Deliverable D5.3

Modules and Interfaces for Social-Enhanced Content Centric and Mobile Network Infrastructures (V2)

Public deliverable, Version 1.0, 19 December 2014

Authors

FT Bertrand Mathieu, Patrick Truong
AL-BELL
IMDEA Foivos Michelinakis, Nicola Bui, Joerg Widmer
TSP
ALUD Klaus Satzke, Ivica Rimac
TUD Fabian Kaup, Leonhard Nobach
TI Claudio Venezia, Fabio Mondin Luciano

UC3M

Reviewers Mondin Fabio Luciano (TI)

Abstract

After having defined the functional and technical requirements in D5.1 which have been derived from the eCOUSIN use cases as defined in D2.1 and D2.2, the present deliverable D5.3 captures the design of interfaces and modules and details of their implementation and their simulative evaluations in final version (V2), using the description of the requirements and the designs in first version as provided in D5.2 as guideline. The document provides details about the architectural decisions taken for the eCOUSIN system on both at network and application layers. The technical system architecture is detailed in particular for the eCOUSIN social-enhanced Content Centric and Mobile Network Infrastructures infrastructure levels.
EXECUTIVE SUMMARY

The deliverable D5.3 presents the final view on solutions proposed by the eCOUSIN project on the infrastructure layer, consisting of functionalities required for monitoring the network status, policy-based in-network routing and caching of content, as well as for configuring of network resources. While Deliverable 5.1 was focused on the description of requirements for the modules and interfaces as described in first version in the Deliverable D5.2, the work described in the present document describes further developments, new functionalities and extensions for the final version in different areas.

These solutions have been elaborated to improve the efficiency of content delivery beyond the current state of the art. Social-content interdependencies, which can be extracted from Online Social Networks (OSN), targeting both fixed content-centric as well as mobile infrastructures, have been used to drive this improvement.

This document describes the final implementation of the network infrastructure architecture.
# TABLE OF CONTENTS

**EXECUTIVE SUMMARY** ................................................................................................................................. 2

**TABLE OF CONTENTS**.......................................................................................................................................... 3

1. **INTRODUCTION** ........................................................................................................................................... 5

2. **RELATED WORK** ........................................................................................................................................... 5

3. **ECOUSIN INFRASTRUCTURE ARCHITECTURE OVERVIEW** ...................................................................... 6

4. **ECOUSIN SOLUTION COMPONENTS** ........................................................................................................ 28

4.1 **Solutions for Social-Enhanced Content Naming** .................................................................................. 28

4.1.1 **NDN-based OSN delivery** ................................................................................................................. 28

4.1.2 **Personal Sharing clouds (TI)** ............................................................................................................. 28

4.1.2.1 **Personal Sharing clouds (TI)** ........................................................................................................ 28

4.1.2.2 **Assessment/Update on the Personal Sharing clouds Naming Scheme** ........................................ 29

4.2 **Solutions for Content-Centric Data Plane** ............................................................................................ 46

4.2.1 **Adaptive Routing for Locality-Awareness in OSNs** ......................................................................... 46

4.2.2 **Dynamic Routing Configuration for Local Routing** ......................................................................... 48

4.2.3 **Mobile federated social networks** ..................................................................................................... 53

4.2.3.1 **Mobile federated social networks integration plan** ....................................................................... 54

4.2.3.2 **WiFi-Direct-capable Content Synchronizing Application for the Android OS** .......................... 55

4.2.4 **NDN-based nodes and interfaces** ...................................................................................................... 49

4.2.4.1 **OSN server** ..................................................................................................................................... 50

4.2.4.2 **NDN Forwarding Nodes** .............................................................................................................. 52

4.2.4.3 **Use Case elements implementation progress and updates** ....................................................... 53

4.3 **Solutions for Social-Enhanced Mobile Infrastructures** .......................................................................... 53

4.3.1 **Mobile network offloading in social places** .................................................................................... 55

4.3.1.1 **Use Case elements definition and tasks** .................................................................................... 30

4.3.1.2 **Use Case elements implementation progress and updates** ....................................................... 31

4.3.2 **Control Plane for Content-Centric Routing/Caching in NDN** .......................................................... 31

4.3.2.1 **Overall NDN modules and interfaces design** .............................................................................. 32

4.3.2.2 **ICN Controller architecture concept** .......................................................................................... 33

4.3.2.3 **ICN node modules and interfaces** ................................................................................................ 35

4.3.2.4 **ICNControl Application Programmable Interface (API)** ........................................................... 36

4.3.2.5 **Implementation and Evaluation** .................................................................................................. 38

4.3.3 **Solutions for Content-Centric Data Plane** .................................................................................... 46

4.3.3.1 **Adaptive Routing for Locality-Awareness in OSNs** ..................................................................... 46

4.3.3.2 **Dynamic Routing Configuration for Local Routing** .................................................................... 48

4.3.3.3 **Use Case elements definition and tasks** .................................................................................... 30

4.3.3.4 **Use Case elements implementation progress and updates** ....................................................... 31

4.3.3.5 **Implementation and Evaluation** .................................................................................................. 38

4.3.4 **Mobile network offloading in social places** .................................................................................... 55

4.3.4.1 **Use Case elements definition and tasks** .................................................................................... 30

4.3.4.2 **Use Case elements implementation progress and updates** ....................................................... 31

4.3.4.3 **Implementation and Evaluation** .................................................................................................. 38

4.3.5 **NDN-based nodes and interfaces** ...................................................................................................... 49

4.3.5.1 **OSN server** ..................................................................................................................................... 50

4.3.5.2 **NDN Forwarding Nodes** .............................................................................................................. 52

4.3.5.3 **End-User nodes** ............................................................................................................................ 53

4.3.6 **Solutions for Social-Enhanced Mobile Infrastructures** .................................................................... 53

4.3.6.1 **Mobile federated social networks integration plan** ....................................................................... 54

4.3.6.2 **WiFi-Direct-capable Content Synchronizing Application for the Android OS** .......................... 55

4.3.6.3 **Use Case elements implementation progress and updates** ....................................................... 53

4.3.6.4 **Implementation and Evaluation** .................................................................................................. 58
1. INTRODUCTION

The work presented in this deliverable describes this work packages’ final contribution to the laboratory prototypes and system tests of the of the technical solutions implementations elaborated by the project. The descriptions include:

- Final design and implementation of infrastructure components and interfaces created for information-centric and mobile infrastructures
- Methodology and software tools created and used for functional testing and evaluations
- Description of the created applications for D2D Content Discovery and Network monitoring/Bandwidth availability modelling

The eCOUSIN project proposes to analyse the social-content interdependencies extracted from Online Social Networks (OSN) to improve the efficiency of content delivery. This document aims at the required network infrastructure layer for such a solution. The overall goal of WP5 is

- to study the utilization of Information-centric networking (ICN)/content centric networking (CCN) as network layer for OSN applications,
- to design content aware naming and routing strategies that consider social properties,
- to define energy-efficiency content delivery strategies in mobile environments that exploits user mobility patterns and social properties
- to design time unconstrained content delivery strategies based on social information to reduce the operational cost of mobile operators and improve the users’ quality of experience.

This work is divided in two tasks representing activities on Social-enhanced Content-Centric Network Infrastructures and Social-enhanced Mobile Network Infrastructures. All the results achieved within the tasks are grouped within this deliverable for better clarity and understanding.

2. RELATED WORK

In parallel with the proliferation of OSNs the information-centric networking (ICN) approach has emerged as a new networking paradigm where the focus is on the content the users wish to obtain instead of identifying servers that could provide this content.

This ICN paradigm continues representing an excellent match to OSN applications requirements and can provide some natural benefits, such as lower response time due to pervasive caching and nearest-replica routing, intrinsic content integrity without relying on network-level indicators, simplified traffic engineering and improved support for user mobility. Some more recent updates on specific topics in related work and state of the art will be outlined in the following section.

A Controller-based routing scheme (CRoS) for Named Data Networks (NDN) has been proposed [TORR12]. CRoS defines a special network element called controller who is responsible for the named data location storage and routing. The controllers learn the topology in the bootstrap phase and compute routes to all the routers. In this phase, the router-controller routes are installed in routers while routes to named data are not. After the bootstrap phase, new named data is registered in the controllers that store the named data locations. Then a router can request the controllers for installation of a new route to an unknown prefix. Since all named data locations are now registered in the controllers, they can calculate routes to any valid named data.

CRoS runs on top of the NDN and uses only NDN packets. Therefore it preserves NDN features such as congestion control, network failure detection and path diversity. Nonetheless, CRoS uses special interest packets with semantically meaningful names to reduce control overhead. To avoid the
explosion of named data location storage in the controllers, Distributed Hash Tables (DHT) have been used to balance the storage between existing controllers. However, no details on the simulative or experimental evaluation are given which is left for future work.

iDNS [SEVI14] has been presented as an evolutionary path towards deploying ICN at Internet scale based on modifications to the DNS. At the core of iDNS is a new type of DNS record called Content Record (CR) that refers to a particular named data object (NDO) or name prefix. Clients desiring an NDO or name prefix must first resolve the corresponding CR through the DNS, which contains the address of one or more servers hosting the content along with protocol-specific metadata necessary to fetch the content.

In the event of a client receiving several address records it must assume that the records have been ranked by the DNS for locality, availability, or some other metric, but the paper does not discuss how this ranking can be performed by the DNS system. Also, only iDNS-aware nodes along the DNS response path can add address records or reorder CR records in the set.

Finally, adoption of the Virtual Aggregation (ViAggre) approach [BALL09] for NDN has been proposed for shrinking of the FIB size by splitting the Forwarding Information Base (FIB) entries between several routers [CHOI14]. However, the proposal calls for tunnelling of interest packets between NDN routers, and consequently the maintenance of a corresponding Publisher Information Base (PIB) to avoid interest packet loops.

3. ECOUSIN INFRASTRUCTURE ARCHITECTURE OVERVIEW

Detailed technical requirements have been derived for the infrastructure layer and documented in D5.1. These requirements have then been used to drive the design decisions and implementations of modules and interfaces at the eCOUSIN infrastructure layer described in D5.2. The current deliverable D5.3 provides more details for these modules and interfaces and provides first evaluation results.

In the following the final eCOUSIN infrastructure building blocks which initially have been described in deliverable D5.2 will be further detailed, and initial evaluation results will be presented.

3.1 Network Monitoring

3.1.1 International Measurement Campaign

In order to develop, test and validate the observations and techniques presented in the following sections 3.1.2 and 3.1.3 over many different scenarios and configurations, the project partners, jointly generated and used a vast dataset of mobile parameters. This dataset was further expanded by measurements made by individual partners based on the specific needs of their objectives.

The measurement campaign covered two cities in two different countries, Madrid (Spain) and Darmstadt (Germany), during 24 hours a day for 7 days. During this time, 5 persons per city moved around as they normally do, carrying one measuring device each and performing their usual tasks involving mobile networking on the measuring devices. Various parameters regarding the context of the phone were measured either constantly (channel quality information) or periodically (active bandwidth and round trip time (RTT) measurements). The specific parameters used by each partner, as well as the potential use of further measurements are analysed in the following sections.

The phones used in the campaign were the following: five Nexus 5 phones located in Germany, and four Sony Xperia Miro and one Samsung Galaxy S3 located in Spain. Also, while the Nexus 5 phones are LTE capable, the other phones only support radio technologies up to HSPA (see Table 1).
Table 1: Smartphone types used in the measurement campaign

<table>
<thead>
<tr>
<th>Phone type</th>
<th>Nexus 5</th>
<th>Samsung Galaxy S3</th>
<th>Sony Xperia Miro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of devices</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Location</td>
<td>Germany</td>
<td>Spain</td>
<td>Spain</td>
</tr>
<tr>
<td>Most recent radio technology supported</td>
<td>LTE</td>
<td>HSPA</td>
<td>HSPA</td>
</tr>
</tbody>
</table>

3.1.2 Lightweight Mobile Bandwidth Availability Measurement Module

In this section, a novel lightweight measurement technique that can be used as a basis for advanced resource optimization algorithms to be run on mobile phones will be presented. Our main idea leverages on packet dispersion techniques to estimate both per user capacity and asymptotic dispersion rate. This allows passive measurements using only existing mobile traffic, such as the periodic updates sent by Smartphone apps. Our technique is able to efficiently filter outliers introduced by mobile network schedulers. In order to verify the feasibility of our measurement technique, we run a week-long measurement campaign spanning two cities in two countries, different radio technologies, and covering all times of the day. The campaign demonstrates that our technique is effective using just a few tens of consecutive packets sniffed periodically. This makes the measurement algorithm a good candidate for inclusion in mobile OS libraries to allow for advanced resource optimization and application-level traffic scheduling, based on current and predicted future user capacity.

![Diagram of packet dispersion](image1)

Figure 1: IP packet dispersion from when they are sent back-to-back from the server (1), then forwarded by the eNodeB (2) and finally received by the UE (3).
Figure 2: Pairs of a) scatterplots of $c_{W}(i)$ (left side) and b) histograms of $\gamma_{W}(c)$ (right side) computed for $t_T = \{1, 5, 10, 20, 30\} \text{ ms}$ from top to bottom. When the dispersion time is computed on windows larger than the TTI, $t_T > t_s$, the dispersion time distribution gets more stable.
3.1.2.1 Mobile Bandwidth Estimation

In a mobile network, $C_U$ is the capacity of the last hop of the downlink path. It is dependent on the cell congestion and the channel quality. With $R$ we denote the end-to-end TCP throughput achieved by mobile applications, which depends on the link with the smallest capacity in the downlink path.

