. This allows identifying videos which have been watched repeatedly in different crawling cycles, since the watch time change indicates a new video watch event in the history.

As music is a very popular category of YouTube videos, also the play lists and watch later list of YouTube have been collected and monitored. For all videos added to an aforementioned list, a time stamp when it entered or left the list is collected.

Media Player

All videos played with the app's video player are monitored. This includes capturing of events. The specific events collected by the app's media player are listed and described in

Table 16. As our app is based on the native Android media player, these events capture all typical interactions by the user with the media player. All events which belong to the playback of one video are considered as a video session. If the media player is quit, a video session ends.

Table 16: Collected media player events

Event	Explanation
Session Start	The video player is started and displayed
Session End	The video player is quit
Video Played	The video plays back
Video Skipped	The user skips to a playback position
Network Status Change	The Internet connection type changes to {2G, 3G, 4G, Wi-Fi, none}
Device Rotated Horizontal	Video is shown in landscape mode

Device Rotated Vertical	Video shown on vertically adjusted
Video Paused	The video is paused by the user
Video Stopped Fin	The video playback is finished
Buffering Start	Playback buffering starts
Buffering End	Playback buffering ends

System Events

The collected system events provide information about several aspects of the participant's smartphone usage. For example, the time when the display was switched on and off is traced. By doing so, we could identify daily patterns of smartphone usage. The described values are collected each 15 minutes on each device at most. The collected event types and their corresponding values are shown in Table 17. For each event occurring, also the time stamp of the local device is collected.

Table 17: Collected system events

System Event	Values / Description
Battery percentage	The percentage of the battery level
Battery charging status	Charging, discharging, battery full, not charging
Battery power source	USB, on battery, ac
Boot event	Started up, shut down
Traffic cellular	The number of bytes transferred over cellular networks
Traffic total	The number of bytes transferred over cellular and Wi-Fi networks
Screen status	On, off
Network status	Mobile connected, 2G, 3G, 4G, Wi-Fi connected, no network connection

2.4.12.5 *Data Analysis*

The following section performs an in-depth analysis of the data collected during the user study. As stated by Figure 45

- 36 participants used Facebook only,
- 3 used YouTube only,
- and 14 participants used both social networks during the study.

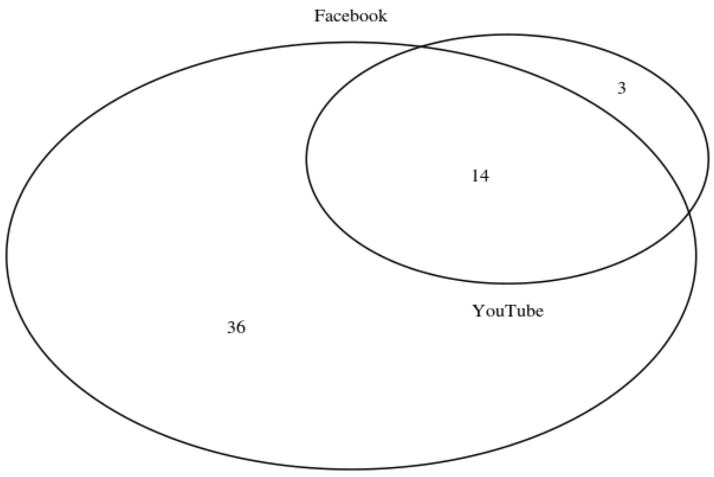


Figure 45: Veen diagram of participants using Facebook and YouTube

2.4.12.5.1 Facebook

Data from 50 users which logged in into Facebook with the Social Monitor was collected. 14 of these participants also used YouTube. In the following, different characteristics of the user behaviour are analyzed based on the data collected.

Posts

Compliant to existing studies in this area, most of the videos on the Facebook feeds are either from Facebook or YouTube. An overview of the videos watched from their Facebook feed, is given in Figure 46. Here the y-axis is scaled logarithmically. Overall, 436 Facebook videos, 394 YouTube videos, and 55 Dailymotion videos have been watched during the study on mobile devices. There were also videos watched from other platforms but they are not considered here as they only state a minority of video requests.

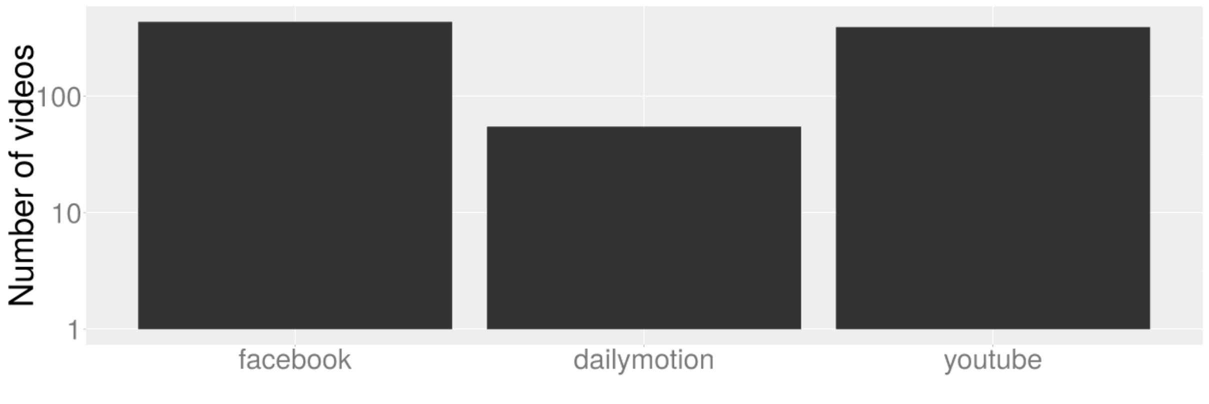


Figure 46: Videos watched from different video platforms from the user's Facebook feed

In Figure 47, an overview of the number of video posts shown on the participants' Facebook feeds is presented. Each dot represents a count of video posts for a distinct user for one day. The median count for all participants is 5. This indicates the potential for prefetching on a daily basis. A more detailed statistical overview of the number of videos per user feed and day is given inTable 18. Here, the heterogeneity of different participants becomes clearer. While the border between the first and the second quantile is 2, between the third and fourth quantile it is 14. This is the 7-fold. As a prefetching system is especially beneficial for consumers of a large number of videos, most of the participants would benefit from a prefetching system, e.g. the upper 50%. But also for the participants who have only a couple of videos, e.g. 1 or 2, the energy cost for prefetching over Wi-Fi is considerably smaller than a request over 3G. While this might drain the battery unnecessarily if the videos are prefetched and not watched, it has to be carefully considered if prefetching over Wi-Fi is reasonable for distinct users. More details about the users' energy resources are given in section 2.4.12.5.3.

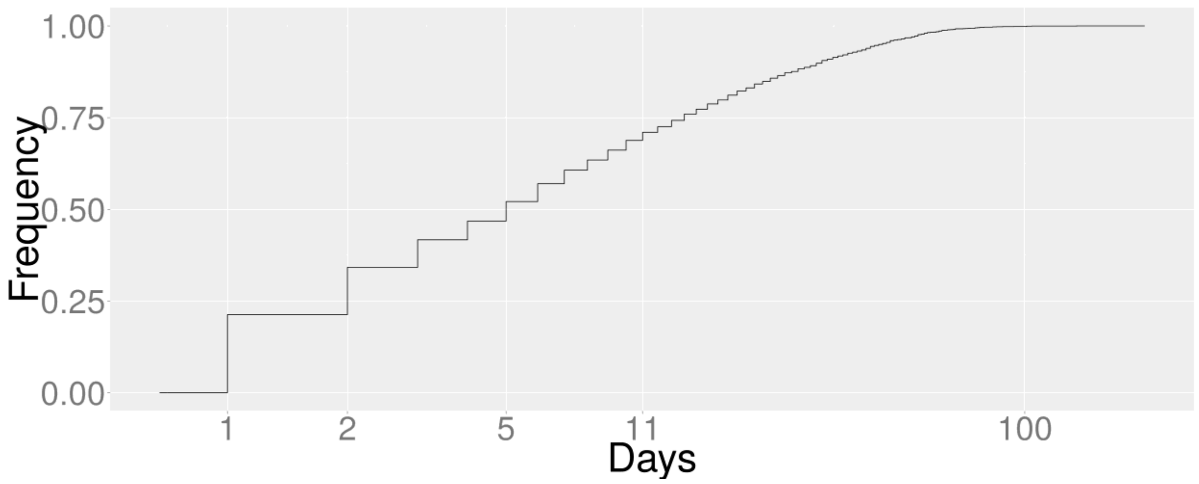


Figure 47: CDF of Facebook video posts per day for all users

Table 18: Number of videos per user feed and day

Type	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Number of video posts on Facebook per day and user	1.00	2.00	5.00	11.03	14.00	135.00

In the last paragraph, the numbers of videos shown to the user on his Facebook feed were analysed. In this section, we are going one step further and showing how many videos the participants watched mobile. The observations show that the median for videos watched mobile based on the Facebook feed is rather low by 2 videos. However, about 25% of the participants consumed 5 videos or more and one even 74 videos in a single day. A detailed statistical overview of the data is given by Table 19. The border between the third and the fourth quantile increases from the beginning of December to the beginning of February. One reason for this might be the Christmas holidays, in which fewer videos are watched. Considering the median of videos presented to the user and watched by the user, 2 of 5 videos are watched which results in 40%. As Wi-Fi offloading is about 10-times less energy-intensive on a Smartphone, even a prefetching solution which is able to download all videos in advance would result in a considerable benefit for the users. E.g., for a user representing the median: If the power consumption for one video gets assigned a cost of 1 then prefetching all videos would result in an energy saving of 90%: 5 – 0.1*5 = 4.5/5 = 90%.