Figure 1 shows the packet dispersion due to the transmission over links with different capacities. Initially, (1) the server sends a burst of IP packets (A-H in the example) back to back. Subsequently, (2) the base station (eNodeB) receives the packets, which have suffered variable delays due to the different link capacities and cross traffic encountered along the path. When the scheduler allocates a transport block (horizontal arrows) to the receiver or user equipment (UE) (3) as many packets as possible are encapsulated within it. Therefore, all these packets arrive within the same transmission time interval (TTI) at the UE.

Considering a single packet pair crossing the two-link path in Figure 1, the first link (backhaul) and the second link (cellular) are characterized by bandwidth $C_B$ and $C_U$ respectively. If $C_B > C_U$, the two packets arrive at the second link with a delay which is inversely proportional to $C_B$ and shorter than the average time needed for the second link to serve the first packet. Also, depending on the scheduling strategy, the two may or may not be served within the same subframe by the eNodeB. Conversely, if $C_B < C_U$, the two packets arrive to the second link separated by a delay which is longer than the average serving time of the eNodeB. We thus have three cases: i) bursty arrival [HUAN13][XU14] (e.g.: packets E-F), if $C_B > C_U$ and packets are in the same subframe, ii) last hop capacity if $C_B > C_U$ and packets are in different subframes (e.g.: packet train A-C), or iii) lowest hop capacity if $C_B < C_U$.

In order to estimate $C_U$, we have to filter both i) and iii) cases. To do so, we consider a train of $N$ packets sent back-to-back from a server and received at the UE so that the $i$-th packet is received at time $t_i$, with $i = \{1, \ldots, N\}$. Also, we define the dispersion time $d_W(i) = t_{i+W} - t_i$, and the per user dispersion rate $c_W(i) = \frac{\sum_{i=1}^{W-1} L_j}{d_W(i)}$, where $W$ is the number of intervals between packets over which the computations are made and $L_i$ is the length of $i$-th packet. In Figure 1, $c_{W=2}(A)$ is computed by dividing the sum of sizes of packets A and B by the dispersion time $d_{W=2}(A) = t_C - t_A$.

Subsequently, we define $\delta_W(d)$ and $\gamma_W(c)$ as the statistical distributions of the packet dispersion time and the dispersion rate, respectively. Note that the three arrival cases above contribute to those distributions in different ways: arrivals of type i) cause a tiny $d_W(i)$ and, thus, influencing the right part $\gamma_W(c)$, while the left part of it is influenced mostly from type iii) events, which show larger $d_W(i)$. To better visualize what is discussed next, Figure 2 shows a set of scatterplots and histograms related to $c_W(i)$ and $\gamma_W(c)$ computed on a single download performed using the Speedtest application [SPEE14] over a HSPA connection.

First, we may want to try to limit the impact of type i) arrivals by setting $W$ appropriately: the idea is to include in each measurement packets belonging to different subframes in order to make sure that the highest throughput $c_W(i)$, we can measure is only related to the cell capacity and not to bursty packet arrivals, as it would have happened had we chosen $W = 1$ in the example of Figure 1.
to achieve that, it is sufficient to study groups of $\hat{W}(i)$ intervals so that the minimum dispersion time is longer than the maximum TTI, $t_s$:

$$\hat{W}(i) = \min_{w} \{ d_w(i) > t_s \}; (1)$$

In fact, this guarantees that at least two packets within the $\hat{W}(i)$ window are scheduled in two different subframes since $t_{i+\hat{W}(i)} - t_i = d_{\hat{W}(i)}(i) > t_s$. In other words, we are averaging the burstiness over two subframes. The TTI $t_s$, is technology dependent and can be as short as 1ms for LTE or at least 10ms for UMTS.

Now that type i) events are filtered, the minimum dispersion time $\min d_{\hat{W}(i)}$ cannot be smaller than the minimum time needed for the packet train to cross the last link, which corresponds to the maximum per user cell capacity. Thus, $C_U$ can be found as the maximum of the distribution $\gamma_{\hat{W}}(c)$:

$$C_U = \max_c \gamma_{\hat{W}}(c), (2)$$

note that, with (1) we are filtering the effect of type i) arrivals ($\min$) and with (2) the delays introduced by type iii) arrivals ($\max$).

Ideally, we would like to sample $c_{\hat{W}}$ until the $\gamma_{\hat{W}}(c)$ is stable, but $C_U$ is varying because of both user movements and fast fading, hence we can only obtain an estimate $\tilde{C}^{(K)}$ of it from a set of $K$ consecutive dispersion time samples. Although estimating the distribution from a limited number of samples reduces the accuracy of our measurement, we can at least guarantee that we are not overestimating $C_U$:

$$\tilde{C}^{(K)} = \max_c \tilde{\gamma}^{(K)}(c) \leq \max_c \gamma_{\hat{W}}(c) = C_U, (3)$$

where $\tilde{\gamma}^{(K)}(c)$ is the distribution of $\tilde{c}^{(K)}$ obtained using $K$ dispersion samples. This follows from the probability of the distribution of a sampled random process to contain the maximum of the theoretical distribution of the process, which is increasing with the number of collected samples:

$$\lim_{K \to \infty} \tilde{C}^{(K)} = C_U, (4)$$

Following a similar approach the asymptotic dispersion rate, $R = E[c_{\hat{W}}][DOVR14]$ and its sampled version $\tilde{R}^{(K)}$, tends to the actual value when the number samples tends to infinity:

$$\lim_{K \to \infty} \tilde{R}^{(K)} = R, (5)$$

### 3.1.2.2 Bandwidth Measurement

This section describes the feasibility of lightweight active and passive measurements of per user capacity $C_U$ and asymptotic dispersion rate $R$ based on packet train dispersion samples. We compute the dispersion time by using an adaptive window $\hat{W}(i)$, which represents the number of inter-packet intervals. Its size is computed by summing the time duration of the inter-packet intervals until the following condition is true:

$$\hat{W}(t_r) = \arg \min_{w} \{ t_{i+\hat{W}} - t_i > t_r \}; (6)$$
where \( t_T \in [1, \ldots, 50] \text{ms} \). This allows to satisfy Eq. (2) a posteriori if the TTI duration is not known.

![Graph](image)

**Figure 3: Ratio \( \Delta(t_T) \), varying \( t_T \in [2, \ldots, 50] \text{ms} \). The measurements get stable from \( t_T > t_S = 10 \text{ ms} \).**

We exemplify the dispersion time in Figure 2 through two series of plots obtained by time-stamping the arrival time of the packets of a 6-MB HSPA download.

During the slow start phase of a TCP connection an increasing number of packets are sent back and forth from the server, and after a few round trip times (RTT) the congestion window is large enough to allow the transmission of packet trains long enough to measure capacity as high as 100 Mbps. In fact, \( C_U \) should be proportional to the maximum number of packets that got scheduled in a single subframe and, if (1) is satisfied and \( t_T > t_S \), the impact of outliers due to bursty arrivals is removed. In fact, with reference to Figure 2, it can be seen that the maximum of the capacity distribution \( \max_c \gamma_{W(t_T)}(c) \) is approaching a stable value of about 10 Mbps when \( t_T \geq 15 \text{ ms} \).

Moreover, Figure 3 shows the stability of the maximum of the capacity distribution by plotting the ratio \( \Delta(t_T) \), computed between the maximum of the distributions obtained with windows of \( t_T \) and \( t_T-1 \):
\[ \Delta(t_T) = \frac{|C_{w_{t_T}} - C_{\hat{w}_{t_T-1}}|}{C_{\hat{w}_{t_T-1}}} \] (7)

Ideally, the ratio \( \Delta(t_T) \) should stabilize to 0 as soon the scheduling outliers are filtered \( (t_T > t_s) \) and further increasing \( t_T \) should only make the distribution smoother. However, in actual experiments increasing \( t_T \) makes it more difficult to obtain a sample of the maximum capacity which is consistent over different subframes. In this preliminary example, we can see that \( \Delta(t_T) \) becomes stable for \( t_T > 20 \text{ ms} \), which is in line with the HSPA TTI of 2–10 ms.

**Figure 4:** Normalized root mean square error \( \varepsilon_c \) of the capacity estimate computed over a fraction \( f = K / N \) of continuous samples for varying bin sizes \( \{0.2 \text{ s}, 0.5 \text{ s}, 1 \text{ s}, 2 \text{ s}\} \)

Next, we divide the time duration of a download into fixed sized bins and we apply the above method taking into account only a percentage \( f = K / N \) of consecutive capacity samples in each bin. Figure 4 shows the normalized root mean square error, \( \varepsilon_c \) of the estimate by varying \( f \):
where $N_b$ is the number of bins in a flow. The computations have been repeated for different bin sizes varying in {0.2,0.5,1,2} seconds (dotted, dash-dotted, dashed and solid lines, respectively). It can be seen that the error decreases below 20% when more than 20% of the samples are used.

To complete this preliminary evaluation of our measurement technique, Figure 5 shows the variation of per user capacity $C_U(t)$ measured every 500 ms and its estimates $\tilde{C}^{(K)}(t)$ computed with $f = \{10,20,50,100\}$%.

Although with 10% of samples the estimates are quite different from the actual capacity values, we will be showing next that it is possible to exploit these coarse estimates to obtain a sufficiently accurate asymptotic throughput estimate.
3.1.2.3 Measurement Campaign

The majority of the data used to generate and test the techniques presented in this section are drawn from the measurement campaign presented in section 3.1.1.

In order to be able to compare results of both passive and active measurements, we performed in this campaign automated periodic file downloads (active tests).

All the devices were running a simple Android application, which was periodically sampling the available capacity by starting two download types: short downloads of 500 KB to study the TCP slow start phases and long downloads of 2 MB to measure TCP steady state throughput. The two types were organized in a sequence with a long download, preceded by two small downloads and later succeeded by another two. We use tcpdump on the measurement devices to monitor the arrival time and size of all incoming packets. The download sequence was repeated every 50 minutes, so that we downloaded a total of 800 MB and 8 GB per device during the campaign, in Spain and Germany respectively. Additionally, we log other related phone parameters: GPS, ID of the connected cell, Channel Quality Indicators (ASU, dBm) and network technology (2G, 3G, LTE).

3.1.2.4 Results and Discussion

We verified our measurement technique by analyzing more than 3000 unique TCP flows extracted from the communication of each of the phones participating in the campaign. For each of them we compute the asymptotic dispersion rate \( \hat{R} \), using (5) accounting for the whole information available, and the per user capacity \( \hat{C}^{(K)} \) for different percentages of used information \( f \in [1\% - 100\%] \) and different bin sizes (from 0.1s to 2 s).
Figure 6: Scatterplot of the average estimate of per user capacity \( \mathbb{E}[\tilde{C}^{(5\%)}] \) computed over 5% of the available information against the estimated end-to-end throughput measured using all available information. The dashed line shows the linear regression between the two quantities.

The first and most important result, Figure 6, shows a scatterplot where each point is obtained from a different flow for which the average per user capacity is used as the abscissa and the estimated end-to-end throughput as the ordinate. We also plot the linear regression between the two quantities as a dashed line in order to show that one is a good predictor of the other. We compute the per user capacity in each bin of the flow by using 5% of the samples of the bin and we average over the bins, while the throughput is obtained using all the available information.

The figure is plotted in double logarithmic scale in order to emphasize that the relationship between \( \tilde{R} \) and \( C_U \) can be observed over all the measured connection rates. Although outliers are visible, we can obtain quite an accurate estimate of the end-to-end throughput by exploiting as few as 5% of the packets sent during a TCP connection. This allows for quite an effective passive monitoring technique as, even by monitoring light data exchanges (e.g., periodic updates sent or requested by applications), it is possible to obtain frequent and accurate mobile bandwidth availability measure necessary for user throughput prediction and resource allocation.

As a side note, our technique is also able to estimate fast per user capacity variations. However, it obtains a lower accuracy since a larger fraction of samples are needed to estimate the maximum of the distribution \( Y_w(c) \). Nonetheless, it is often sufficient to use 20% of the samples collected in a bin to achieve a reasonable estimate of \( C_U \). In addition, \( T_r \) must be taken slightly longer than the TTI (in...
some cases even twice as long) to avoid the measurement being impacted by many bursty arrivals. In line with Eq.(1) of Section 0, $\Delta(t_R)$ approaches zero for $t_R > 15$ ms for most of the recorded flows since TTI is about 10ms long for most of technologies used (HSPA, HSPA+ and LTE).

Figure 7: Contour graph of $\varepsilon_C$ varying $t_R$ and $f$ for a bin size of 200ms.

Figure 7, shows the normalized RMSE for different combinations of $t_R$ and $f$. The bin size is set to 200ms to give an example of this technique’s results when it collects very frequent measurements. As expected $\varepsilon_C$ decreases when $t_R$ and $f$ increase. For values of $t_R \geq 15$ ms and $f \geq 20\%$, the error is small enough for the model to give trustworthy results ($\varepsilon_C \leq 15\%$).

Table 2: Average asymptotic dispersion rate $R$ and average optimal $t_R$ per technology.