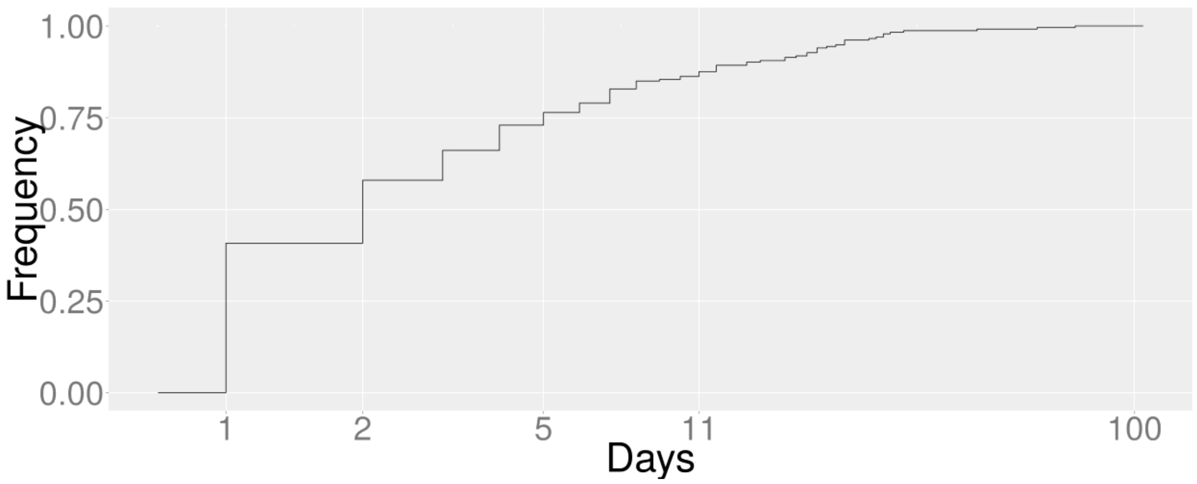


Figure 48: CDF of the number of videos watched from the Facebook feed, December'14 – February'15

Table 19: Number of video posts watched from Facebook per day and user

Type	**Min.**	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**
Number of video posts watched from Facebook per day and user	1.00	1.00	2.00	5.318	5.00	74.00

2.4.12.5.2 Facebook and YouTube

In the dataset, 14 participants used both social networks, YouTube and Facebook.

Figure 49 compares the number of distinct sources of videos on YouTube and Facebook. In the case of YouTube, sources are equal to channels. In the case of Facebook, there are more possibilities, e.g. persons, pages, and groups. Figure 49 shows that the participants request videos from more distinct sources on Facebook then they do on YouTube. The graphs shown apply for the subscriptions which were crawled from the participant's feeds. In other words, those are the videos presented to the user by his personal feed. The participants show different behaviour.

For a prefetching system, this result indicates that the number of potential sources that are interesting for a user is higher on Facebook than on YouTube. As each source is a potential object of interest for the user and requires separate statistical evaluation, the complexity for Facebook is higher than for YouTube. E.g., a prefetching mechanism needs to keep statistics about the number of shown and the number of consumed videos per video source to compute a certain probability (used as a measure for user engagement) if the user watches a video from this source again. This has to be recomputed each time a new video shows up on the feed or is watched. This results in a larger storage matrix. On the other hand, YouTube which typically has fewer sources a user might consume from, offers detailed meta information (https://developers.google.com/youtube/v3/docs/videos) for each video, e.g. the video's age, the number of views, and a category assigned to the video by the content uploader. This allows for more features on which a machine learning approach could identify the user's preferences. This would on the other hand result in a higher computational complexity. Therefore, the features used for such a computational intensive approach have to be carefully considered as the battery capacity is a scare resource on mobile devices.

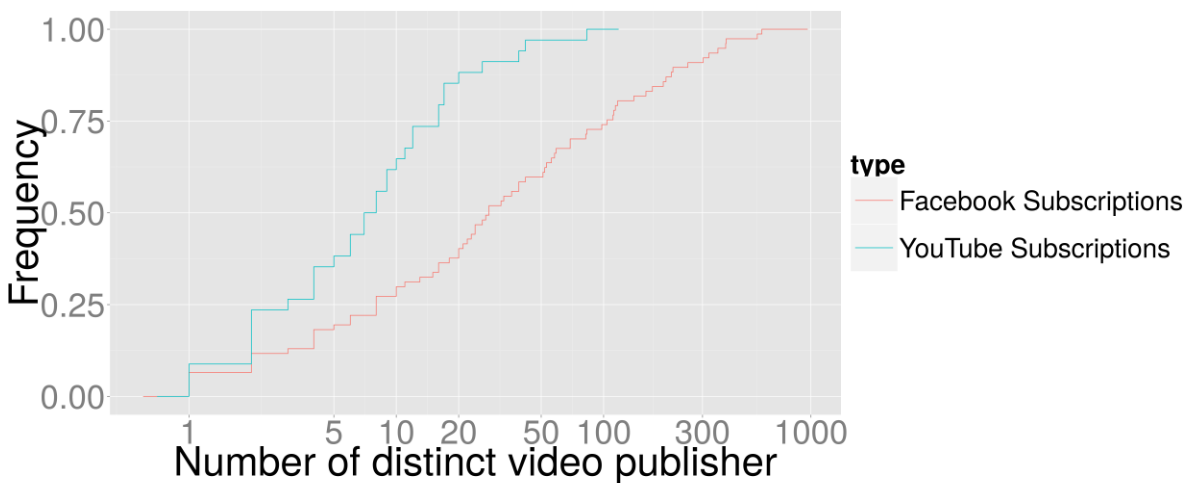


Figure 49: CDF of the number of distinct video publisher on YouTube and Facebook per day and user

A key question which has to be made by a prefetching approach is not only if a video is watched but also if it is going to be watched on a mobile device or on a fixed device. To give a first answer to this question, two heat maps have been created shown in Figure 50. Both heat maps are coloured according to the relative video start times for each user. Each line in the figures represents a distinct participant. These distinct participants are ordered the same way in both figures. Therefore, the left and the right side of the figure allow for a comparison of stationary and mobile video watching behaviour for each user. This figure contains data of all users which have contributed to the YouTube dataset independent of their participation time. The reason for this is that even a short contribution time of the participant allows gathering about 10 to 30 videos from his YouTube history. Considering the comparable small number of watched and shown videos from Figure 49, this is likely to capture several days of video consumption already.

On the left side of Figure 50, a heat map for the hours of the day when YouTube videos have been watched is shown. The heat map covers only videos included in the YouTube history and not those watched with the app's media player on mobile devices.

On the right side, a heat map for the hours of the day when a user watched a video with the app's video player is shown. This covers videos from Facebook and YouTube as well which have been watched on a mobile device.

It is worth noting that the watching pattern of mobile videos seems to be sparser than the non-mobile pattern, for almost all users. Also, for the majority of users there is a burst of mobile video request covering only a few hours. For a prefetching system, this insight indicates the predictability of future mobile video watches of the users. As this pattern tends to be relatively stable over time, the mechanism can precisely predict the video watch opportunities for which prefetching videos would be beneficial. Furthermore, this information in combination with the network connectivity patterns of the participants allows for the delay of the prefetching process if there are better opportunities, e.g. a Wi-Fi connection in the time between the video was posted on the corresponding OSN feed and the time when the user is likely to watch videos mobile.

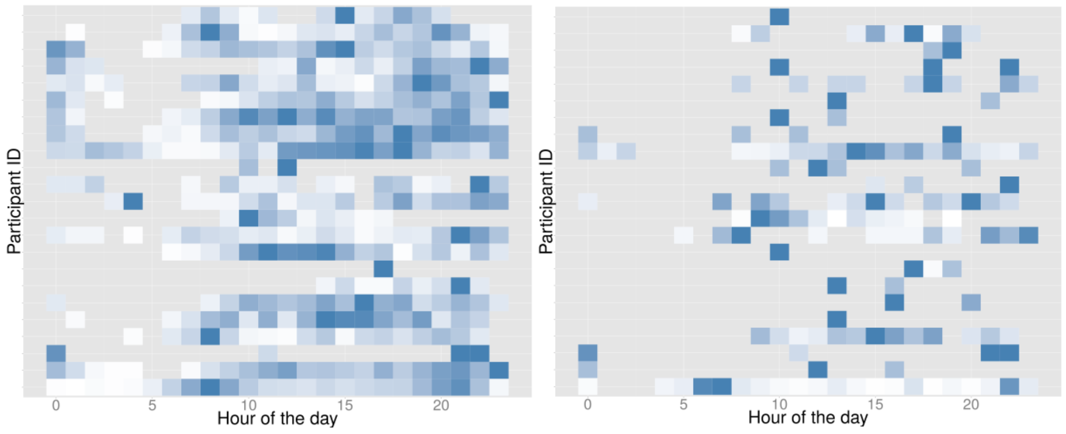


Figure 50: Heat map of the relative frequency for videos watches per user and hour of the day, YouTube fixed & mobile (left) and mobile Facebook & YouTube (right)

Daily Usage pattern

The time which is left between the publishing and the consumption of a video is essential for prefetching algorithms.

If the time span is large enough then moments when the user is connected to Wi-Fi or is charging his battery can be awaited to prefetch. In the following, the daily usage pattern of 30 of our users is described in detail.

Cacheability of Videos

Here we want to give an insight about the repeated requests of video for individual users for Facebook video requests. The numbers of unique and repeated requests for distinct users have been computed and are presented in the CDF shown in Figure 51. The same has been done for repeated requests for the videos of all participants. However, the CDF looks the same which proofs the heterogeneity of our participants. Instead of showing the different CDFs, a table of the most interesting values is given by Table 20. Here, the number of requests is evaluated for Facebook and YouTube in two ways: First, we investigated the number of requests for each video per user. Second, the number of requests for each video is calculated without regard which user issued the request. For each column, the median, mean, and maximum value is given. Since most requests were issued only once, even the border between the third and fourth quantile would be 1.

About 25% of the video requests are repeated requests and could lead to energy and bandwidth savings for the users if they were cached. A prefetching mechanism would save videos on the client device and can easily allow for the caching of video files to leverage the benefits. Especially for the videos which are watched many times, e.g. music videos, the benefit would be significant.

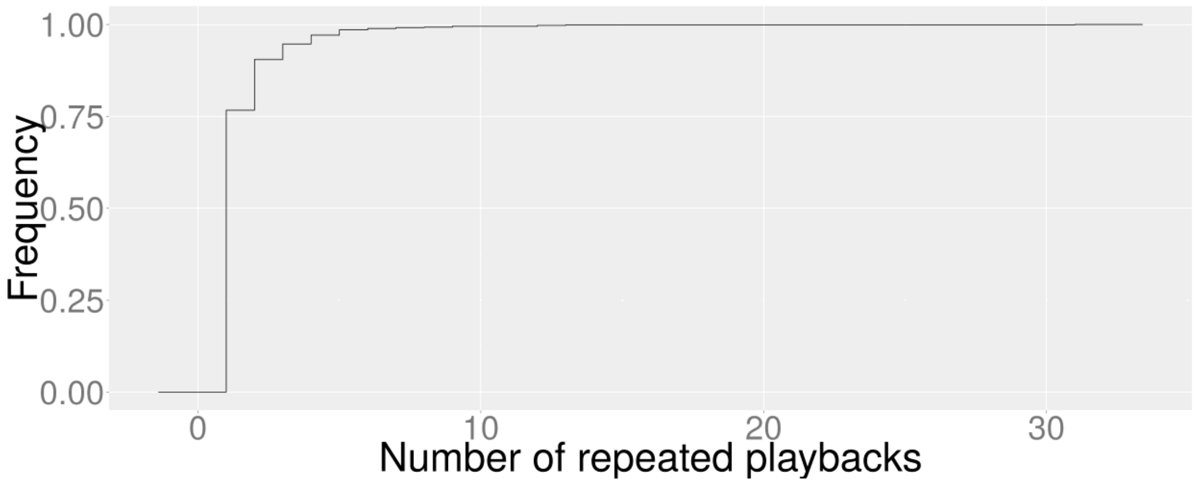


Figure 51: CDF of the request count of a unique user for each video

Table 20: Statistics about repeated video playbacks for each video for Facebook and YouTube

	Facebook per user	**Facebook all users**	**YouTube per user**	**YouTube all users**
Median	1	1	1	1
Mean	1.489	1.57	1.293	1.336
Max	31	40	13	21

Media Player

Different playback events of the videos played by the Social Monitor media player were collected. This allows observing the user behaviour with regard to the watching duration, if he jumps forward in a video, or if he re-watches certain episodes of a video.

Partial Video Views

An optimal prefetching approach would not only try to figure out the videos a user might watch but also how much of this video a user is likely to watch. The video player data set has been investigated to uncover the differences between videos watched on YouTube and Facebook. The CDFs presented in Figure 52 shows clearly that videos on YouTube experience longer playback sessions then those watched from Facebook. This result shows that there is a general tendency for Facebook videos to be watched only for at most 1 Minute in about 80 of the cases whereas YouTube videos are watched for about at most 10 minutes in 80% of the cases. Therefore, the data volume caused by YouTube video streaming is, on average, about 10 times the duration of video streamed from Facebook. Assuming the data volume caused per seconds between both OSNs is similar, the correct prediction of a YouTube video is more beneficial than for a Facebook video. This statement covers the average user and not every distinct user indicating the potential of prefetching for videos from both OSNs.

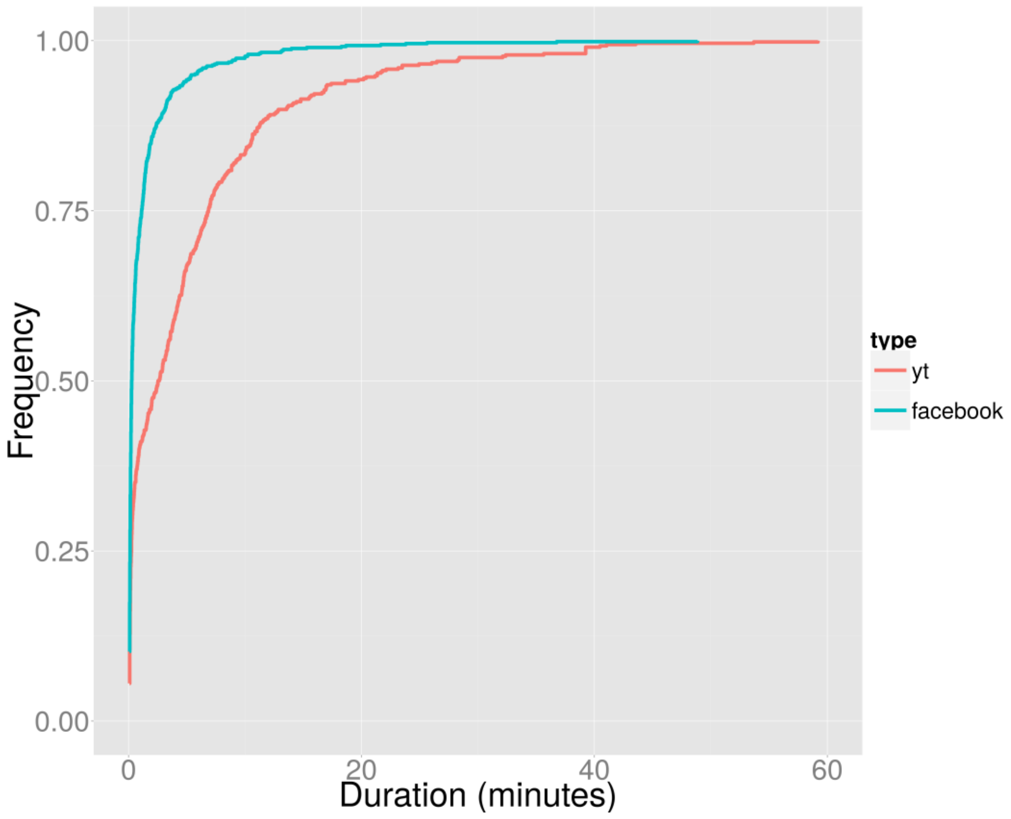


Figure 52: Number of seconds videos are watched on Facebook and YouTube

2.4.12.5.3 System Events

Energy Consumption

Figure 53 and Figure 54 show the quantiles of our participants for 5 consecutive days between February 16th and 20th 2015 for week days and for week end days respectively.

The black line connects the medians of the battery status and thereby shows that most participants tend to charge their device overnight. The median of the battery status is most of the time over 50% but for some users also battery status beyond 20% is observed. These situations occur often in the early night hours, e.g. 10 PM. Generally, the battery status is on average about 60%. This allows arguing that a prefetching approach deployed to the devices would have enough energy left to work well and not drain the battery of the users. This, of course, depends on the number of videos to prefetch. But as shown previously, the median number of videos on Facebook is about 5. As a result even an easy prefetch-all policy would not drain the battery in general. However, such a policy is not optimal but an option in the cold-start phase where the prefetching algorithm has no information about the user habits and cannot optimize its policy. The results indicate that a prefetch-all policy is a preferable solution compared to a prefetch-nothing policy for most users. The reason is that prefetching over Wi-Fi is about 10-times more energy efficient. Therefore even a precision of 10%

would result in no drawbacks regarding the battery and introduce benefits regarding the usage of the mobile data plan and the reduction of buffering events during the playback.

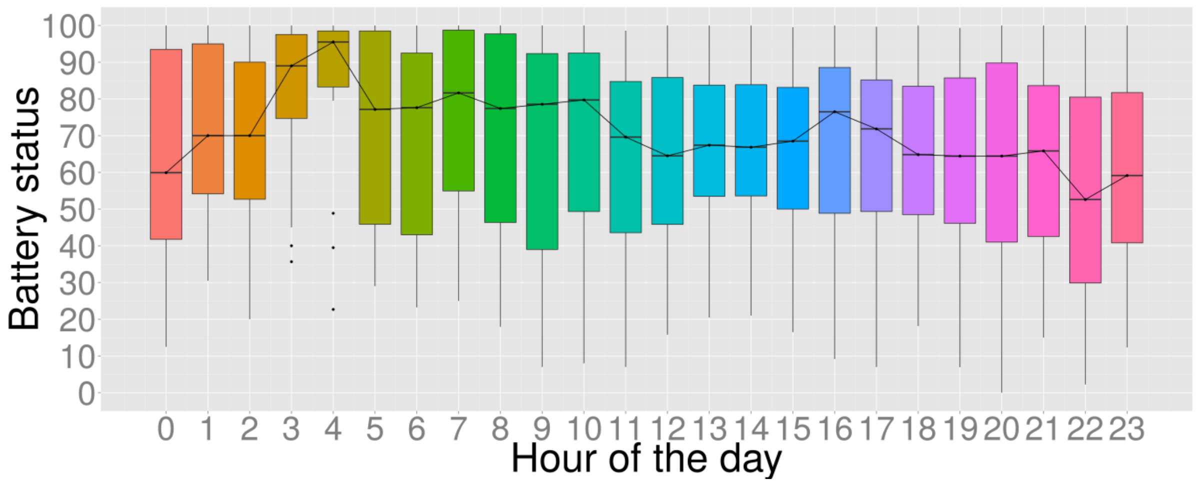


Figure 53: Battery quantiles for week days

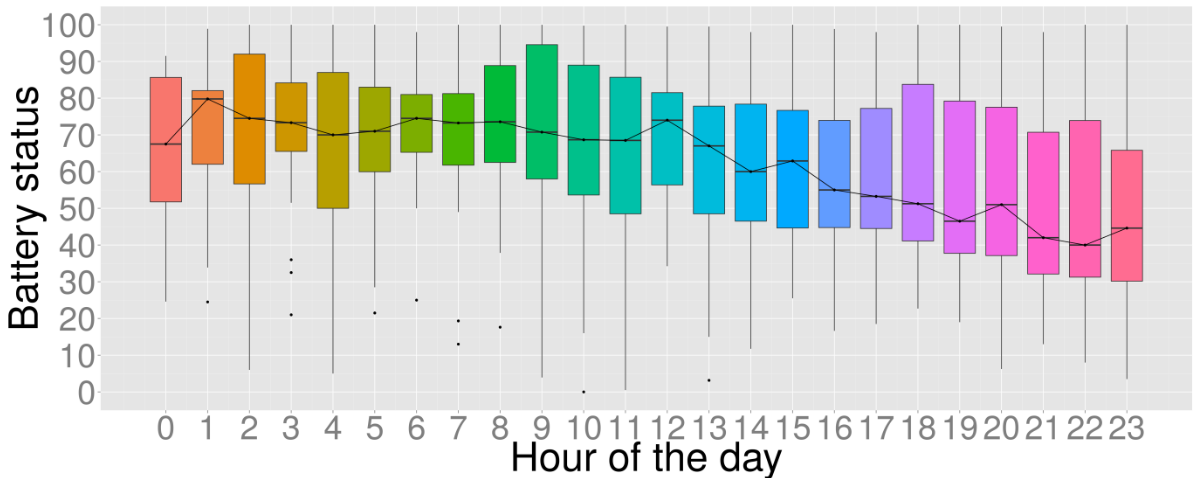


Figure 54: Battery quantiles for week end days

Figure 55 shows a heat map of the hours when distinct users are charging their device. This map is created by counting the hours of the day when users charge their device. After that, this sum is divided by the maximum value of counts per hour of the day. By doing so, the count values for each hour of the day are normalized on a per-user-basis so that the values are between 0 and 1. The same normalization method is applied for all heat maps.