<table>
<thead>
<tr>
<th>Technology</th>
<th>UMTS</th>
<th>HSPA</th>
<th>HSPA+</th>
<th>LTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR (Mbps)</td>
<td>10.83</td>
<td>1.4</td>
<td>10.74</td>
<td>24.3</td>
</tr>
<tr>
<td>Optimal $t_R$ (ms)</td>
<td>19</td>
<td>23</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Finally, Table 2 shows some of the overall evaluation of the traces obtained with $f=25\%$ averaging over the bin size and using the optimal $t_R$ ($\min t_R \mid \Delta(t_R) \to 0$). Optimal $t_R$ and average asymptotic dispersion rate are computed mimicking the Speedtest method. While some of the flows are transmitted using EDGE, the results are not included since there are too few such flows for statistical significance.
3.1.3 Measurement Study of Cellular Service Quality

Contrary to the passive measurements described in the previous section, the location based service quality was targeted in the part of the study run by TU Darmstadt by running active measurements wherever the user was located. For this, periodic RTT and throughput measurements were executed. For each, the time and location of the measurement was recorded with the goal of creating a QoS map, including the RTT and throughput measured at the individual locations. Based on this map, a traffic scheduling algorithm can decide where to execute data transfers, and such increase the service quality by simultaneously reducing the power consumption of the mobile device. A detailed description of approach and findings can be found in [KMB+14].

In particular, this work addresses the following questions related to the estimation of the cellular network quality:

1) What information on the network state can be derived from “cheap” information like the cell ID?
2) Can congestion of the cellular network be recognized based on inexpensive local measurements?
3) Can the cellular network throughput be predicted based on passive measurements only?

A number of publications deal with the measurement of cellular network performance [HQG+13, SSM13, YKH08], its prediction [YKH08], [BMW14], energy consumption [BBV09], [VN13], or mechanisms improving the mobile QoS, while keeping the quality of experience (QoE) high [IWF13]. A large number of studies focus on 3G networks [YKH08], [VN13], [BMV10], while others [HQG+13], [RHS11, GWF+13, HQG+12] evaluate the low-level network-based performance or model the energy efficiency. The work conducted during this study differs, as first, the evaluation of the RTT is executed on a per-cell basis, uses the end-user’s devices to estimate the network quality, and derives a model to estimate the network throughput based on the signal strength, which is available without requiring active measurements. A more detailed comparison with the related work is available in [KMB+14].

3.1.3.1 Measurement Methodology

The measurements in Germany were taken on the Nexus 5. First, the signal strength as returned by the Android framework, together with the network provider, operator, network type, cell ID and location area code (LAC) are recorded on the device. Each of these network samples is augmented with the current time and location to later allow deriving temporal and spatial models of the network behaviour. These data points are further called coverage points. Additionally, periodic RTT measurements were run on all devices to determine the response time of the cellular network. These are appended to the last recorded coverage point. The combination of both is referred to as ping data point. The downlink bandwidth available to the mobile device is determined by periodically running throughput measurements. Therefore, a patched version of iPerf is installed on the measurement server and the mobile device, allowing the downlink throughput measurements. This is required, as conventionally, iPerf measures only the upload from the device executing the test. These tests are run once every 25 minutes, ensuring that the data cap of the used data contracts is not exceeded. These throughput measurements are also appended to coverage or ping data points, and are referred to as throughput data points in the following sections.

3.1.3.2 Overview of the Acquired Data Set

The data gathered during the measurement study consists of 446,000 coverage points, 45,000 ping data points and 1,400 throughput data points. These were measured mainly in 4G networks. The dataset contains data from 1161 cells, from which the majority was discovered in 3G networks.
detailed number of samples of each group is given in Table 3. For the prediction of the RTT and throughput, an extended data set is used. The data was measured beginning from December 2013 until the end of the measurement study. Hence, the data from the measurement study is a subset of the overall available data. The details on the full dataset are given in Table 4.

### Table 3: Overview of the measurements collected during the measurement study

<table>
<thead>
<tr>
<th>Type</th>
<th>4G</th>
<th>3G</th>
<th>2G</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Strength</td>
<td>261k</td>
<td>158k</td>
<td>6k</td>
<td>446k</td>
</tr>
<tr>
<td>Cells</td>
<td>264</td>
<td>767</td>
<td>129</td>
<td>1161</td>
</tr>
<tr>
<td>RTT</td>
<td>14k</td>
<td>11k</td>
<td>273</td>
<td>45k</td>
</tr>
<tr>
<td>Throughput</td>
<td>583</td>
<td>420</td>
<td>1</td>
<td>1.4k</td>
</tr>
</tbody>
</table>

### Table 4: Overview of all collected measurements

<table>
<thead>
<tr>
<th>Type</th>
<th>4G</th>
<th>3G</th>
<th>2G</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Strength</td>
<td>1.55M</td>
<td>804k</td>
<td>78k</td>
<td>236M</td>
</tr>
<tr>
<td>Cells</td>
<td>952</td>
<td>63k</td>
<td>1.5k</td>
<td>8.8k</td>
</tr>
<tr>
<td>RTT</td>
<td>89k</td>
<td>44k</td>
<td>4.9k</td>
<td>139k</td>
</tr>
<tr>
<td>Throughput</td>
<td>3.5k</td>
<td>1.2k</td>
<td>36</td>
<td>4.8k</td>
</tr>
</tbody>
</table>

Evaluating the data confirmed the results of other publications. As expected, the throughput in LTE is higher and the RTT is lower compared to 3G. This is visualized in Figure 8. Interesting is the observation that the RTT for LTE builds clusters around 20, 30, and 40ms. Most of the throughput measurements are in a range of 20 Mbps to 60 Mbps, although there are measurements with rates of up to 100 Mbps, or as low as 500 kbps.

The dependencies between signal strength in arbitrary strength units (ASU) and throughput for LTE are visualized in Figure 9. The achievable data rate increases with increasing signal strength, but flattens out after average rates of 30 Mbps have been achieved.
Based on the collected data, it is also possible to determine the cell coverage for a given location. The measurements resembling the extent of a cell are determined by finding all samples with the same provider, 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 488 cells is 2.4 km², while the median is only 0.27 km², indicating a small number of large, and a large number of small cells.

The cell density for the city of Darmstadt is evaluated by creating a grid of sample points (see Figure 11) covering the inner part of the city. The CDF of the number of observed cells at each location is...
given in Figure 10. The cell density was sampled at the centres of each circle in Figure 11. 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 10: Number of cells covering one evaluation point](image1.png)

![Figure 11: Heatmap of the number of cells of a single network provider covering each location](image2.png)
3.1.3.3 Data Correlation

The relation between the RTT and the cell utilization is analyzed based on the samples collected during the two week measurement study. Furthermore, the collected data is restricted to a single provider. By comparing the RTT distribution in individual cells with the overall distribution the congestion of the network is estimated.

While Figure 12 shows the probability density function (PDF) of the 4G RTT values, the Figure 13 shows the count of the RTT values in a single cell obtained over the course of the measurement study. The bins on the time axis are one hour wide and begin at midnight on the day the measurement study started. The RTT values are given on the vertical axis in 1ms wide bins. The density of the colour indicates the number of samples observed within the defined time interval with the value indicated on the vertical axis. The plot shows a high variability of the RTT values of a single cell. However, clustering around certain peak values is apparent at any time. For example, between 48 and 72 hours, the RTT values first cluster around 30 ms, and later at around 20ms. The high variability becomes even more apparent when comparing the RTT measurements from multiple cells.

Figure 12: Distribution of the 4G RTTs in ms

Figure 13: RTT values within a single cell at the given hour

Figure 14 shows the count of the residual error of the RTT measurements at the respective hour in all cells observed during the study. The residual error, which is given on the vertical axis in 1 ms wide
bins, is defined as the time difference after subtracting the mean of each time bin over the RTT measurements of all cells at the respective time. The plot shows the high deviation from the mean of the RTT over all cells compared to a single cell as depicted in 6. This is in particular visible in the high number of observations with an error of ± 10 ms at day 3.

Figure 14: Deviation of the RTT measurements around the mean RTT of all cells at the given hour

However, Figure 15 shows the aggregated errors relative to the mean of the individual cells. This results in a much lower error, where most errors are in the range of 3ms from the mean RTT of the respective cell. This relation is further elaborated in Figure 16, where the cumulative distribution function (CDF) of both distributions is shown. The solid line represents the average error observed when comparing the measured RTTs with the means of each cell. The dashed line is the error distribution for the measured RTTs compared to the average RTT of all cells. Clearly, the average error is lower when calculating the error for the individual cells first. For example, the error of the RTT for 90% of the time intervals is lower than 15ms when knowing the mean RTT of the given network, while it is smaller than 12ms when knowing the mean RTT of the cell. Hence, the residual error can be reduced by up to 3ms. This is caused by the different mean RTT values observed in the individual cells.

Figure 15: Distribution of the difference between the RTT samples and the mean over all cells and the difference between RTT and the mean of the individual cells
This observation is also confirmed by running a Kolmogorov-Smirnov test. Concluding, the RTT observed within a cell at a given time is an indicator for the cell utilization. Considering that this result is based on the comparison of measurements conducted at the same time, implications on the operator’s network can be drawn. If the change in RTT of any two individual cells connected to the same backbone is different, the backbone is likely not congested. Different delays in the backhaul link of the individual cell can be ruled out, as the RTT deviation is calculated relative to the mean RTT of the respective cell. If the backbone was congested, the RTT over all cells would need to change uniformly. This is an effect, which has not been observed. Hence, it can be concluded that the delay in the individual cell is caused by the scheduler of the respective base station. In the case the backbone and the cell are congested, both effects are expected to superimpose, increasing the RTT on all observations. A further measurement study is to be conducted, focusing on this effect.

3.1.3.4 Prediction

This subsection presents first the results for machine learning employed for prediction the user RTT value when connected to different generation networks, and then the results for prediction of the user throughput value for the 4G network. The predictions employ a machine learning-based analysis of the data acquired in the study.

The RTT is predicted based on 31449 measurement samples (EDGE (774), UMTS (4180), HSDPA (81), HSPA+ (7323) and LTE (19091)). The features selected for machine learning are day of the week, hour, cell-ID, LAC, network type and received signal strength indicator (RSSI) value. The “InfoGain” algorithm, ranking the features’ predictive power by analyzing the influence of each on the entropy of the target class variable (RTT), indicates that cell-ID, LAC and network type are amongst the most predictive values for RTT. There were 1227 unique cell-IDs and 59 LAC values in the data set, RSSI values ranged from -51dB to -125dB, and the RTT values ranged from 15 to 13927ms, with 95.37% of the RTT values below 150ms. There were no missing data in this set. A prediction task has been defined as well as a classification task. First, a prediction of the RTT value has been analyzed assuming 21 classes, (i.e., a new class defined every 10ms from 10ms to 200ms and one class above). However, the predictions’ accuracy was very low, as there are too few discriminating variables for the 21 classes. Instead, the classification task has been changed into a binary one, i.e., towards a prediction of the RTT value being below/above some predefined threshold. There have been multiple selections of RTT thresholds potentially matching the user’s required RTT: 20, 30, 40, 60 and 80ms. The machine learning method selected for this task is a classification tree (J48 in WEKA suite), as it has been proven to perform in similar prediction tasks as suggested by Wac [IFW13]. The machine
learning-based analysis has been executed such that a stratified 90% of the data have been used for a training of a model, while the remaining 10% of the data for the testing.

Table 5: Predictions for given RTT thresholds

<table>
<thead>
<tr>
<th>RTT Threshold [ms]</th>
<th>Accuracy of Educated Guess [%]</th>
<th>Decision Tree accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>80.53</td>
<td>93.08</td>
</tr>
<tr>
<td>30</td>
<td>80.50</td>
<td>93.01</td>
</tr>
<tr>
<td>40</td>
<td>61.15</td>
<td>82.03</td>
</tr>
<tr>
<td>60</td>
<td>84.46</td>
<td>87.54</td>
</tr>
<tr>
<td>80</td>
<td>94.10</td>
<td>94.24</td>
</tr>
</tbody>
</table>

Table 5 presents the results of the predictions for given binary RTT thresholds. The first column represents the assumed threshold, while the second shows the “educated guess” accuracy value, i.e., when the prediction is made based only on the historical values of the RTT observed in the training data set. The third column presents the results when the decision tree is employed and evaluated on the testing data set.

For the prediction of the throughput, the selected subset of 637 measurement instances has been leveraged for this purpose; representing the 4G measurement sub-set. The features have been selected from the set of available features (e.g., user ID, LAC, min RTT, time). This way we have retained features representing the user’s cell-ID, its signal strength, and the RTT max values. There were 41 different cell-IDs in the data set, RSSI values ranged from -51 db to -116 dB, and the RTT value ranged from 17ms to 209ms. Again, there was no missing data in this data set. First, the prediction task has been defined as a binary classification task, i.e., towards a prediction of the user throughput being below/above some predefined threshold. There have been multiple selections of throughput thresholds potentially matching the user’s required throughput: 20, 25, 30, 35, 40 and 50Mbps. This throughput corresponds to different required classes when e.g., streaming real-time video at different levels of quality (sizes and resolutions). Table 6 presents the results of the predictions for given binary throughput thresholds, the columns correspond to the ones in Table 5, this time applied to throughput.

Table 6: Predictions for given throughput thresholds

<table>
<thead>
<tr>
<th>Throughput Threshold [Mbps]</th>
<th>Accuracy of Educated Guess [%]</th>
<th>Decision Tree accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>84.60</td>
<td>84.40</td>
</tr>
<tr>
<td>25</td>
<td>67.80</td>
<td>74.17</td>
</tr>
<tr>
<td>30</td>
<td>53.70</td>
<td>74.17</td>
</tr>
<tr>
<td>35</td>
<td>61.41</td>
<td>75.27</td>
</tr>
<tr>
<td>40</td>
<td>72.59</td>
<td>78.89</td>
</tr>
<tr>
<td>50</td>
<td>87.80</td>
<td>87.00</td>
</tr>
</tbody>
</table>
For a prediction of throughput being below 25Mbps or above 40Mbps, an educated guess method will be preferred as it predicts the class with around 70-80% of accuracy. For predicting if the user throughput will be above/below 25, 30, 35 to 40Mbps, it is recommended to employ a decision tree and acquire accuracy of 74-78%, which may be sufficient to decide upon use/none use of streaming applications by the user (e.g., video streaming).