Figure 55 and Figure 56 show that every user shows different usage characteristics. The participants where ordered according to the time when most events occurred(hot spots). The darker the blue colour of a segment for a distinct hour is drawn, the more evens have been observed at this hour for a unique user.

The results underline that a prefetching approach cannot be generalized for all users, but have to adapt to unique users and their unique habits. This can be achieved by an appropriate machine learning approach which is not in the scope of this analysis.

The characteristics of charging events collected during the study are less deterministic than for display on events. However, the steadier a user characteristic is over time, the better it is suited for a prefetching prediction. A finding of this study is that the display on activity is steadier over time than the charging behaviour. As display on activities indicate potential times when a user will watch a video, this characteristics provides valuable input for a prefetching mechanism. Deadline defining when the content has to be downloaded on the user's device at latest determine if the download should be performed immediately, e.g. over a more costly Internet connection like 2G or if there is appropriate time till the video is likely being watched so that expected Wi-Fi connections can be waited for.

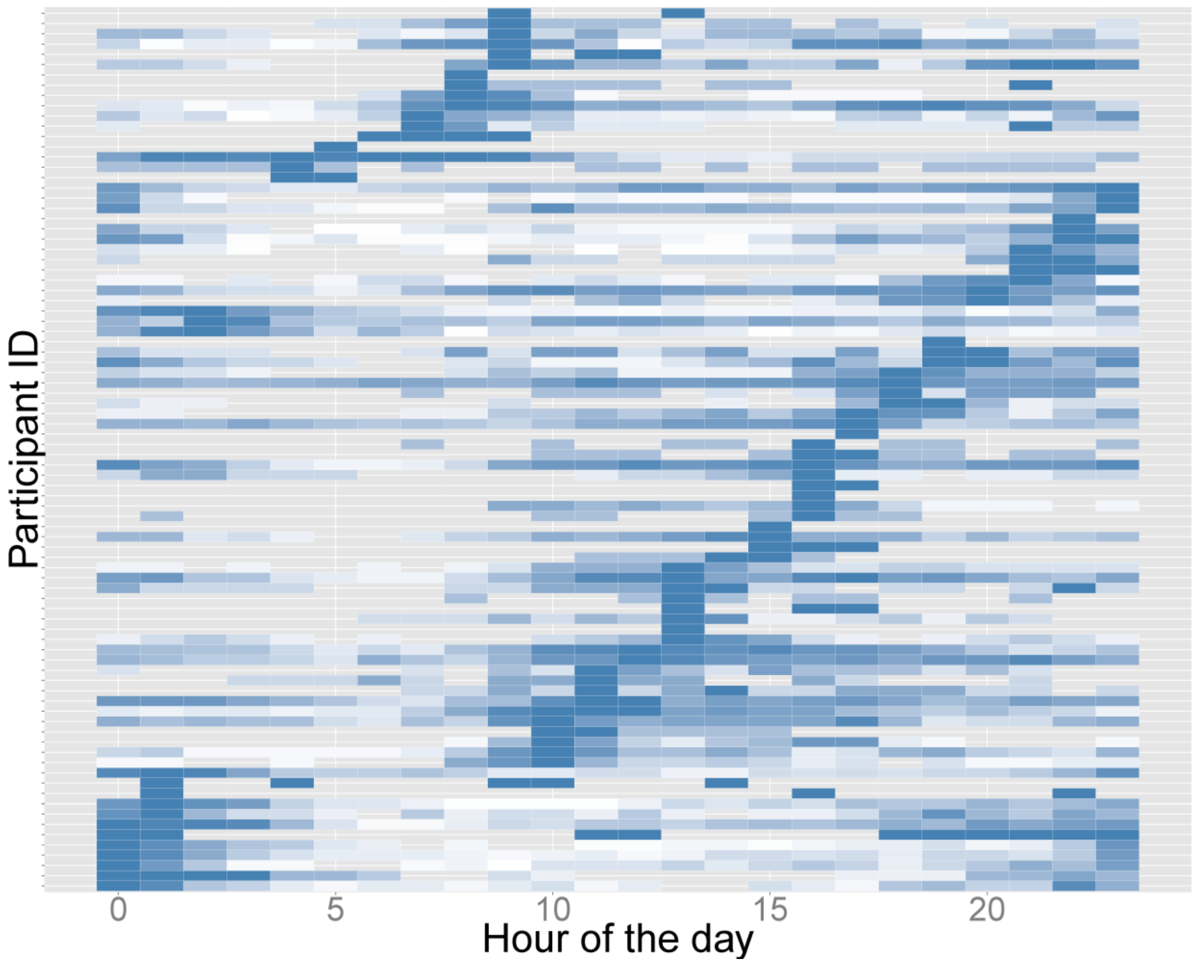


Figure 55: Heat map of the normalized number of charging events for distinct users

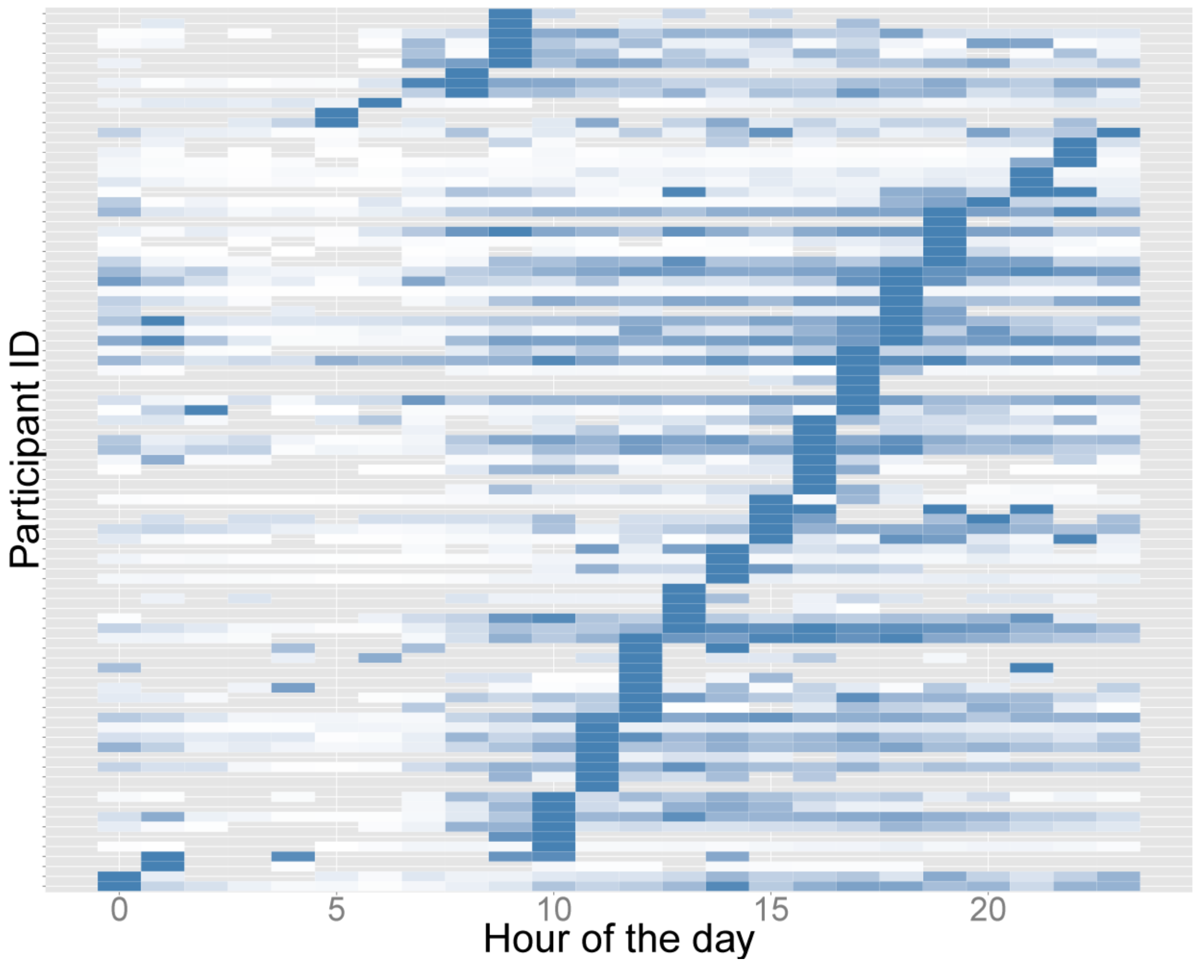


Figure 56: Heat map of the normalized number of display on events for distinct users

2.4.12.6 *Conclusion*

In this section, the key findings of the analysis of the mobile prefetching study and their meaning for a mobile prefetching mechanism based on Wi-Fi offloading are described.

Based on a large participant basis, it has been shown that the mean value of video posts on a Facebook feed is about 11 (median: 5), while the amount of videos watched from this feed is on mean about 5 (median: 2). This indicates the need for an optimal solution which can predict the subset of videos which is likely watched by the users.

One essential task for a prefetching mechanism is to keep track of the videos offered and consumed by the users and their corresponding features. It has been shown that the number of distinct video sources per user is significantly higher on Facebook than on YouTube. This leads to a larger storage capacity needed for a Facebook-tailored prefetching approach. On the one hand the storage complexity would be larger for Facebook, but on the other hand, the features offered for YouTube videos allow for a more computational intensive prediction mechanism to determine the most relevant videos to the user. In a further step, it has been shown that contrary to this complex prediction mechanism, a larger gain can be achieved as most videos on YouTube are watched much longer as those on Facebook. Thereby, they cause typically larger energy and bandwidth costs.