Additionally, we have experimented with cascading these predictions by firstly predicting the RTT value (along the six classes as in Table 5, employing the majority voting for classifiers to make a final decision on what is the predicted class), and, based on this predicted RTT value – in predicting the throughput class (along the seven classes as in Table 6, via majority voting as well). The educated guess is 13% for this approach; assuming cascading probabilities for prediction of RTT and then throughput. The overall result is accurate to 51%, i.e., around half of the time we will be able to indicate the correct interval of throughput. The most accurate (94%) prediction is for a throughput below/above 30Mbps. Using double prediction reduces the accuracy, but eliminates the need for active measurements, before deciding whether to allow the network request. This allows saving energy by not powering on the interface to send probe packets of a few kB, which still triggers the costly pre and post-connection high power states.

3.1.3.5 Conclusions

The goal of this work was to determine which information can be gathered from the network using a crowd-sourcing based measurement study. Analyzing the collected data, the following answers to the questions initially raised in the beginning of section 3.1.3 can be given:

**What information on the network state can be derived from “cheap” information like the cell ID?**
The results suggest that based on the cell density, first the overall deployed bandwidth at a given location can be determined, as the networks are usually upgraded according to traffic demand. Secondly, the existence of smaller cells increases the probability of higher data rates for the individual user, as the signal-to-noise ratio is expected to be better. Still, in peak times, these might also be congested. The information on the cell size can then be used in mobile traffic management to decide whether it is feasible to delay network requests until another cell is observed.

**Can congestion of the cellular network be recognized based on inexpensive local measurements?**
The cell-based performance of the mobile network, in particular the RTT, has been analyzed in a fine-grained manner. The measurements lead to the conclusion that for the observed LTE networks the end-to-end RTT depends more on the scheduler of the individual cell than the network backbone. Consequently, if the scheduler is busy, the utilization of the cell is higher, increasing the likelihood of higher RTTs and lower data rates.

**Can the cellular network throughput be predicted based on passive measurements only?**
The results indicate that for a prediction of RTT using an educated guess, accuracy between 70% and 80% can be achieved for the throughput prediction. Similar results (accuracy of 74% to 78%) can be achieved by employing a decision tree. Using the RTT prediction as an input to the throughput prediction an overall accuracy of 51% for 10Mbit wide bins can be achieved, and 94% to decide whether the rate is above or below 30Mbps. This could be leveraged in a real system if the user does not have data access and knows only the cell-ID and LAC and needs to decide, if it worth to switch on the data. This allows saving energy on the wireless interfaces by reducing the number of state changes. The prediction accuracy can be improved, if measured RTT values are known. These are “cheap” compared to more “expensive” throughput measurements, which would need to be made ad-hoc to quantify the current network throughput.

In the context of eCOUSIN, these results build the basis for the offloading decisions, as they determine the QoS achievable via the cellular network, and allow estimating the power consumption.
when transferring data at a given time and location. This allows scheduling of data transfers to a different time, location, or network technology (i.e. WiFi). Using the throughput predictions based on the passive measurements allows predicting the QoS of the available network independently of any data connection, while the crowd-sourcing approach relies on the aggregation of the collected data on the server. The server generated QoS maps can then be cached on the mobile device to allow offline decisions on when and how to connect to which network.

3.2 Network Resource Configuration

The objective of the “Network Resource Configuration” is to configure the infrastructure resources to perform the content delivery. This includes for example pre-reservation of network resources if required by the application, and configuration of the forwarding nodes in the network.

As outlined in D5.1 ALUD’s main objective in the project is to investigate and design network technologies that are both information-centric and practical and incrementally deployable at the same time. While the ICN networking model promises a number of benefits to users as well as service providers in terms of performance, security and mobility, these benefits come at a cost as many ICN proposals add significant complexity to the network design, e.g. by having routers acting as content caches, and supporting nearest-replica routing etc.

As outlined in D5.2 section 4.2, eCOUSIN proposed an ICN architecture concept conforming to a 2-tier architecture with a clear separation between forwarding plane and control plane. The forwarding plane is responsible for performing time-critical tasks such as address lookups and transferring packets from ingress to egress links. The control component consists of routing protocols. Through the exchange of routing information, the control plane provides the information required for performing network resource configuration tasks, i.e. the building and maintaining the routing tables of the forwarding nodes.

3.3 In-Network Content Routing/Caching

This workpackage also comprises an investigation of the relationship between the Open Social Network (OSN) and the information-centric networking layer. In deliverable D5.1 we have demonstrated the differences between OSN user behaviour and the respective behaviour of the networking layer. Accordingly we have proposed the OSN content delivery efficiency by taking into consideration the locality aspect of OSN end users. The routing is slightly modified compared with respect to the current OSN behaviour as local routing of data can be exploited thanks to the features of NDN. The routing of data towards the OSN server for non-local end-users but also for local users is kept in order to keep the OSN server aware of all data exchanged between end-users. But instead of having all end-users connected to the OSN server to retrieve data, we can have a local exchange of data between end-users located in the same (or close) region. For this, the CCN network is dynamically configured for routing interests for such content directly toward the local end-users themselves.

Technically mainly two options are envisioned to implement the dynamic configuration of content routing:

a. Since the ICN network is a CCN one, we propose to use OSPFN [OSPF12] together with CCN. OSPFN is a routing protocol similar to OSPF but announcing contents (or prefixes) instead of IP addresses (or subnetworks). With OSPFN, local end-users who want to make their content available in the network (e.g. when publishing a tweet) will announce it in the network via the OSPFN announcement message. When receiving this message, the CCN router will populate their local forwarding tables (FIB in CCN terminology) accordingly. This
announced message will be propagated in the network up to x CCN nodes (defined by configuration) or only within a domain (e.g. Autonomous System (AS)).

b. Since the network should be dynamically configured a dynamic solution exploiting Software Defined Networking (SDN)-like network programmability is taken into consideration. A candidate design choice is based on a logically centralized ICN controller, mitigating scalability issues inherent to flooding approaches as used by alternative ICN designs. Instead of having every node discovering the content topology and calculating best routes, the centralized controller (e.g., responsible for a single AS) will maintain the ICN topology information, implement routing logic, and configure forwarding tables of the ICN nodes. When an ICN node receives a content request, it routes the request towards source(s) matching the label of the request with its forwarding entries. This is line with the network resource configuration module presented in section 4.2.2.1.

4. ECOUSIN SOLUTION COMPONENTS

4.1 Solutions for Social-Enhanced Content Naming

This project studies the potential relationship between OSN and the network. As such, the naming of contents is also investigated to improve the match of this relation. The overall goal is to define a naming scheme for content in the OSN that would help to improve the delivery of the contents. The main objective is to name the content according to some OSN specifics, for instance the locality aspect which is taken into consideration.

The naming scheme strategy has first been introduced in D5.1 and further refined in D5.2. Here we remind it for the clarity and understanding of the rest of the deliverable. Since we investigate two different networking technologies (IP and ICN), we propose differing naming scheme, each one being specialised for its use (for IP, we keep the well-known URL/URI scheme. For ICN, a hierarchical content-based scheme is proposed). We present them hereafter, in line with their related use-cases.

4.1.1 NDN-based OSN delivery

The naming scheme for the CCN-based OSN delivery has been modified. As the adoption of the initially proposed naming scheme resulted in some issues with the NDN forwarding scheme it has been simplified. In our current scheme the differentiation between local/non-local users will be directly performed with routing. We propose to use a local routing scheme between users, and a dynamic configuration of routing states in the FIB of forwarding nodes, relying on the ICN controller presented in section 4.2.2.5.1.

4.1.2 Personal Sharing clouds (TI)

The naming scheme is a key part of the personal sharing Clouds use case it even contains somehow a self-explaining indication on how the use case works.

4.1.2.1 Personal Sharing Clouds Naming Scheme

The naming scheme strategy for the Personal Sharing Cloud use case has been already described, as we said each resource has the following structure:

```
```

The three main parts of the identifier are explained below:
1) Client-side Proxy location (Consumer Personal Cloud LAN), i.e.,
   http://192.168.1.10:8081;
2) Server-side Proxy location within the eCOUSIN overlay, i.e.,
   ecousin://nodeX:1276:tcp/;
3) Resource location from a Server-side Proxy point of view (Provider Personal Cloud
   LAN), i.e., http://10.0.0.5:8080/Pictures/myCat.jpg.

The proposed naming scheme is related to the choice to use the Universal Plug and Play (UPnP)
protocol both for local and remote content distribution in this use case. The naming strategy
developed in the use case has a URL/URI structure aiming at making UPnP agnostic of the real
physical location of resources. All resources client side are seen by the final user as local resources
via UPnP and filtering/access regulation is performed locally by the access manager module.

Adopting a URL/URI like naming scheme is motivated by the many advantages related to using well-
known open schemes.

The naming scheme above has been implemented into the prototypal version of the Personal Sharing
Clouds Use Case, shown in eCOUSIN intermediate review meeting in Torino. The prototyping phase is
in general useful to have a first indication about the feasibility of a certain solution.

4.1.2.2 Assessment/Update on the Personal Sharing Clouds Naming Scheme

In the reporting period related to this deliverable, a first stable prototype for the Personal Sharing
Clouds was developed and shown at the intermediate review meeting in Turin (running on a
commercial NAS Media Server behind an LTE). The Personal Sharing Cloud use case can actually work
due to the interconnection of different building blocks, described in different work packages.

During the prototype testing we did not found any reason to change this naming scheme which
seems actually to be the best choice to reach the desired goals for this use case. Moreover, even in
term of evolution of the prototype, this really does not seem to be a limitation in any way,
considering both the extension of the use case with the “Federated social Network” part and an
eventual change of local protocol.

In case one wants to change the “local” protocol (UPnP) to another one (such as DLNA), the URI
“modular” structure would allow this quite easily (even if of course a few modification to the
“Content Lookup” module, developed in WP4, would be necessary), while in terms of “Federated
Social Network”, as it will be explained later the chosen naming scheme is already compliant, since
the only module involved will be the Social Data Collector, described in WP3.

In this sense the prototype development and testing was a successful assessment plan for the
naming scheme.

4.2 Solutions for Social-Enhanced Content-Centric Network
   Infrastructures

4.2.1 Solutions for Content-Centric Routing in Personal sharing Clouds

As reported in the previous deliverables, the Personal Sharing Clouds use case aims at automatically
merging into a Cloud content belonging to different users, exploiting their social relationships to
propose and disseminate information about content production and the content itself. Here we recall how the system works.
### 4.2.1.1 Use Case elements definition and tasks

Once Personal Clouds are merged, users can discover and invoke services available in other Personal Clouds retrieving remote content as local, leveraging on local discovery protocols such as DLNA or UPnP.

![Figure 17: Personal Sharing Cloud use case](image)

Users accessing resources explicitly instruct and automate a discovery procedure which has to be repeated in order to check if new resources have been recently shared. Users have to periodically refresh the discovery phase.

As it was explained previously, the naming scheme gives the possibility to represent the whole process by means of the following URL:

```
```

Besides, access rules to content are set by the **Access Manager** module, providing locally access rules. Access Manager exploits Resource Locator (specified into WP4) to gather the set of shared resources, such as the set of available services and provided content, and then asks to the user which services/content should be visible to which identities/relationships. In this manner the Access Manager can know how to configure and apply filters to content, so that resources with different levels of privacy can be made visible differently to different set of specific users.

We adopt two filtering mechanisms to modify resource visibility in relation to identity/relationship (whitelist approach: it is allowed to access resources only if the user has explicitly allowed it):

- **Discovery Filter (DF)** - The Discovery filter tailors to differentiate accessible services among different classes of users. For example, remote users who are Facebook friends can be allowed to access and stream remote resources while users who are twitter followers. Even if this specific use case is targeted on media sharing, the system has been designed to allow another level of access differentiation based on services.

- **Content Filter (CF)** - users activate CF only for services that are perceived as particularly important; the goal is to support finer-grained access control to shared resources, eventually dropping unauthorized requests or sorting discovered resources based on priority preferences. CF implementation strictly depends on the target service and must be specialized for each service it is required to support.

To better explain DF and CF mechanisms, consider the following examples based on our UPnP service:
• The user can exploit DF to specify which subnets belonging to her network should be accessible via UPnP by other users, providing different access rules for different social relationships. For instance, the user could grant access to her Facebook friends to UPnP devices located in the home gateway subnet, while UPnP devices located in the office desktop subnet could be visible only to Facebook friends who are also members of the Facebook group “colleagues”;
• The user can exploit the CF mechanism to specify that only DLNA Media Servers are accessible from remote Network users, and only by Twitter followers, while the UPnP service controlling the home thermostat can be invoked only from eCOUSIN nodes within the same network

4.2.1.2 Use Case elements implementation progress and updates

As it was stated above, we presented a first stable prototype in Turin intermediate review meeting. After the prototype implementation an assessment phase was planned in order to check compliance with requirements and all the rest of architecture, both in terms of single modules and overall use case.

The assessment phase evidenced that Access Manager was a critical point in terms of hardware load. We looked at the processing time of a single request and by using logs we discovered that the main processing delay was introduced at the access manager level.

The problem was that all of the UPnP packets (even the ones not related to the application) were processed (and eventually discarded) by the Access Manager module. This problem introduced a considerable performance issue into network with high UPnP traffic load.

This lead to a change into the software: in the final prototype the traffic is filtered at a higher level into the architecture, so that the computational load is better distributed among the modules and the Access Filter is not overloaded.

The filtering operation of “useless” UPnP traffic anyway, did not solve completely the computational load problem anyway, even if it improved performance a lot. UPnP devices tend to produce a lot of traffic, due to the fact that they send loads of information in broadcast and this causes problem in environments such as media centres where the hardware is often not so powerful.

The next development planned on the use case is an extension of the use case by integrating the federated social networks into it. The Integration planned should be a twofold integration, including a mobile part (described in section 4.4.1) and a fixed part. For both of them the Access Manager and content naming are already fine, the only module touched by this integration will be the Social Data Collector (Described in Work Package 3).

4.2.2 Control Plane for Content-Centric Routing/Caching in ICN

The ICN Control Plane module provides content routing and caching functionalities as a network service rather than resorting to the conventional approach of leaving these tasks to the individual applications. We have studied how to apply first principles and concepts from ICN to meet the specific requirements of OSN applications.

Content sourcing, distribution, and delivery in eCOUSIN follows a different model than traditional content networks in which content is ingested to a few centralized locations that distribute and deliver the content in a top-down manner. We consider exploiting a widely available caching infrastructure (e.g. regional and metro data centres, potentially collocated and more tightly integrated with networking infrastructure potentially even in mobile end nodes) which can be used for storing locally generated content as well as for distributing and delivering information driven by
the algorithms developed in WP4. In consequence in some cases information will be generated and replicated very dynamically.

To efficiently deal with these properties, the eCOUSIN in-network routing and caching functional module provides mechanisms for publishing and propagating content availability potentially within different network scopes.

As outlined in deliverable D5.2 section 4.3, a candidate design choice is the use of a logically centralised control of the ICN nodes to mitigate scalability issues inherent to flooding approaches as used by Named Data Networking [NDNX14] and similar ICN designs. If an ICN node additionally hosts a small-size cache module, caching an object and delivering it from that local cache is a straightforward task.

However, the eCOUSIN “Network resource configuration” module can specify caching and cache eviction policies better aligned with OSN activities and based on knowledge of specific characteristics of the OSN layer than an isolated ICN node.

For instance, with extended knowledge about the locality aspects of the OSN interactions caching algorithm can be configured to preferably cache local content instead of caching multiple redundant copies of the same content along the path of the initial interest. In this case the cache hit ratio for popular items will be high and a small number of selected items can be fetched from a close cache. However, a large number of non-cached items (which correspond to the tail part of the Zipf distribution) may have to be downloaded from more distant places and its delivery cost is not marginal.

4.2.2.1 Overall ICN modules and interfaces design

An important driving factor for separating control and forwarding is the potential for distributed forwarding architecture and the scalability that this separation enables. For example, with the separation of control and forwarding components, time-critical forwarding-plane tasks can be distributed and optimized for the required performance.

Figure 18 provides a high-level overview on the ICN control plane architecture and on the general operation. The ICN controller has a complete view on the ICN overlay topology configuration (OSN content and node locations, content distribution policies etc.). The main task of the control plane is the definition of efficient content-dependent network-wide routing policies that have impact of the routing tables controlling the OSN content request routing behaviour of the service routers. The routing tables finally control the OSN content request routing behaviour in the network.

The main objective of our work is to investigate and design network technologies that are both information-centric and incrementally deployable, i.e., that support a step-wise departure from the host-centric paradigm starting off from limited partial deployments.

Since furthermore the network should be dynamically configured according to end-users publication as stated in section 3.3, a dynamic solution such as SDN-like network programmability has to be considered. Our resulting design choice is consequently based on a logically centralized ICN controller, mitigating scalability issues inherent to flooding approaches as used by alternative ICN designs.
4.2.2.2 ICN Controller architecture concept

The ICN controller concept was particularly developed to support incremental and cloud-based deployments in virtualised environments. In this context Software Defined Networking (SDN) seeks to simplify and enhance network management by decoupling the management logic from its implementation. To this end we have extend the SDN paradigm in the ICN Controller architecture by adding the capability to handle ICN node registrations manage the behaviour of ICN forwarding nodes as key enablers to content distribution policy enforcement in ICN.

As shown in Figure 19 the approach extends today’s SDN approach along a number of key dimensions:

1. **ICNControl API** between the ICN control application and the ICN forwarding nodes to configure the network-wide content request forwarding behaviour.

2. **ICN controller** that configures the behaviour of the ICN forwarding nodes. The input to the ICN controller is on one hand the policy that the administrator wants to enforce with respect to the FW nodes actions. On the other hand the ICN controller knows the ICN overlay topology, i.e., knows about the network topology and the location of the content providers and consumers. The three main operation modes of the controller are as follows:
   a. **Initialisation**: Given an input network topology and the events generated by the registration of the ICN FW nodes and their information we generate an overlay graph
   b. **ICN forwarding node event registration handling**: corresponding to each forwarding node instance the ICN controller maintains look-up tables to handle the registrations of forwarding nodes attaching to the ICN controller. Also the ICN node are added to the ICN topology which is also managed by the ICN controller
c. **Forwarding node FIB entry handlers**: per-node FIB entry generation and consumptions per forwarding node is also managed by the ICN controller. E.g., when a FIB entry expires or gets modified the ICN controller look-up table is updated accordingly.

d. **Routing**: FIB table modifications are pushed to the ICN forwarding nodes enforcing the decisions of a routing algorithm.

3. **ICN forwarding nodes** that account for on incoming content request by their identifiers, according to their respective FIB prefix entries and rules (i.e. FORWARD, DELAY, DMUX, DROP,...) as configured by their respective ICN controller.

---

**Figure 19**: Interfaces between different components in the ICNControl architecture.

The described ICNControl architectural approach enables a clear separation of concerns: While e.g. an OSN server has a detailed insight of the social graph between end-users and can provide this inside in form of notification sent to the ICN controller with appropriate social information parameters, the ICN controller relies on its knowledge of network topology to configure the NDN forwarding nodes accordingly to ensure the local routing between content sources (represented by end-users). The ICN controller application thus does not have to store any user-related or social information or information about user devices, but can decide to take social information and content distribution policies into account when calculating forwarding table updates based on the knowledge of network and content topology graphs.

As an example, the ICN controller can receive OSN information from the OSN server via the ICNControl API that the popularity of a particular content located at node A exceeds a pre-defined threshold. The controller then can start searching for local node B close to A (e.g., located in the
implementations for the realization of content routing.

As our main research focus is control functions, the required functionalities of a service routing system allowing for the mapping from content to the data plane: underlying forwarding nodes. In our work we have considered and realised two options for pushing routing information updates to the data plane:

This ICN node design enables abstraction from the characteristics of the specific capabilities of the underlying forwarding nodes. In our work we have considered and realised two options for pushing the routing information updates to the data plane:

a) The routing information can be used to make modifications to the forwarding/service tables of Serval-based ICN nodes [SERV14], where additionally the caching nodes functionalities and caching node update events can be emulated.
b) Alternatively, the routing information can either be used for modifications to the Forwarding Information Base (FIB) of some NDnx nodes [NDNX14] to support the scenario employing content-centric routing strategies for NDN and Twitter with automatic configuration of NDnx node FIB tables. Updates to caching can be tracked by deploying daemon modules which on registration of new contents send the corresponding registration messages to the ICN controller.

The required functionalities of a service routing system allowing for the mapping from content namespace to forwarding labels has been discussed in D5.1 in conjunction with a discussion of background work. As our main research focus is control functions, we have adopted already existing implementations for the realization of content routing system.

Figure 21: ICN node architecture

The ICN Agent offers an interface to the ICN controller to receive routing information from the ICN controller, containing information about how the node has to handle the requests for certain content. Additionally, the ICN Agent has to translate the routing information issued from the controller into a format that can be handled and executed by the particular ICN forwarding module.
As the additional requirement of an incrementally deployable ICN solution, if possible leveraging IP transport also has to be considered by considering the background work which has been presented in D5.1 in our realisations.

ICN nodes offer an API to the ICN controller which enables the configuration of service tables of forwarding nodes as described in the previous section. Since the network should be dynamically configured according to end-users content publication, a dynamic solution such as network programmability (SDN-like) has been taken into consideration. In particular we have adopted a number of concepts from the OpenFlow API for the realisation of our ICNControl API.

### 4.2.2.4 ICNControl Application Programmable Interface (API)

The ICNControl API suite is used for the communication between the ICN controller and the ICN Node in an Information-Centric Network. The API supports three types of messages: controller-to-node, asynchronous and symmetric messages.

#### Controller to node messages

The controller-to-node messages are sent by the controller to inspect the state of the forwarding nodes. The different types of controller node messages are the following:

**ICN_FEATURES_REQUEST/ICN_FEATURES_REPLY:**

This message is sent to request the capabilities of the ICN node, usually after ICNControl channel establishment. The ICN forwarding node in return will respond with a features reply message specifying the unique identifier and optionally the capabilities of the node.

**ICN_CONFIG_REQUEST/ICN_CONFIG_REPLY:**

This message is used to receive the configuration parameters of the ICN forwarding nodes. The following information can be registered:

- **ICNID:** unique identifier of the ICN forwarding node
- **ICN_FW_PROTOCOL:** forwarding protocol used by the node. Currently, the available options are “ccnx” or “serval”
- **VendorID:** identifier of the ICN vendor
- **DPIP:** data path IP of the forwarding node

**ICN_MOD:**

The ICN Modify-State message is used to add or delete existing FIB entries in the service tables of the forwarding nodes.

The parameters of the ICN_MOD detail the parameters and entries to modify in the FIB. ICN nodes with different implementations can interpret the parameters in different ways according to their particular requirements.

- ‘cmd’: command, can be either ‘add’ or ‘del’
- ‘id’: prefix
- ‘ip’: data path IP
- ‘weight’: weight parameter, used to control the balance in request forwarding if multiple entries for the same prefix exist in the service table.

**ICN_FIB_DUMP_REQUEST/ICN_FIB_DUMP_REPLY:**
The FIB dump message is used by the controller to read the current FIB from the ICN forwarding node.

**ICN_CACHE (not yet implemented):**
The cache command is sent by the controller to instruct a node to store content locally and serve back the content locally if a content request is received.

- cmd: “cache”
- “id”: ServiceID
- “url”: url_to_retrieve_content (optional)

**Asynchronous messages**
Asynchronous messages are sent by an ICN forwarding node in order to notify the ICN controller of ICN forwarding node states changes or on forwarding errors. As an example, an ICN_CACHE_ACK is sent to the ICN controller if a new content has successfully been added to the local cache.

**Symmetric messages**
Symmetric messages are exchange between ICN controller and switches and can be started by either one of them.

**ICN_HELLO/ICN_HELLO_REPLY:**
This message is exchanged upon the connection startup/registration of the ICN node.

**ICNControl connection setup**
The ICN controller-ICN forwarding node connection must be to a user’s configured URL and using user-specified port. When the connection is first established both the ICN node and ICN controller exchange ICN_HELLO and ICN_HELLO_REPLY messages with the highest version of the ICNControl protocol supported by the sender. Following the exchange of these messages, the lowest version of ICNControl protocol supported by both parties is selected. If that version of ICNControl protocol is supported by the both, the connection is established. Otherwise, an ICN_ERROR message will be sent back to the sender.

**Message Handling**
The ICNControl protocol provides reliable message transmission and processing.

**Message Delivery:**
The messages are guaranteed to be delivered, unless the connection between ICN controller and ICN forwarding node fails.

**Message Processing:**
Messages that arrive at the ICN nodes from the controller must be processed. If the ICN forwarding node cannot completely process a message should send an error message back to the controller. Furthermore the ICN nodes must send asynchronous messages generated by the change of state of the ICN forwarding node to the ICN controller.

Nonetheless, the ICN Node could drop some packages received on the data ports due to congestion or QoS policies. The ICN controller is free to drop packets, but must respond to the ICN_HELLO messages of the ICN nodes.
4.2.2.5  Implementation and Evaluation

4.2.2.5.1  ICN controller implementation

We have implemented the ICN controller as POX module and conducted experiments in a fully
virtualised lab environment using Open vSwitch instances on the host, and deploying each ICN
forwarding node and ICN Controller on a separate Virtualbox instance, to perform functional tests
and measure the performance and scalability of our approach.

According to the NOX/POX website [GUDE08], NOX [NOX014] was initially developed by Nicira and
donated to the research community and has become open source in 2008, making it to one of the
first open source OpenFlow controllers. It was subsequently extended and supported via the ON.LAB
[ONLA14] activity at Stanford University with major contributions from UC Berkeley and ICSI. POX
provides an API to OpenFlow as well as an asynchronous, event-based programming model.

POX is both a primordial controller and a component-based framework (Network OS) for developing
SDN applications. It provides support modules specific to OpenFlow but can easily be extended. The
POX core provides helper methods and APIs for interacting with OpenFlow switches, including a
connection handler and an event engine. Additional components that leverage the API also are
available, including host tracking, routing, Link Layer topology detection (LLDP), and a Python
interface implemented as a wrapper for the component API.

The POX SDN controller was adopted as starting point for the prototype realisations for the following
reasons:

- Open and modular design
- Reusable sample components for path selection, topology discovery, etc.
- Support of multiple Oss (optional bundling with installation-free runtime)
- GUI and visualization support
- Special support for Open vSwitch and OpenFlow 1.0

4.2.2.5.1.1  Repositories:

A number of repositories have been added to the implementation of the ICN Controller to maintain
the topology information and enable the control of registered ICN nodes:

- **icn_nodeinfo**: contains information about the ICN node configuration and connection
  identifier
- **icn_topo**: ICN nodes and edges information, visualised employing POXDesk
- **icn_content_map**: constructed from forwarding table information, contains information
  about published prefixes and locations (IPv4s) form all registered ICN forwarding nodes.