Besides finding the relevant videos for prefetching, the time when the user probably watches the videos is important. The time span between publication and consumption allows waiting for a Wi-Fi connection to download and thereby offload the video from Wi-Fi instead of the mobile network which would consume the valuable mobile data plan of the user. Our findings indicate strong hot spots in time where users tend to watch videos mobile. This pattern allows for much more accurate prediction then non-mobile views and thereby ensuring that the predicted videos can be downloaded timely. According to the data, also some false positives are not a problem for the battery as the average battery status is quite high, especially at night and in the morning when, e.g. news videos are published.

A general observation made is that our user study participants are behaving heterogeneously in many aspects, e.g. when they use their smartphone, when they charge their smartphone, how many videos they get presented and consume but also with respect to their mobile data subscription plan. An optimal prefetching mechanism would learn from the user's behaviour over time to adapt its behaviour and increase its performance over time.

The results stated are key findings for the implementation and the deployment of a prefetching mechanism, e.g. by large content providers like Facebook and YouTube. However, the content portals must ensure to own the rights for letting the user download the content and not only stream it. Furthermore, statistical information if the video has been played and for how long have to be sent to the content provider so he does not experience a lack of statistics for the provided content. Also the exploitation for models and simulations to test different prefetching policies is planned in further research based on the findings of this work.

2.4.13 Test CO13: Analysis of the D2D Traces

In communication networks, the delivery of high-resolution content to a consumer device is a major challenge. Different strategies for content delivery exist, like client-server, IP Multicast or Content Delivery Networks (CDNs). However, many state-of-the art approaches for content delivery only insufficiently reduce load of the access network and avoid its congestion. Congestion avoidance is especially crucial for mobile infrastructures, as the shared wireless medium is a transmission bottleneck.

Device-to-Device (D2D) content delivery is an emerging approach, where consumer devices (e.g. smartphones) transfer content between themselves, instead of receiving it over a mobile operator's access network (BS). This way, the access network is offloaded from popular on-demand content.

However, D2D content delivery has limited effect in certain situations. To retrieve content via D2D communication, a high content popularity is required, as well as a sufficient device density in proximity. Only the combination of both aspects makes it likely that there is a device in proximity serving the content.

The question about the potential of D2D content delivery under certain circumstances was not yet answered. In this work, we evaluate position and request traces of a major European content provider to provide an answer. The model applied here was elaborately described in D3.3 (Section 2.7). This section now explains the final setup used, and shows and discusses the results.

2.4.13.1 *Related Work*

An attempt to evaluate opportunistic communication with real-world measurements was conducted by Han et al. [HAN10]. Here, content is delivered by the mobile base station to a small subset of mobile subscribers (the target set) in a first step, while these nodes propagate the information to the entire set of interested nodes afterwards. Their work focuses on the selection of an optimal aforementioned target set. Han et al. put an increased focus on evaluating the selection strategies using mobility traces. Unlike in our work, the traces were obtained from restricted groups, like the Haggle [CHAI07] and the Reality Mining project [EAGL09]. Besides the smaller user base of the traces obtained by both projects, a major part of the devices participating has likely belonged to students and university staff which took part in the study. The devices have actively been collecting mobility trace data. The authors also discuss available incentives to participate in the target set.

A data collection method very similar to the one used in this work was used by Shafiq et al. [SHAF13]. The anonymized dataset contains usage data of 100 million customers of a large mobile ISP of the United States. The data was collected at the GGSN nodes at the time of various highly crowded events in metropolitan areas. However, the work focused on evaluating radio resource allocation and opportunistic connection sharing rather than D2D content delivery, requiring a different set of metadata, such as performance-related information.

2.4.13.2 *Evaluation Setup*

The input location data of mobile devices were collected in the south of France (the area of Marseille).The data collection started on March 4thand ended on March 19th 2015 (both at midnight), thus lasting for 15 days. A more detailed description of the data format that was used was described in D3.3. The dataset was comprised of ca. 450 million location UID (anonymous location) samples of 1.2 million anonymized users.

For every device in this dataset, a sample was generated periodically every 3 hours, whenever there is a handover, and upon network login. When the device does not change cells, it would be assumed offline by our proximity model for up to *3h-sampleTimeTolerance,* although it is just idle. To fix this issue, upon a sample, we added artificial samples in the same cell for additional 3 hours (in the distance of sampleTimeTolerance), but not longer than a possible following original sample.

Because of privacy concerns, we did not track the content consumption behaviour of the users; instead, we applied a content consumption model. According to the model, a fraction of randomly picked devices started requesting a piece of content between March 5th and March 18th. For every particular user, the actual probability of the request was equally distributed inside the aforementioned time interval. The fraction of users requesting content was parameterized with p_{req}. The applied consumption model especially describes the delivery of e.g. a software update for a mobile platform like iOS or Android, but may be valid for other consumption behaviour, as well.

The evaluation was conducted on a Linux machine with 96GB of RAM. The evaluation software was written in Java 7 and run in the Oracle JRE.

2.4.13.3 *Evaluation Results*

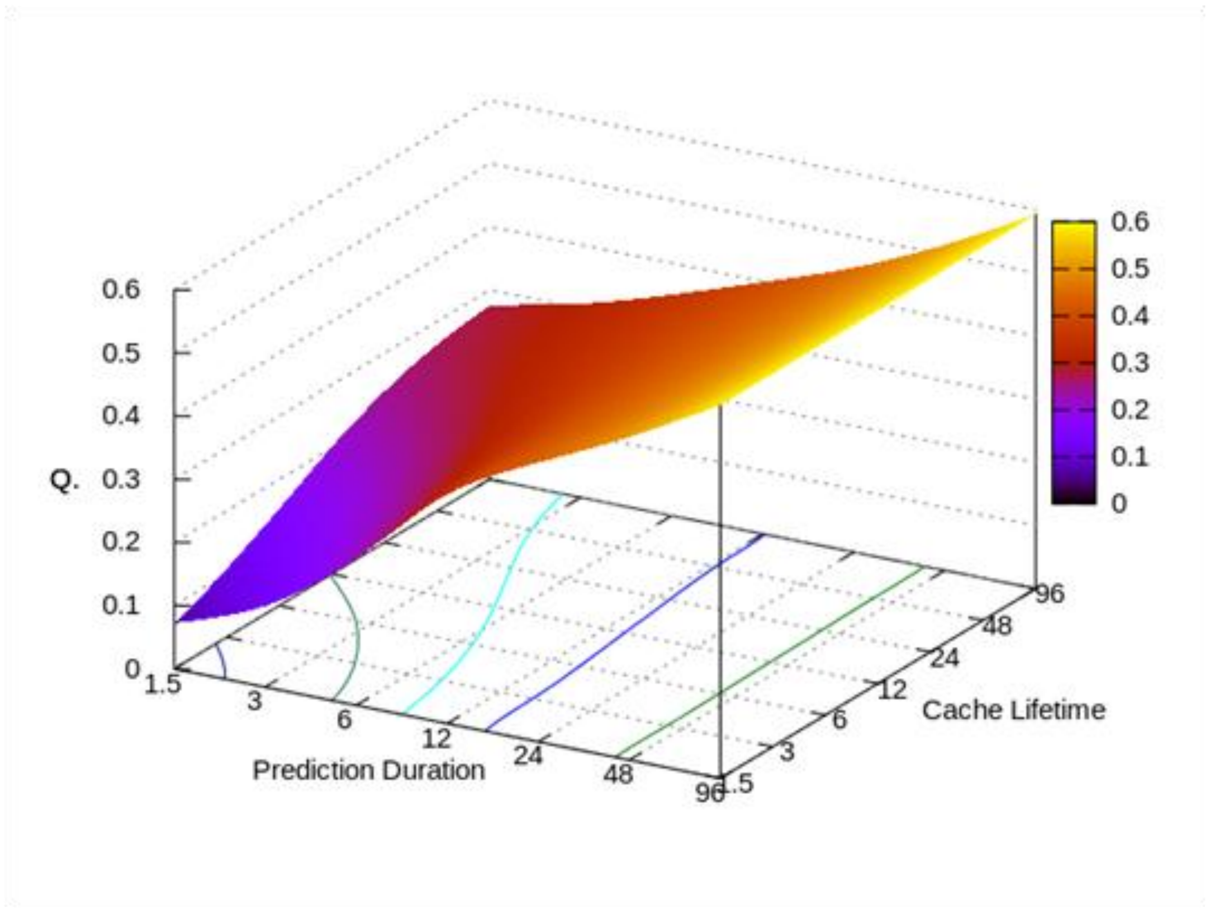


Figure 57: The quota of requests that can be served via D2D content delivery, under different parameters: prediction duration (tp) and cache lifetime (tc), for p_{req}=0.1. For better visibility, the plot was spline-interpolated between 7x7 samples.

Figure 57 shows the quota of the requests generated by the content consumption model which could be served via D2D content delivery (further called *D2D quota*). First of all, it can be seen that for p_{req}=0.1, a fundamental ratio of requests can be served via D2D in general (around 10%-60%).The D2D quota is depicted in relation to the main parameters of the model used for the evaluation. Especially the prediction duration *(t_p)* has significant, positive influence on the D2D quota. The cache lifetime *(t_c)* only appears to have an influence with the prediction duration set below 12 hours. As explained later, this is likely being caused by daily commuting cycles of the device's users, as well as devices powered off or being absent from crowds over night.