4.2.2.5.2  ICN controller GUI Overview

The web-based GUI of the POX controller based on POXDesk has been extended to allow for near-
real time monitoring of the content of the FIBs of connected ICN nodes. A new module has been
added for the near-real time display of the development of the ICN overlay topology in any web
browser (see Figure 22 - Figure 24). The ICN overlay topology view allows for monitoring of topology
updated e.g. introduced by new ICN node registration or de-registration events in near real-time.
Figure 22: Extended POXDesk GUI with new application entries, a) ICNTableViewer to monitor the FIB entries for ICN forwarding nodes and b) ICNTopoViewer to view the ICN network overlay topology.
Figure 23: ICN controller GUI showing the ICN overlay topology window with currently four ICN nodes attached to different hosts in the topology.
Figure 24: ICN controller GUI showing the respective ICN forwarding tables of all connected nodes shown in Figure 23.
4.2.2.5.3 Setup used for ICN-based LB experiments

In order to enable the demonstration of the ICN system in a first step local testbeds have been used. The different nodes or the ICN architecture described in the previous sections (i.e. ICN controller and ICN nodes) have been configured inside different headless Virtualbox virtual machines (VMs), each configured with different network interfaces to enable connectivity of the VMs as part of a layer 2 topology and to separate the traffic on the control path from the data path. An additional interface has been configured to enable remote access to each of the VMs via ssh.

As an example, a separated subnet has been defined to enable the connections from the ICN node to the ICN controllers. The screenshot below shows the demo setup including 4 ICN nodes, each connected to the ICN controller which shows the respective registration messages in the window on the bottom left of Figure 25.

Figure 25: Demo setup as employed for ICN-based load balancing measurements in the ALUD testbed, consisting of (from top left to bottom right) a) Server 1, b) Server 2, c) Consumer client, d) ICN service router, e) ICN controller running in their respective headless VMs configured with ssh access. All nodes employ the Serval forwarding module.
4.2.2.5.4 Tests involving different forwarding module implementations

A main feature demonstrating the flexibility of the selected overall concept employing a centralized ICN controller and the corresponding ICN forwarding nodes is that the concepts allows for abstraction of the specifics of implementations used in the forwarding plane.

In order to enable functional tests on this concept, an ICN forwarding node based on the current implementation of the NDN forwarding node has been realized. The node agent realizing the translation between the ICNControl protocol and the specific commands required for the NDN implementation has been realized as a wrapper script.

On startup the wrapper script realizes the registration of the ICN client with the ICN controller as described in the previous sections.

In order to enable forwarding of the current status and any update of the node’s name-based forwarding table to the ICN Controller, the actual content of the NDN node’s FIB needs to be accessed. The current implementation of the NDN forwarding module allows displaying of the internal status of a running ndnd daemon by pointing a browser at (for example): http://mylocalip:6363.

In our implementation the ndnd web browser interface is used to extract the required information about prefixes employing a parser as part of the wrapper script. The extracted information is subsequently formatted into an ICN_FIB_DUMP_REPLY message which is transmitted to the ICN controller where the received information can e.g. be visualized (see Figure 26).

Figure 26: Top left: ICN controller window showing forwarding tables of ICN nodes registered with the controller. a) ICN node employing the Serval forwarding module and b) ICN node employing the NDN forwarding node implementation both showing different content prefixes.
### 4.2.2.5.5 ICN Controller based dynamic load balancing

As part of the ICN controller application test, a time-dependent request routing policy has been implemented. The policy allows for dynamic balancing of the content request forwarding to a section of content servers. Following the time-dependent request routing policy the ICN controller pushes updates to the ICN Service Router node employing ICN_MOD command. The service router translates the command and implements the corresponding rules through FIB modifications.

In parallel the consumer issues a series of content requests for a particular content ID which are received either by server 1 or 2 according to the current policy implemented by the controller.

We have used a test topology containing five networked ICN nodes (Content Consumer, two replica servers ICN forwarding node and the ICN controller) in our measurements (see Figure 27).

![Figure 27: ICN testbed configuration used for request routing and dynamic load balancing, controlled by different entries in the SR forwarding table pushed by the ICN controller.](image)

Using the “weight” factor implementation of Serval-based forwarding nodes, the ICN Controller based time dependent load balancing policy has been configured to cycle through four different phases (100% means that 100% of the requests are handled by a particular server, see Table 7).

<table>
<thead>
<tr>
<th>Phase</th>
<th>Server 1 rel. load(%)</th>
<th>Server 2 rel. load (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>70</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>70</td>
<td>30</td>
</tr>
</tbody>
</table>

A script has been created to run download tests on a content id specified by the user on the consumer node. The script initiates content downloads by running a test client. After completion of
each download it registers the ICN server location for the download. For each measurement, 100 content requests have been evaluated by the script running on the consumer node.

The measured relative load distribution between two ICN servers after 1645 consecutive downloads following the time-dependent balancing policy as described above is shown in Figure 28. The square-ware shape in the relative load reflects the different phases imposed on the ICN service router by the ICN controller.

![Figure 28: Measured load balancing between servers following time-dependent policy phases.](image)

It can be seen that the relative load for server 1 cycles between 100%, 70%, 0%, 70% while server 2 cycles between 0%, 30%, 100% and 30%. Because of the implementation specifics of the Serval-based forwarding nodes the modes located at 30% and 70% show a FWHM of ~15%. The modes at 0% and 100% relative load do not show any noticeable broadening of the relative load distribution.
4.2.3 Solutions for Content-Centric Data Plane

In this section, we present our proposed architecture for a locality-aware NDN-based delivery system for OSN applications. With respect to the use case “Information-Centric and Social-Driven Content Delivery” described in the deliverable D2.1, we consider a Twitter-like application, which allows users to write short messages, but also share videos with their followers, and the whole process over our locality-aware and NDN-based architecture for the content delivery plane.

We first focus in this section on the data plane of the proposed architecture. It is worth of noting this section is an update of the description of the data plane we described in the deliverable D5.2. The modifications are mainly due to the fact we have simplified the naming scheme: we no longer have the differentiation of Local versus non-local users in the naming scheme (as mentioned in Section 4.1.2). As such, we propose an adaptive routing scheme, which preserves the locality behaviour of OSN end-users.

4.2.3.1 Adaptive Routing for Locality-Awareness in OSNs

In order to describe the adaptive routing concept in the following section, we will use concrete sample event sequences to explain the concept in more detail.

In these sequences, content objects that are disseminated or retrieved in the OSN are named within the NDN delivery infrastructure as follows: the name /Twitter/UserXXX/TweetAAA denotes a tweet AAA by a user XXX. A tweet is defined as any content that can be a text message, a video or a picture.

4.2.3.1.1 Publication of Tweets

When a user, e.g. TF1 in Figure 31, wants to post a tweet, an Interest message ndn://Twitter/SendTweet_UserTF1 is first sent to request the remote Twitter server to send a file (named “SendTweet_UserTF1”), containing the number, let say BBB, that TF1 needs to use to
name/number his tweet to publish. Upon reception of the Data message carrying this file, the user TF1 can then write his tweet and name it as /Twitter/UserTF1/TweetBBB.

After having sent the file “SendTweet_UserTF1”, the server prepares an Interest message for the tweet “TweetBBB” and sends the Interest to the user TF1, to let him post his tweet. On the user side (TF1), the application client is listening on the prefix /Twitter/ForServer_UserTF1 and when it receives the interest, it replies back to the Twitter server with a Data message containing the “TweetBBB” to publish, as schematized by the green line workflow in Figure 30.

Figure 30 depicts the call flow for publication of a tweet.

4.2.3.1.2 Locality-Aware Retrieval of Tweets

In order to guarantee that local followers that are located near the user TF1 can receive his tweets without spending time chasing the remote OSN server, TF1 is also required to announce his name prefix “/Twitter/UserTF1” in the network, using either a routing protocol such as OSPFN or a SDN-like approach, but with a limited TTL (Time-to-Live) value to reflect the locality end-users behaviour in OSNs. For example, with TTL = 2, the entry for “/Twitter/UserTF1” will then be present only in the FIB (Forwarding Information Base) of the local NDN routers being at most 2 hops from the end-user TF1. The Interest messages for the TF1’s tweets can then be forwarded using the longest name prefix match: i) either locally towards TF1 with the name prefix “/Twitter/UserTF1” in the FIB of local NDN routers if available; ii) or upwards to the remote Twitter server thanks to the prefix “/Twitter” when no entry exists for the user TF1 name in local NDN routers.

For illustration of our local routing scheme, let us consider the Figure 31 again. The TF1 user announces his name prefix “/Twitter/UserTF1” with TTL=2 as illustrated by the black dotted line workflow in the Figure. When Bertrand requests a tweet by TF1, e.g. named by /Twitter/UserTF1/Tweet23567, his Interest message is forwarded locally and directly to TF1 thanks to the TF1 name prefix announcement: TF1 is then responsible for serving Bertrand’s request. When delivering the tweet back to Bertrand, the intermediate NDN routers also cache the tweet in their
content store. This native caching feature in NDN allows the user Patrick to get the same TF1 tweet rapidly from the cache of the NDN access router when he will request it later. Now, if the user Jean-François wants to get the same Tweet23567, his interest message will be forwarded towards the remote server with the match of the prefix “/Twitter” in the traversed NDN routers. Indeed, Jean-François is located far away in a different network region, and hence he cannot benefit from the local name prefix announcement by TF1, which is only propagated within a limit of TTL=2.

Since end-users request data for the following tweet (the tweet numbered just after the one they previously received), it is possible that this one has not yet been published by the author. The interest message will be forwarded toward either the end-user herself or the Twitter server, but no reply data message will be sent back since the tweet does not yet exist. In NDN, every Interest message which has not been fulfilled is retransmitted after a given timeout (4s by default). However, the NDN interface allows configuring this timeout per interest. Following the Twitter behaviour we have analysed in D3.1, we configured the interest timeout for our application to 30s. As such, if an interest for a given tweet is not fulfilled it is retransmitted after 30s.

![Twitter-like application over a locality-aware NDN architecture](image)

**Figure 31: Twitter-like application over a locality-aware NDN architecture**

### 4.2.3.2 Dynamic Routing Configuration for Local Routing

For reducing the overload in the FIB tables of the NDN routers in the overall network, we can limit the announcement of the end-user name prefixes just for small periods of time: e.g. 5 minutes or any other value to determine to optimize the system. TF1 is then reachable locally during this 5
minute period, and TF1 needs to regularly repeat his prefix announcement if he wants to maintain the forwarding state in the local NDN routers.

However, for ensuring better scalability in the network, we adopt a more dynamical approach, based on the ICN control plane we present in section 4.2.2. A logically centralized ICN controller is set up to control the NDN routers, which allows to dynamically configure the NDN forwarding tables, 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). When the controller receives a routing update message, it re-calculates routes using topological and OSN information. If a newly calculated route requires modifications to one or more NDN forwarding elements, the controller communicates the changes to the ICN agents running on the impacted forwarding nodes. An ICN agent on a NDN forwarding node receives the instructions from the controller and then manipulates the required modifications in the FIB table accordingly.

For example, when a larger number of users connect to the OSN, the responsible OSN server registering the connection events can decide to send an update message to the ICN controller, which in turn checks if alternative sources for the requested content can be located in the near vicinity. In this case, an entry for the name prefix of the user will be pushed towards the NDN routers located close to the user (e.g. being at most TTL=2 hops from the user). Our proposed local routing scheme can thus be active with the interactions between end-users and the ICN controller.

Details about the communication between end-users, the OSN server and the ICN controller for setting up our locality-aware routing scheme will be provided in the following Section 4.2.4.

4.2.4 NDN-based nodes and interfaces

This section describes the implementation of the different nodes that are involved in our locality-aware NDN-based architecture. We also detail the networking interfaces for each node.

We have set up in our Orange lab the following network configuration for evaluating our locality-aware NDN-based architecture (see Figure 32). We deploy the software NDNx on different Linux machines, which then behave as NDN forwarding nodes for content routing and caching. These Linux machines are connected to a large-scale test network used by Orange for evaluating various networking services. As such, we can implement different network regions to simulate a deployment of our NDN-based network in nearly real conditions of an ISP.
4.2.4.1 **OSN server**

The OSN server hosts our Twitter-like application, which is developed in JavaScript as a node.js server application. This node.js server includes:

- a NDNx JavaScript library [NDNJ14] to implement the NDN stack at the OSN server level, so that the server can communicate with NDN routing protocol.
- a MongoDB [MONG14] to store detailed information about each end-user.
- an ICN agent to inform the ICN controller of information about end-users.

For example, the MongoDB database contains the following entries for the user TF1:

- TF1.following = ["France24","Orange","Bertrand"]; this array stores all users following TF1
- TF1.region = "1": the network region in which TF1 is located
- TF1.node = "b": the NDN forwarding node on which TF1 is connected
- TF1.ipaddress = "172.20.74.66": the current IP address used by TF1

To enable dynamic configuration of local routing (i.e. between OSN end-users that are located in immediate vicinity as described in Section 4.2.2), we rely on the logically centralized ICN controller we present in Section 4.2.3. As a consequence, we also implement the ICN agent inside the node.js server as a thread to notify the ICN controller (based on social information retrieved from the MongoDB) about routing states that need to be modified for the NDN forwarding nodes. For
example, when TF1 is online (resp. disconnected) on the OSN, the OSN server notifies the ICN controller that this user should serve as content source and therefore an entry for the prefix "Twitter/UserTF1" need to be added (resp. removed from) into the FIB of the forwarding nodes located TTL=2 (or any other value to be configured) from the node ccn_1b on which TF1 is currently connected. The ICN controller, based on its knowledge of network topology, can then identify that the forwarding nodes ccn_1b and ccn_1a are affected by this FIB modification. In a similar way, when Bertrand is disconnected from the OSN application, using the MongoDB database, the OSN server can check the list of users Bertrand follows to evaluate if the forwarding node to which Bertrand is linked can be cleaned by removing the entries corresponding to Bertrand's followers (in the case when there is no other user using those entries).