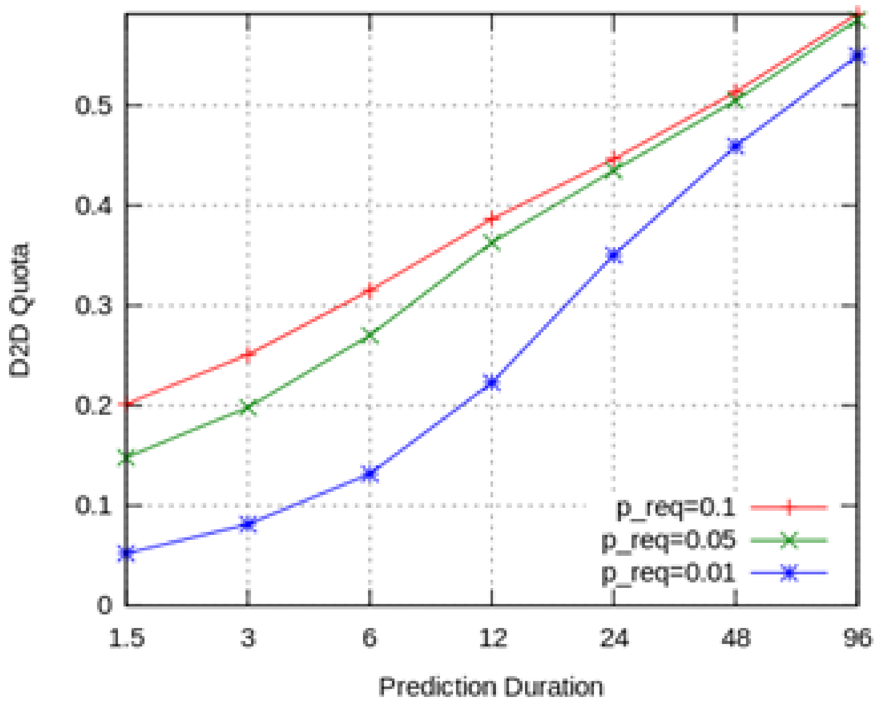


Figure 58: D2D quota for various prediction durations and probabilities of request (preq) for a device in the scenario.

A first finding is that if we have a mechanism available being able to predict at least the next 12 hours, we therefore may reduce the cache lifetime to a minimum or may even switch off further caching after content consumption. However, if such a prediction mechanism is not available, we should enable long (altruistic) caching instead, and for this, we require incentives for the user.

Figure 58 depicts the D2D quota in relation to various prediction durations and request probabilities (p_{req}). Here, it can be observed that - although a low popularity results in a low D2D quota - with increasing prediction duration, the D2D quotas of content items with different popularity converge.

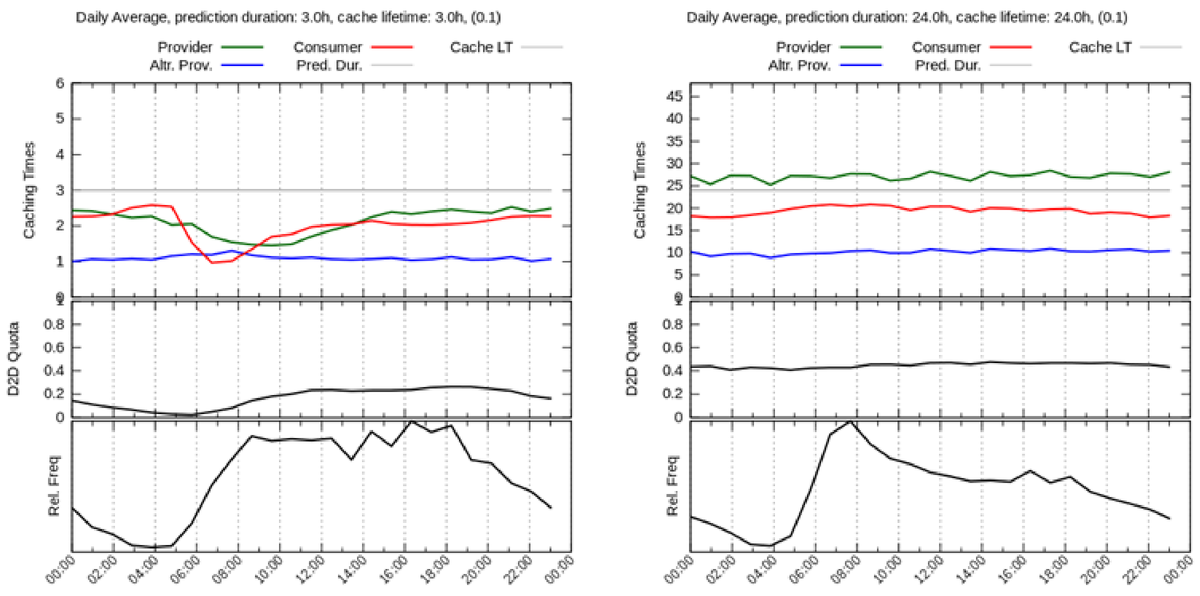


Figure 59: Average values over the day. In the upper graph, for content requested at the specified time, the times it resides in the provider's and consumer's cache is depicted. Below, we can see the D2D quota of requests made at the specified time. In the bottom, the relative amount of D2D exchanges at the given time is shown.

Figure 59 shows various average values over the day. The upper graph shows measured caching times. The green line is the whole time the content has spent in the provider's cache until it is exchanged to the consumer, while the blue line only includes altruistic caching time (the time after the provider's content consumption until the exchange with the consumer). The red line depicts the time the content has spent in the consumer's cache before consumption.

In the graph below, we can see the average D2D quota regarding requests that occurred at this time of the day. In the left figure (prediction duration = cache lifetime = 3h), we can see that the D2D quota significantly drops at night time. Here, neither the prediction can look more than 3 hours ahead, nor can the cache keep the content longer than 3 hours afterwards. Between 06:00 and 08:00, the D2D quota increases again. The consumer cache time drops at the same time of the day, which means most of the content requested was exchanged only a short time ago, likely during increased commuting in the morning. Because of the small prediction and cache time, this content could not be exchanged the day before.

The overall D2D quota is not as good as it is in the right part of Figure 59, where the prediction duration and cache lifetime was set to 24h instead. Here, the overall D2D quota more than doubles, and is especially better over night, caused by the possibility to fetch content the day before. Also, caching times do not significantly differ during the day. It can also be observed that with increased prediction and caching deadlines, more D2D exchanges (bottom graph) tend to happen in the morning, as the longer prediction times always exploit the first possibility to obtain content.

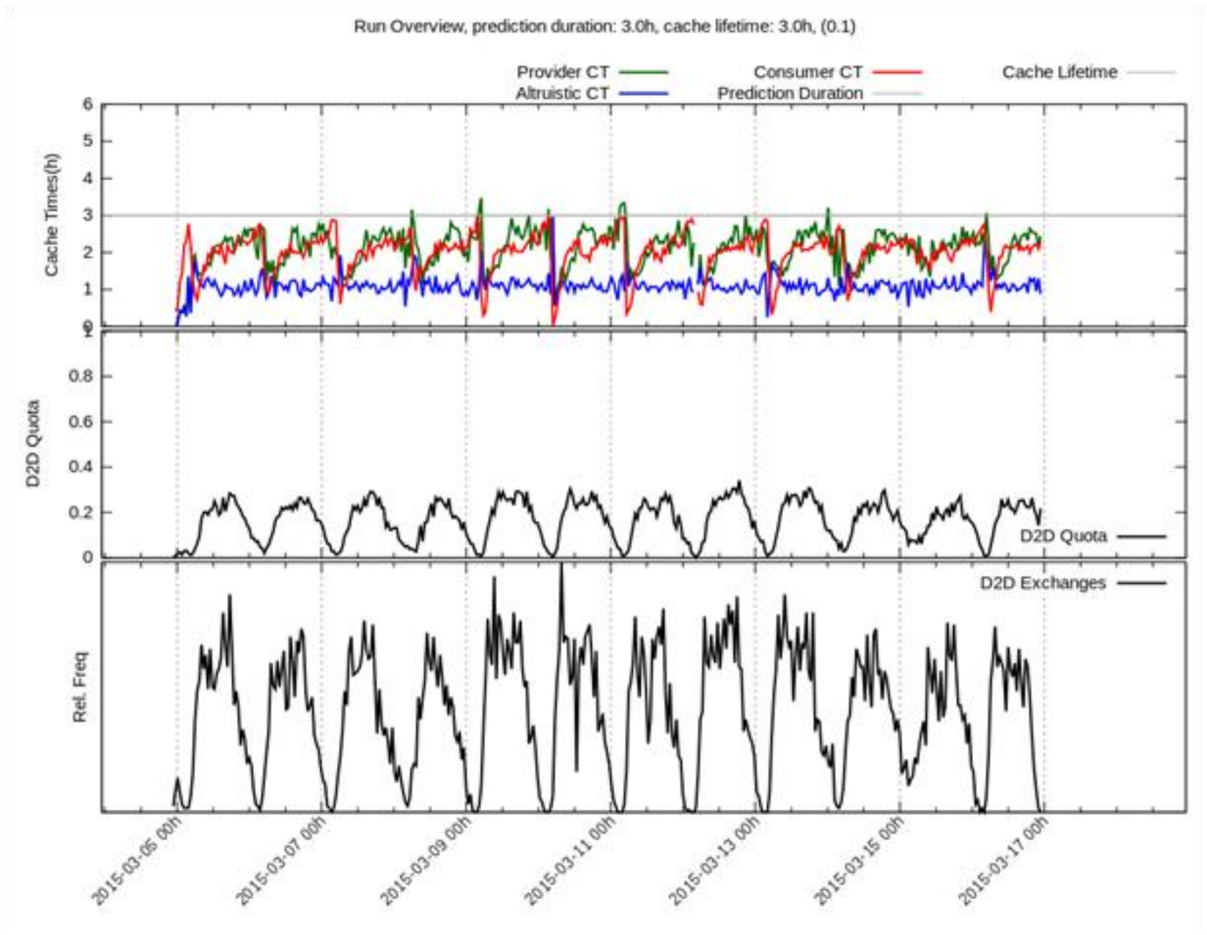


Figure 60: An overview over the whole simulation run as above. Prediction Duration = Cache Lifetime = 3h

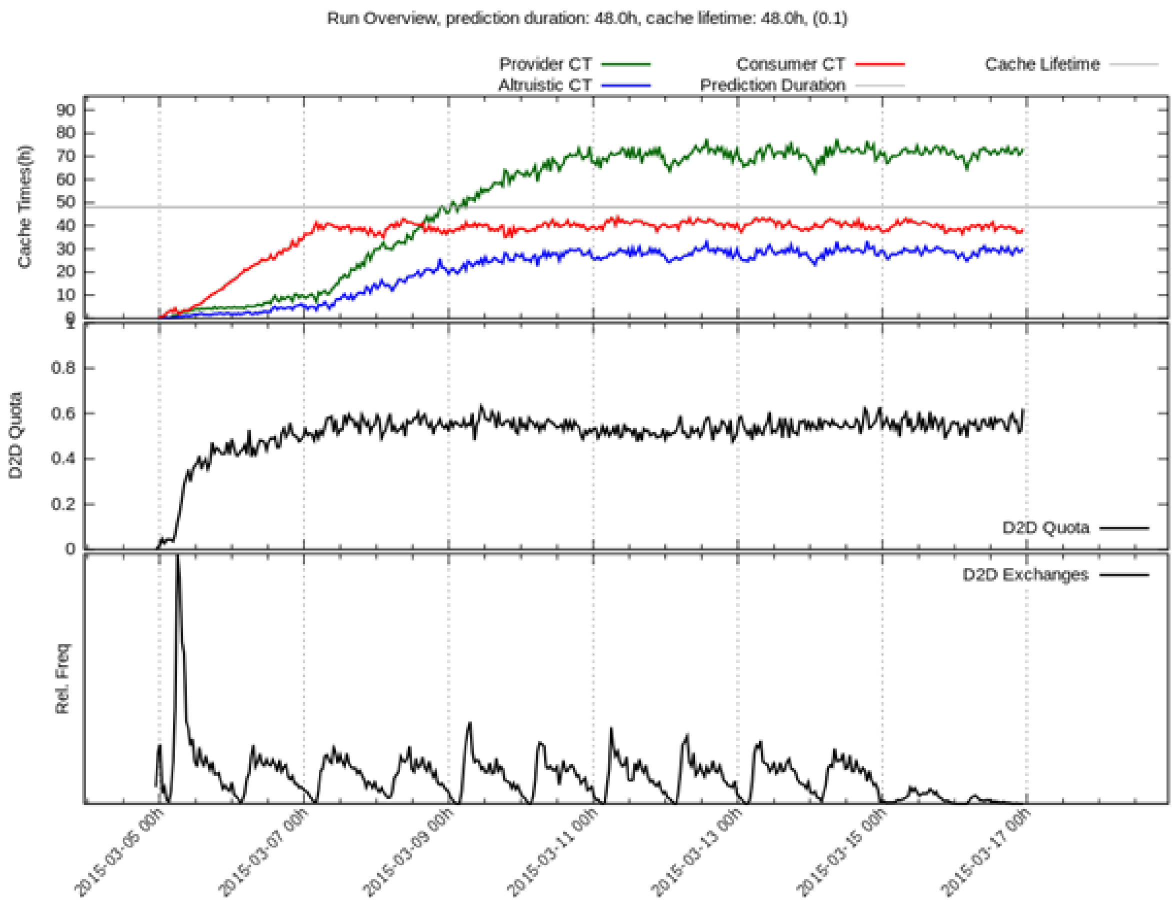


Figure 61: An overview over the whole simulation run, with the same values depicted as above. Prediction Duration = Cache Lifetime = 48h

In Figure 60 (3h) and Figure 61 (48h), we can see an overview of measured parameters over a whole simulation run. In the latter figure, we see a long-term behaviour worth to discuss: As the prediction duration is set to 48h here, on the one hand we can see that, especially on the first day in the morning after the content becomes popular, there is a peak of D2D content exchanges. Again, the long prediction time enables the devices to take the first chance to exchange content which is then consumed during the next two days. On the other hand, we can observe fewer exchanges during the last two days of the experiment. Here, many devices do not require subsequent exchanges, as the content was already fetched the days before.

A finding here is that a long prediction time can deteriorate the positive effect of “smooth” (i.e. approximately equally distributed) consumption models on spectrum usage, by shifting D2D exchanges at the start of content popularity periods. However, if the spectrum does not allow the required amount of D2D exchanges at this time and requests become denied, they may as well be conducted at a later point in time.

2.4.13.4 *Conclusion and Outlook*

Given our content consumption model, we could show that for popular content (requested by 1-10% of the devices) there is a large opportunity to reduce infrastructure utilization by making use of D2D content delivery mechanisms (up to 60%). We also find that a mechanism that *predicts* content consumption is crucial for efficient D2D delivery, and that it should at least predict content that may be likely consumed 12 hours in advance. In turn, *caching* content after consumption for others (altruistic caching) is only necessary, if content consumption can just be predicted one day in advance or less.

However, the findings about the D2D quota must be seen as an upper bound for the success of content delivery. In reality, the quota will likely be below this value because of three reasons: First, for the most consumption behaviour, there will not be a perfect prediction mechanism, as it was assumed here. Secondly, radio obstacles will often deteriorate the D2D communication range, leaving it lower than the range of mobile cells. Thirdly, spectrum interference and D2D congestion at crowded places was not considered here. This study was conducted to motivate the potential of D2D content delivery, and leaves the three aforementioned challenges for future research.

Regarding future work on assessing D2D potential, other content consumption models may be applied (Weibull distribution). The *Otraces* evaluation component is also capable of using real content requests, however, due to privacy issues, the experiment could not be conducted with this type of data. Furthermore, information about the cell sizes may be exploited in the model, if available.

3. CONCLUSIONS

Deliverable D6.5 concludes the activities of the eCOUSIN project and reports the outcome of all the tests performed to validate the planned objectives. The project has been able to respect the planned timeline for the software release as well as for the early test. This allowed us to solve the majority of the problems between the early and the final software release. In turns, we could verify that all the requirements identified during the early phases of the project have been satisfied by the project results. In particular, we have been able to verify that the set of tests performed covers every architecture functionality and obtained a successful outcome.

REFERENCES

[D2.1] EU eCOUSIN: Public Deliverable D2.1 – Initial report on uses cases and requirements. May 2013.

[D2.2] EU eCOUSIN: Public Deliverable D2.2 – Initial system architecture specification, July 2013.

[D2.4] EU eCOUSIN: Public Deliverable D2.4 – Final Report on Use Cases, Requirements, System Architecture Specification, Privacy and Regulations, July 2014

[D3.1] EU eCOUSIN: Public Deliverable D3.1 – Measurement, Modelling, and Prediction of Social-Content Interdependencies, October 2013

[D3.2] EU eCOUSIN: Public Deliverable D3.2 – Initial Release of Measurement and Prediction Software, April 2014

[D3.3] EU eCOUSIN: Public Deliverable D3.3 – Measurement, Modelling, and Prediction of Social-Content Interdependencies (Final Version), December 2014

[D4.1] EU eCOUSIN: Public Deliverable D4.1 – Preliminary Report on the Design of Technical Solutions on Content Placement and Delivery, October 2013.

[D4.2] EU eCOUSIN: Public Deliverable D4.2 – Final Report and Initial Software Release of the Design Extensions and Preliminary Implementation of the Technical Solutions on Content Placement and Delivery, April 2014

[D4.3] EU eCOUSIN: Public Deliverable D4.3 – Final Implementation and Software Release of the Technical Solutions on Content Placement and Delivery, December 2014

[D5.1] EU eCOUSIN: Public Deliverable D5.1 – Requirements for social-enhanced content centric and mobile network infrastructures, October 2013.

[D5.2] EU eCOUSIN: Public Deliverable D5.2 – Modules and Interfaces for social-enhanced content centric and mobile network infrastructures (V1), April 2014.

[D5.3] EU eCOUSIN: Public Deliverable D5.3 – Modules and Interfaces for social-enhanced content centric and mobile network infrastructures (V2), December 2014.

[D6.1] EU eCOUSIN: Public Deliverable D6.1 – Preliminary Plan for System Integration and Assessment (without commitment about the final implementation), January 2014

[D6.2] EU eCOUSIN: Public Deliverable D6.2 – Final Plan for System Integration and Assessment, June 2014

[D6.4] EU eCOUSIN: Public Deliverable D6.4 – Final Software Release, March 2015

[BALA09] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in ACM SIGCOMM Conference on Internet Measurement, 2009.

[BALA10] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting Mobile 3G using Wi-Fi," in ACM SIGMOBILE Mobile Computing and Communications Review, 2010.

[BRESL99] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker, Web caching and zipf-like distributions: evidence and implications, INFOCOM'99, vol. 1, March 1999, pp. 126 -134.

[BROX10] T. Broxton, Y. Interian, J. Vaver, and M. Wattenhofer, "Catching a viral video," in IEEE SIASP@ICDM 2010, 2010.

[BUI14] Nicola Bui, Foivos Michelinakis, Joerg Widmer. 2014. A Model for Throughput Prediction for Mobile Users. In Proceedings of European Wireless (EW'14).Barcelona, Spain, 14, May 2014, pagg.1-6.

[BUI14b] Nicola Bui, Joerg Widmer. 2014. Modelling throughput prediction errors as Gaussian random walks. In Proceedings of 1stKuVS Workshop on Anticipatory Networks. Stuttgart, Germany, 29-30, September 2014.

[BUI15] Nicola Bui, Stefan Valentin and Joerg Widmer. 2015. Anticipatory Quality-Resource Allocation for Multi-User Mobile Video Streaming. In proceedings of the 2ndWorkshop on Communication and Networking Techniques for Contemporary Video (CNTCV'15).Hong Kong, China, 27 April, 2015.

[BUI15b] Nicola Bui and Joerg Widmer. 2015. Mobile Network Resource Optimization under Imperfect Prediction. In proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, (WoWMoM'15). Boston, MA, USA, 14-17 June, 2015.

[CHAI07] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on opportunistic forwarding algorithms," Mobile Computing, IEEE Transactions on, Vol. 6, No. 6, pp. 606–620, 2007.

[CHE02] HaoChe, Ye Tung, and Zhijun Wang, Hierarchical web caching systems: modelling, design and experimental results, IEEE JSAC 20 (2002), no. 7, 1305-1314.

[CHEN09] X. Cheng and J. Liu, “Nettube: Exploring social networks for peer-to-peer short video sharing,” in INFOCOM 2009, IEEE, April 2009, pp. 1152–1160.

[CLAY10] M. Claypool and K. Claypool, “Latency can kill: precision and deadline in online games,” ACM SIGMM conference on Multimedia systems, pp. 215–222, 2010.

[IGRA15] cran.r-project.org/package=igraph

[DIMA11] S. Dimatteo, P. Hui, B. Han, and V. Li, “Cellular traffic offloading through wifi networks,” in Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, Oct 2011, pp. 192–201.

[DO14] N. M. Do, Y. Zhao, S. Wang, C. Hsu, and N. Venkatasubramanian,“Optimizing offline access to social network content on mobile devices,” in 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014, 2014, pp. 1950–1958.