We actually make a separation between the roles of the OSN server and the ICN controller. Thanks to the MongoDB database, the OSN server has a detailed insight of the social graph between end-users, and can then provide and notify the ICN controller with appropriate social information about any end-user (who has just been connected, is offline, who followers who, etc). Then, the ICN controller relies on its knowledge of network topology to configure the NDN forwarding nodes accordingly to ensure local routing between end-users.

We develop the User Interface of our Twitter-like application as a web server application in a HTML5/CSS3 based responsive design using Knockout.js template (for separating UI views from data based on the Model-View-Model pattern) and REST communication to communicate with NDNx JS library and the MongoDB database. As a result, end-users can use and connect to our Twitter-like application on any device (mobile, tablet or desktop computer) using a simple web browser.

The following Figure 33 illustrates the OSN server architecture. Detailed information about the Web technologies we use for the implementation is provided in the deliverable D6.3.

Figure 33: Architecture of the OSN server
The following Figure 34 shows the web interface an end-user can see when connecting and logging to our Twitter-like application via a browser. As you can see, users can send short messages (like with official Twitter) and also watch videos shared by their friends.

Figure 34: Interface of our Twitter-like application using any web browser

4.2.4.2 NDN Forwarding Nodes

The forwarding node architecture (Figure 35) relies on the NDN architecture and hence includes the traditional NDN functional modules for name-based routing:
• a Forwarding Information Table (FIB), used for routing Interest messages to a potential source (that stores the requested content) or another forwarding node;

• a Pending Interest Table (PIT), used for managing multiple similar Interests and for delivering Data messages following the reverse path;

• a Caching module that maintains and manages a Content Store, used to cache content objects and to serve any Interest message if the requested content is in the cache;

• a Content Publication module which allows to configure the FIB, according to the in-path network advertisement messages (using the OSPFN protocol between NDN-based nodes to publish and announce content availability). In particular, at startup, the OSN server needs to advertise its name /Twitter in the network using OSPFN to get accessible by end-users. The Content Publication is then responsible for adding in the FIB of the forwarding node the name prefix “/Twitter” as announced by the OSN server.

The NDN forwarding node also includes an ICN agent that can modify the FIB depending on received routing configuration requests from the ICN controller. These configuration requests from the ICN controller are ICN Modify-State (ICN_MOD) messages used to add or delete routing entries in the FIB of the forwarding node, in order to dynamically activate local routing.

4.2.4.3 End-User nodes

There is no requirement for end-users. They simply connect to our Twitter-like application using a web browser to load a http link, that points to the OSN server. Then, the user device gets the NDN networking capabilities thanks to the browser that downloads the NDNx JavaScript library embedded in the node.js server.

4.3 Solutions for Social-Enhanced Mobile Infrastructures

4.3.1 Mobile federated social networks

As explained before, as well as from the point of view of the different modules, belonging to different work package, the first prototype shown at the intermediate review meeting in Turin did not include the federated social network.

The final step for finalizing the prototype in WP6 will then be the inclusion of federated social network, which is planned to be a twofold inclusion, strongly tied to the Opensocial [OSOC14] and OStatus [OSTA14] protocol suite which is now becoming a W3C standard, thanks also to the projects effort in standards.

The Opensocial/Ostatus Protocol Suite aims at standardising the interaction among different social networks, in order to overtake the typical “walled garden” paradigm. The basic idea is to have an account on a preferred social network, which is able to give access to some content (matching specific rules) to users belonging to other social networks.

This last feature could open the way to a scenario in which the “social network” is actually a “home social network” linked to other “home social networks”, thanks to software running on media centres itself. The Opensocial/Ostatus Protocol suite (which is now becoming a W3C standard) covers both client to server and server to server communication. That is why we envisioned a twofold level of integration with the federated social network:

- Exploit Opensocial to grab info on how to connect to my friends’ remote media centre searching for this information into the profile data (involving mainly the Social Data Collector module).
- Exploit Opensocial Client/Server specification to publish data on the media centre: In this case the files published standing on specific folder into the media centre will automatically get accessible as local resources from remote endpoints according to the normal behaviour of the use case

4.3.1.1 Mobile federated social networks integration plan

To integrate a mobile part we could leverage on the integration of the second type. The implementation of such a solution requires a web server with PHP, but in fact a webserver has already to be active on the mediacentre, so the implementation just consists of adding a specific module without any need to change the other modules in the architecture.

Following the OpenSocial specifications, the unique user bob@domainB, logged in using the mobile device should be able to publish multimedia content using HTTP directly on his/her own mediacentre, with a specified request format, which is basically activitystreams over HTTP. Activitystreams [ACST14] is a standardization effort to define common types of objects and actions (verbs) taken on various social media sites. The mediacentre exposes a RESTful web interface on top of which users can perform their operation via HTTP.

The RESTful interface exploits a number of endpoints specifying the type of operation using the HTTP method. For example, a user posting a note on a federation end-point will request for:

```
POST /osapi/activitystreams/acct:user@ecousin.domain.eu/@self
HTTP/1.1
HOST ecousin.domain.it
Authorization: Bearer hh5s93j4hdidpola
Content-Type: application/json
Accept: application/json
Content-Length: ...

{
    "actor": {
        "id": "acct:user@ecousin.domain.eu",
    },
    "object": {
        "objectType": "note",
    },
    "location": {
        "position": "+48.52+002.20/"
    },
    "title": "This is my new status update",
    "verb": "post"
}
```

For each requests the (endpoint,method) couple, the above specifies the type of operations to perform. Main endpoints will provide /osapi/activitystreams/ for access to activities and /osapi/people/ for access to user's data.

So this kind of integration just requires the implementation of the Opensocial Stack on both mobile terminal and media centre and to set the repository directory so that it is visible from remote eCOUSIN endpoints.
4.3.2 Mobile network offloading in social places

4.3.2.1 WiFi-Direct-capable Content Synchronizing Application for the Android OS

During the development of a Device-to-Device (D2D) content delivery mechanism, it is beneficial to gain insight into metrics of the all-day usage of D2D. Therefore, we are developing PrivateShare, an application allowing a user to share content with friends via D2D. In this context, content may be a photo album, a music folder, or folders of arbitrary files. Whenever the user changes her shared content collection (e.g. she adds music or videos), the content gets updated on friend’s devices whenever they are in communication range.

With every friend, the app establishes mutual trust between friends by pairing. In an initial version, the pairing may be achieved via Near-Field Communication (NFC) or Quick Response (QR) code scanning. However, the mechanism is kept extensible.

The app provides a one-hop, delay-tolerant, incremental content exchange: Whenever devices are in range, they automatically update content shared with each other. While having a benefit, the user does not have mobile carrier costs, and additionally his privacy is ensured.

The app is prepared for a field study: It anonymously collects evaluation data on a central server. Transparency is additionally ensured, as every user has the opportunity to review the data before submitting it. The results obtained in the field study shall contribute to the following questions:

- In which situations should D2D content exchange be preferred over infrastructure mode?
- How large are the incentives for a device user to use D2D content delivery?
- How should D2D content exchange be configured and used in practice?
- How can the underlying technologies (WiFi, Bluetooth) be improved?
4.3.2.1.1 Use Case Specification

Users have to specify friends and share cryptographic secrets with them. This can be done in an initial authentication phase. A user transfers the identity of a friend to its local database via NFC or QR Code (see also Figure 40). Furthermore, the users specify different folders or other content collections on their phone which they want to share with a certain group of friends (which they also specify).

Content shared with friends is then automatically transferred to friends and kept up-to-date whenever they are in range, and without any user interaction.

![Figure 37: Initial Phase before synchronization](image)

Figure 37 shows the initial phase in an example. Alice shares Movie 1, Movie 2 and Song 4 with Bob. Bob also shares content with Alice. As the peers are not in range, no synchronization takes place.

![Figure 38: Synchronization during devices being in range](image)

Figure 38 depicts the synchronization phase. The devices are in range and have discovered themselves. The movies and songs are then synchronized. Bob gets all the content of Alice he does
not have and vice versa. After the synchronization phase, devices get out of range again, but have synchronized the content for later usage (Figure 39).

Figure 39: Devices after Synchronization

4.3.2.1.2 Metrics Collected

With the PrivateShare app, we want to get insight into various metrics regarding device to device user behaviour and environmental conditions. The metrics are always collected anonymously and include:

- The number of devices in range, regarding all WiFi-detectable devices. The application uses a periodic polling for devices in range.
- The duration of a device being in range and the time available to communicate. This will allow determining if a device is online sufficiently long enough for transfer of the desired content.
- The Signal strength development during a device being in range. This is currently not available, as the Android API drivers are broken.
- The upstream and downstream bandwidth available and used. The available bandwidth is measured only once per device to assess the transfer capabilities.
- The success of D2D content transmissions. This includes the size of content and the cause (e.g. a device out of range)
- The ratio of successful transfers in all transfer attempts (i.e. the fails per byte).
- Implementation aspects in Android, for example failed attempts to use the API. Reasons may be missing support or broken WiFi Peer-to-Peer stacks.
4.3.2.1.3 Goal of Field Study

The PrivateShare app will be rolled out on the TUD campus, and will be advertised in lectures, seminars and exercises. The study will last 1-2 months. Like the mobile provider trace analysis (D3.3, Section 2.7), this field study has the aim to evaluate the potential of D2D content delivery. However, it is complementary to the latter:

- The provider trace analysis provides a macroscopic view on a large user base. However, it is based on a restricted set of metrics (content consumption, users in same cell): the real D2D operation is only assumed and simulated through a model.
- The PrivateShare app instead provides a fine-grained view on a smaller user base and the behaviour in using D2D mechanisms. PrivateShare is residing on the data collection plane in the eCOUSIN infrastructure. Furthermore, it fulfils some of the Network Monitoring tasks, like device proximity detection and availability.

The detailed results of the field study evaluation will be reported in an upcoming WP6 deliverable (i.e., in deliverable D6.5).

4.3.3 Mobility-aware interest-driven data flow and time-unconstrained content delivery

4.3.3.1 Novel mobile traffic offloading algorithms using OSN information

A highly interesting trend in mobile network optimization is to exploit knowledge of future network capacity to allow mobile terminals to prefetch data when signal quality is high and to refrain from communication when signal quality is low. While this approach offers remarkable benefits, it relies on the availability of a reliable forecast of system conditions. In this section we will focus on the lower layer networking aspects than enable the application of such traffic offloading algorithms. We will analyze the reliability of simple prediction techniques and their impact on resource allocation algorithms. Our proposed resource allocation technique, that is robust to prediction uncertainties,
can be found in D4.3. This algorithm combines autoregressive filtering with statistical models for short, medium, and long term forecasting and is validated by means of an extensive simulation campaign for different network scenarios.

4.3.3.1.1 General forecast model

In this section we propose a general model describing the forecasting reliability of a system. In particular, we split our model in three time periods based on the prediction horizon:

- **The short term period** considers the near future and predicts capacity through time-series filtering techniques [SADE04][QIAO04]. It is characterized by the reliability time \( \tau_{p} \), which defines how many slots of the sequence can be predicted and that we will discuss in Section 4.3.3.1.1.1.
- **The medium term period** describes the evolution of the system in terms of available capacity statistics. During this period one or more network cells can be accounted according to the mobility predictor: Markovian predictors [NICH08] can usually compute the likelihood of visiting a given cell, while trajectory-based predictors [FROE08] provide a more accurate estimate by computing the actual distribution of the user position along time.
- **The long term period** provides an overall statistical evaluation of the available capacity based on the steady state distribution of the user position in the network. Both the medium and the long term periods are discussed in Section 4.3.3.1.1.2.

Our resource optimization algorithm is presented in D4.3.

4.3.3.1.1.1 Short term forecast with filters

This section addresses the reliability time \( \tau_{p} \) achievable by filtering techniques applied to available capacity time series. In particular, we study autoregressive-moving average (ARMA) filters and their setup according to the system dynamics defined by the slot time \( t \) and the user speed \( v \). We opted for ARMA instead of GARCH [SADE04], since capacity elements belonging to the short term period are characterized by the same finite variance.

For each \((t \in [0.5,5], v \in [0.5,5])\) tuple we consider a set of 100 capacity traces computed as per [OSTE11] starting from the mobility paths of a user moving at constant speed in a random network deployment. We apply the Box-Jenkins [MAKR97] method to determine the type and the order of the filter to be used with each sequence. Through the analysis of autocorrelation and partial autocorrelation plots, we find that the best technique for our sequences consists of simple autoregressive (AR) filters of order \( \tau_{F} \), and that \( \tau_{F} \) is inversely proportional to the \( tv \) product.

Subsequently, for each of the sequences we estimate filter coefficients by means of the linear least squares procedure [HAMI94] and we use the obtained filter to forecast the values of the other sequences with the same \((t, v)\) parameters. We refer to a forecast sequence as \( \tilde{C} = \{\tilde{c}_{i} \in [0,C_{max}], i \in \mathbb{N}\} \), obtained from \( C \) and to the corresponding error \( \Delta = \{\delta_{i} = \tilde{c}_{i} - c_{i} \in [-C_{max},C_{max}], i \in \mathbb{N}\} \).