[EAGL09] N. Eagle, A. S. Pentland, and D. Lazer, “Inferring friendship network structure by using mobile phone data,” Proceedings of the National Academy of Sciences, Vol. 106, No. 36, pp. 15 274–15 278, 2009.

[FIED10] M. Fiedler, T. Hossfeld, and P. Tran-Gia, “A generic quantitative relationship between quality of experience and quality of service,” IEEE Network, no. April, pp. 36–41, 2010.

[GERB10] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman, “Speed Testing without Speed Tests: Estimating Achievable Download Speed from Passive Measurements,” in ACM SIGCOMM Conference on Internet Measurement, 2010.

[GUIL13a] F. Guillemin, T. Houdoin, and S. Moteau, “Volatility of youtube content in orange networks and on sequences,” in Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013, 2013, pp. 2381–2385.

[GUIL13b] F. Guillemin, B. Kauffmann, S. Moteau, and A. Simonian, “Experimental analysis of caching efficiency for youtube traffic in an isp network,” in Teletraffic Congress (ITC), 2013 25th International, Sept 2013, pp.1–9.

[GUO13] W. Guo, S. Wang, T. O’Farrell, and S. Fletcher, “Energy Consumption of 4G Cellular Networks: A London Case Study,” in IEEE Vehicular Technology Conference, 2013.

[HAN10] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, “Cellular traffic offloading through opportunistic communications: A case study,” in Proceedings of the 5th ACM Workshop on Challenged Networks, ser. CHANTS ’10. New York, NY, USA: ACM, 2010, pp. 31–38. http://doi.acm.org/10.1145/1859934.1859943

[HUAN12] J. Huang, F. Qian, A. Gerber et al., “A Close Examination of Performance and Power Characteristics of 4G LTE Networks,” in ACM Conference on Mobile Systems, Applications, and Services, 2012.

[HUAN13] J. Huang, F. Quian, Y. Guoet al., “An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance,” in ACM SIGCOMM Conference, 2013.

[ICKI13] S. Ickin, K. Wac, and M. Fiedler, “QoE-Based Energy Reduction by Controlling the 3G Cellular Data Traffic on the Smartphone,” in ITC Specialist Seminar on Energy Efficient and Green Networking, 2013.

[KAUP15] Fabian Kaup, Florian Jomrich, David Hausheer: Demonstration of NetworkCoverage – A Mobile Network Performance Measurement App. In: International Conference on Networked Systems (NetSys 2015), March 2015.

[KHEM12] S. Khemmarat, R. Zhou, D. Krishnappa, L. Gao, and M. Zink, “Watching user generated videos with prefetching,” Signal Processing: Image Communication, Vol. 27, No. 4, pp. 343 – 359, 2012, modern Media Transport – Dynamic Adaptive Streaming over fHTTPg(DASH). http://www.sciencedirect.com/science/article/pii/S0923596511001342

[KHIR02] S. Khirman and P. Henriksen, “Relationship between Quality-of-Service and Quality-of-Experience for Public Internet Service,” Workshop on Passive and Active Measurement Conference, vol. Session 6, 2002.

[KOCH14] Christian Koch, Julius Rückert, Nicola Bui, Foivos Michelinakis, Guido Fioravantti, David Hausheer, Joerg Widmer. 2015. Demo: Mobile Social Prefetcher using Social and Network Information. In IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks 2014. Athens, Greece. 1-3December 2014.

[KOCH14b] Christian Koch, David Hausheer: Optimizing Mobile Prefetching by Leveraging Usage Patterns and Social Information. In: IEEE International Conference on Network Protocols (ICNP 2014), PhD Forum Paper, October 2014

[KOCH15] Christian Koch, Nicola Bui, Julius Rückert, Guido Fioravantti, Foivos Michelinakis, Stefan Wilk, Joerg Widmer, David Hausheer. 2015. Demo: Media Download Optimization through Prefetching and Resource Allocation in Mobile Networks. In ACM Multimedia System Conference 2015.Portland, Oregon, US. 18-20 March 2015.

[LANE12] M. Laner, P. Svoboda, P. Romirer-Maierhoferet al., “A Comparison Between One-way Delays in Operating HSPA and LTE Networks,” in Wireless Network Measurements and Experimentation, 2012.

[LI12] Z. Li, H. Shen, H. Wang, G. Liu, and J. Li, “Socialtube: P2p-assisted video sharing in online social networks,” in INFOCOM, 2012 Proceedings IEEE, March 2012, pp. 2886–2890.

[LI15] W. Li, R. K. P. Mok, D. Wu, and R. K. C. Chang, “On the Accuracy of Smartphone-based Mobile Network Measurement,” in INFOCOM, 2015.

[MAHA00] A. Mahanti, C. Williamson, and D. Eager, Traffic analysis of a web proxy caching hierarchy,IEEE Network (2000), 16-23.

[MICH15] Foivos Michelinakis, Nicola Bui, Guido Fioravantti, JoergWidmer, Fabian Kaup, David Hausheer. 2015. Lightweight Mobile Bandwidth Availability Measurement. In IFIP Networking conference 2015. Toulouse, France. 22-25 May, 2015.

[NICH08] A. J. Nicholson and B. D. Noble, “BreadCrumbs: Forecasting Mobile Connectivity,” in ACM International Conference on Mobile Computing and Networking, 2008.

[PAUL15] T. Paul, D. Puscher, S. Wilk, and T. Strufe, “Systematic, large-scale analysis on the feasibility of media prefetching in online social networks,” in CCNC, 2015.

[PROA09] J. G. Proakis and M. Salehi, Digital Communications. McGraw-Hill Higher Education, 2009.

[REZA11] F. Rezaei, M. Hempel, and H. Sharif, “LTE PHY Performance Analysis under 3GPP Standards Parameters,” in Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks, 2011.

[SEDH13] S. Sedhain, R. Kidd, S. Sanner, and P. Christen, “Social Affinity Filtering: Recommendation through Fine-grained Analysis of Categories and Subject Descriptors,” in ACM Conference on Online Social Networks (COSN), No. 2, 2013.

[SHAF13] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, “A first look at cellular network performance during crowded events,”in ACM SIGMETRICS Performance Evaluation Review, Vol. 41, No. 1.ACM, 2013, pp. 17–28.

[SONN13] S. Sonntag, L. Schulte, and J. Manner, “Mobile Network Measurements – It’s not all about Signal Strength,” in IEEE Wireless Communications and Networking Conference, 2013.

[STTW] An exhaustive study of Twitter users across the world, http://www.beevolve.com/twitter-statistics/

[TRUO15a] Patrick Truong, Bertrand Mathieu, Jean-Francois Peltier. Networking Impact of a Local-Aware Content-Based Delivery for Twitter-Like Applications. ICIN 2015. February 2015.

[TRUO15b] P. Truong, K. Satzke, B. Mathieu, E. Stephan. Named data networking for social network content delivery. Internet Draft draft-truong-icnrg-ndn-osn-00.txt. ICN Research Group. March 4, 2015.https://tools.ietf.org/id/draft-truong-icnrg-ndn-osn-00.txt

[VECI13] Z. Lu and G. de Veciana, “Optimizing stored video delivery for mobile networks: The value of knowing the future,” in Proceedings IEEE INFOCOM, 2013, pp. 2706–2714.

[VERG13] E. J. Vergara and S. Nadjm-Tehrani, “EnergyBox: A Trace-Driven Tool for Data Transmission Energy Consumption Studies,” in Energy Efficiency in Large Scale Distributed Systems, ser. Lecture Notes in Computer Science, J.-M. Pierson, G. Da Costa, and L. Dittmann, Eds. Springer Berlin Heidelberg, 2013, vol. 8046.

[WAC09] K. Wac, “Collaborative Sharing of Quality of Service-Information for Mobile Service Users,” PhD thesis, University of Geneva, Geneva, Switzerland, 2009.

[WANG11] Z. Wang, L. Sun, S. Yang, and W. Zhu, “Prefetching strategy in peer-assisted social video streaming,” in Proceedings of the 19th ACM International Conference on Multimedia, ser. MM’11. New York, NY, USA: ACM, 2011, pp. 1233–1236. http://doi.acm.org/10.1145/2072298.2071982

[WILK15] Stefan Wilk, Julius Rückert, Timo Thräm, Christian Koch, Wolfgang Effelsberg, David Hausheer: The Potential of Social-aware Multimedia Prefetching on Mobile Devices. In: IEEE International Conference on Networked Systems (NetSys 2015), March 2015.

[WYLI10] M. P. Wylie-Green and T. Svensson, “Throughput, Capacity, Handover and Latency Performance in a 3GPP LTE FDD Field Trial,” in IEEE Global Communications Conference, 2010.

[YAO08] J. Yao, S. S. Kanhere, and M. Hassan, “An Empirical Study of Bandwidth Predictability in Mobile Computing,” in ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization, 2008.

[ZHAO13] Y. Zhao, N. Do, S.-T. Wang, C.-H. Hsu, and N. Venkatasubramanian, “O 2 sm: Enabling efficient offline access to online social media and social networks,” in Middleware 2013, ser. Lecture Notes in Computer Science, D. Eyers and K. Schwan, Eds. Springer Berlin Heidelberg, 2013, Vol. 8275, pp. 445–465.

ACRONYMS

API	Application Program Interface
ASU	Arbitrary Signal Unit
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CO	Content Offloading
CP	Content Placement
D2D	Device-to-Device
dB	Decibel
DNS	Domain Name System
E2E	End-to-End
FIFO	First In First Out
GPS	Global Positioning System
GUI	Graphic User Interface
HR	Hit Ratio
ICN	Information-centric Networking
ICMP	Internet Control Message Protocol
IRTF	Internet Research Task Force
ISP	Internet Service Provider
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
LAC	Location Area Code
LRU	Least Recently Used
LTE	Long Term Evolution
NAT	Network Address Translation
NDN	Named Data Networking
OSN	Online Soocial Network
PoP	Point of Presence
PSC	Private Sharing Clouds

QoS	Quality of Service
RTT	Round-Trip Time
SDN	Software Defined Network
SINR	Signal over Interference and Noise Ratio
SQL	Structured Query Language
URL	Unique Resource Locator
WP	Work Package