We consider a prediction to be reliable as long as the error \( \Delta \) is statistically smaller than estimating the capacity from its distribution \( f_{C}(c) \) or, in other words, when the standard deviations of the two processes are equal \( \sigma_{\Delta} = \sigma_{C} \).
Thus, we compute \( \mu_\Delta \) and \( \sigma_\Delta \) as the average and the standard deviation of all the error sequences with the same \((t,v)\) parameters. Figure 41 shows on the abscissa the prediction time index normalized on \( t \) and on the ordinate \( \sigma_\Delta / \sigma_c \) the standard deviation of the prediction normalized on the standard deviation \( \sigma_c \) of the original series \( C \).

Figure 41: The shaded area represents how the standard deviation of the short term prediction error increase with increasing prediction distance varying the user speed \( v \) and the slot time \( t \). \( \tau_p \) represents the time after which \( \sigma_\Delta \geq \sigma_c \).

While the actual steepness of the curves varies with the parameters, for all of them the normalized error standard deviation \( \sigma_\Delta / \sigma_c \) approaches 1 almost linearly. Hence, we set \( \tau_p = \arg\min_i s.t. \sigma_i / \sigma_c > 1 \). In addition, we observe that both \( \tau_p \) and the filter order can be approximated with simple linear models with the inverse of the \( tv \) product and that \( \tau_p \) is usually $10$ times as large as the order of the AR filter.

Finally, it is sufficient to tune a set filters for varying \( t \) and \( v \) and select the one to use according to the actual user mobility. Also, since filters can be normalized on \( \sigma_c \) it is not needed to have different filters for different numbers of active users in the cell, but it is sufficient to rescale the constant and the variance parameters of the filter.
4.3.3.1.1.2 Statistical models and uncertainties

This section describes the second technique of our general forecast model, which adopts statistical models to describe the user capacity availability for medium and long term prediction. In particular, in order to describe the distribution of per user capacity we started again from the model proposed in [OSTE11], since to the best of our knowledge it is the only one which takes into account the scheduler impact and thus is able to model user contentions.

In order to account for the impact of uncertainties on the user position and/or the number of active users in the cell we need to modify the expression of the capacity distribution $f_C(x)$ obtained for a specific position $p_i$ and number of users $n_i$ to the actual distribution of the user position $f_P(x)$ and the probability mass function $f_N(n)$ of the number of active users in the cell. This can be achieved through the following equation:

$$f_C(x) = \sum_{i \in N} f_N(i) \int_0^\infty f_{F,P}(g_C^{-1}(x,p), p | i) \left| \frac{\partial g_C^{-1}(x,p)}{\partial x} \right| dp$$, (9)

where $f_{F,P}$ is the joint distribution of fading and position, $g_C$ is the function linking the per user capacity to $p$ and $n$ and $N$ is the support of $f_N(n)$. Since fading and user position are statistically independent their joint distribution $f_{F,P}(x,y) = f_F(x)f_P(y)$ is the product of their distributions. Eq. (9) modifies the original capacity distribution weighting it through the active user probability mass function $f_N(i)$ and the user position probability $f_P(y)$; the partial derivative normalizes the integrand.

For what concerns our analysis, it is sufficient to be able to compute the per user capacity distribution by accounting for limited knowledge of the user position and traffic in the cell by means of their respective distributions.

So far, our model describes capacity only for the case when the cell the user is connected to is known perfectly. However, to account for different cells it is sufficient to consider the weighted sum of the capacity distributions of single cells $f_{C,i}(x)$ is the capacity distribution related to cell $i$ and $\rho_i$ is the probability of visiting cell $i$ in the next time period, we obtain:

$$f_C(x) = \sum_{i \in C} \rho_i f_{C,i}(x), (10)$$

where $C$ is the set of cells that can be visited in the next time period with a probability $\rho_i > 0$.

Figure 42 provides a few examples of the CDF obtained using the model. The solid line is representative of the capacity CDF $F_C(x)$ when both the active user number $n = 5$ and the user position $p = 500$ meters are exactly known so that the distribution is equal to the fading distribution. The dotted line accounts for an error on the number of active users in the cell so that $f_N(x) = \{0.2, 0.6, 0.2\}$ for $x = \{4, 5, 6\}$ respectively. Conversely the dashed line is obtained by accounting for an error on the user position which has a normal distribution with parameters $\mu_p = 500$ meters and $\sigma_p = 100$ meters. Finally, the dash-dotted line is obtained by mixing together two cells with 5 and 10 users with 20 and 80% of visiting probability respectively. The piecewise-constant shape of the curves is due to discrete relationship between SINR and bitrates.
Figure 42: A few examples of the impact of imperfect knowledge on the capacity distribution: the solid line is obtained with accurate information; the dotted and the dashed with imprecision on $N$ and $P$ respectively, while the dash-dotted considers two cells with different $N$.

In a practical implementation of this solution, the capacity statistical models should be known in advance, while the user position statistic $f_p(x)$ and the cell traversal time $\tau^{(i)}$ will be obtained by analysing user mobility patterns; the number of active users statistic $f_N(y)$ in the cell will be estimated from historic information about that cell at that time of the day.

Finally, to define the statistical model for the long term period, we consider a capacity distribution obtained as a mixture of all the capacity distributions of cells in the network weighted through their steady state visiting probability.

5. MAPPING TO REQUIREMENTS

In this section a mapping of the modules and interfaces implemented in final version (V2) as part of the work in this workpackage back to the requirements defined in deliverable D5.1 is provided in. This summary is also relevant for the identification of items that have also been transferred into the eCOUSIN WP6 demonstrators. This workpackage has supported the project demonstrators by delivering modules towards the Information-Centric Networking (ICN), Personal Sharing Cloud (PSC), and Content Offloading for Mobile Networks (COMN) demonstrators.
Table 8: Final Components versus Requirements mapping indicating transfers to Information-Centric Networking (ICN), Personal Sharing Cloud (PSC), and Content Offloading for Mobile Networks (COMN) demonstrators.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Component Description</th>
<th>Related Requirements defined in D5.1</th>
<th>Transferred to demo</th>
<th>Partner</th>
<th>Ref. in Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Controller function to communicate and enforce routing decisions from the control plane to the forwarding plane</td>
<td>SENIR01, SENIR02, SENIR03, SENIR04</td>
<td>ICN</td>
<td>ALUD</td>
<td>4.2.2.2</td>
</tr>
<tr>
<td>2</td>
<td>Service routing system to perform a mapping from content name space to forwarding labels</td>
<td>SENIR01, SENIR02, SENIR03, SENIR05, SENIR15</td>
<td>ICN</td>
<td>ALUD</td>
<td>4.2.2.3</td>
</tr>
<tr>
<td>3</td>
<td>CCN based forwarding node and interfaces</td>
<td>SENIR01, SENIR03, SENIR13, SENIR14, SENIR 15, SENIR 16, SENIR 17</td>
<td>ICN</td>
<td>FT</td>
<td>4.2.3.2</td>
</tr>
<tr>
<td>4</td>
<td>CCN based end user node and interfaces</td>
<td>SENIR01, SENIR03, SENIR13, SENIR 15, SENIR 17, SENIR 18</td>
<td>ICN</td>
<td>FT</td>
<td>4.2.3.3</td>
</tr>
<tr>
<td>5</td>
<td>Client Side/Server-Side Proxy Server and Access manager Modules</td>
<td>SENIR08, SENIR09, SENIR10, SENIR11, SENIR12</td>
<td>PSC</td>
<td>TI</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>6</td>
<td>Home Mediacentre with functionality for federation between social network</td>
<td>SEMI01</td>
<td>PSC</td>
<td>TI</td>
<td>4.3.1.1</td>
</tr>
<tr>
<td>7</td>
<td>Network bandwidth availability prediction module (load, cost, QoE)</td>
<td>SEMI05, SEMI06, SEMI07, SEMI08, SEMI09, SEMI10, SEMI11, SEMI12, SEMI13, SEMI14</td>
<td>COMN</td>
<td>IMDEA</td>
<td>4.3.3.1</td>
</tr>
<tr>
<td>8</td>
<td>Mobile network bandwidth measurements Module</td>
<td>SEMI15, SEMI17</td>
<td>COMN</td>
<td>IMDEA</td>
<td>3.1.2</td>
</tr>
<tr>
<td>9</td>
<td>PrivateShare D2D Study</td>
<td>SEMI06, SEMI08, SEMI09, SEMI11</td>
<td>COMN</td>
<td>TUD</td>
<td>4.3.2</td>
</tr>
<tr>
<td>10</td>
<td>Measurement Study of Cellular Service Quality</td>
<td>SEMI15, SEMI17</td>
<td>COMN</td>
<td>TUD</td>
<td>3.1.3</td>
</tr>
</tbody>
</table>
6. CONCLUSIONS

This deliverable presents the final results of work achieved within WP5 during the second year of the project, including an updated survey of related work, and the final description of the eCOUSIN solutions for social-enhanced content centric and mobile network infrastructures on individual module and interface level.

It also represents the description and final implementations of the functional architecture at infrastructure level in terms of separate modules and interfaces and a description of their respective interactions.

The **social-enhanced content naming scheme** has been modified with respect to initial proposals to adopt the requirements while improving the backward compatibility with background work.

The **Personal Sharing Cloud** provides a solution to automatically merge Cloud content belonging to different users by exploiting their social relationships in order to propose and disseminate information about content production and about the content itself. Moreover it integrates the Mobile Federated Social Network standard. The proposed solution for **Mobile Federated Social Networks** features a twofold inclusion strongly tied to the Opensocial+Ostatus protocol suite, which is now becoming a W3C standard thanks to the project’s effort in standards.

The Opensocial/Ostatus Protocol Suite itself aims at standardizing the interaction among different social networks, in order to overtake the typical “walled garden” paradigm. The solution enables a user in a given OSN to grant access to selected content (matching specific rules) to other users belonging to other OSN.

eCOUSIN also proposed an **ICN architecture** concept conforming to a two-tier architecture featuring a clear separation between forwarding and control plane. The ICN forwarding plane is responsible for performing time-critical tasks such as content identifier lookups and transferring packets from ingress to egress links. The control component consists of routing protocols which can be executed e.g. by an ICN controller. Through the exchange of routing information, the control plane provides the information required for performing network resource configuration tasks, i.e. the building and maintaining the routing tables of the forwarding nodes.

On the forwarding plane the ICN node design enables abstraction from the characteristics of the specific capabilities of the underlying forwarding nodes. Two options have been considered and realised for pushing the routing information updates to the data plane. On one hand the information can be used to make modifications to the forwarding tables of Serval-based ICN nodes. On the other hand, the routing information can either be used for modifications to the Forwarding Information Base (FIB) of NDNx nodes to support scenarios employing content-centric routing strategies for NDN and Twitter with automatic configuration of NDNx node FIB tables. The required functionalities of a service routing system allowing for the mapping from content namespace to forwarding labels has also been discussed in D5.1 in conjunction with a discussion of background work.

A Twitter-like application running on the OSN server featuring a web GUI has been created. The app allows OSN users to send short messages (like with official Twitter) and also watch videos shared with their friends.

A **measurement study** of cellular service quality has been performed. The goal of this work was to determine which information can be gathered from the network using a crowd-sourcing based measurement study. The obtained results build the basis for the offloading decisions, as they determine the QoS achievable via the cellular network and allow estimating the power consumption when transferring data at a given time and location.
This allows scheduling of data transfers at a different time, location, or employing a different network technology (i.e. WiFi). Using the throughput predictions based on the passive measurements allows predicting the QoS of the available network independently of any data connection. The crowd-sourcing approach relies on the aggregation of the collected data on the server. Server generated QoS maps can then be cached on the mobile device to allow offline decisions on when and how to connect to which network.

Additionally, an application (PrivateShare) allowing a user to share content with friends via D2D has been developed. Content in this context can represent an album or a folder containing media or arbitrary files. Whenever users modify shared content collections, the content on friend’s devices gets updated whenever being in communication range.

The application establishes mutual trust between friends by pairing which can be achieved e.g. via Near-Field Communication (NFC) or QR code scanning, the pairing mechanism itself is kept extensible. PrivateShare provides a one-hop, delay-tolerant, incremental content exchange. Whenever devices are in range for direct communication, they automatically update content shared with each other. While having a benefit, the user does not have mobile carrier costs, additionally the user privacy is ensured. Goal of the app, which is accompanied by a research study, is to get an insight into usage behaviour and environmental conditions when using D2D communication.

A lightweight measurement technique that leverages adaptive filtering over the packet dispersion time has been presented which allows the estimate of the available bandwidth in mobile cellular networks. Accurate estimates can be achieved exploiting as few as 5% of information obtained from TCP data flows. Given that this solution can support dense throughput sampling, it is ideal for bandwidth prediction and optimized resource allocation. With the knowledge of the future bandwidth availability it is possible to predict when it is best to communicate. In addition, the solution is able to estimate the fast capacity variations from a mobile terminal by monitoring the traffic generated under normal daily usage.

The technique has been validated in a week-long measurement campaign spanning different locations, devices and communication technologies. Good estimation accuracy has been achieved even when using only short lived TCP connections. Since the technique is based on simple postprocessing operations on the packet timestamps it is possible to integrate it into background processes or OS routines.

In summary, the design and realisations of the eCOUSIN modules and interfaces, which are among the main results expected by the project, has reached its final milestone, and an updated survey of existing work has resulted in an improved understanding of the missing aspects of current solutions, paving the way towards improvements through the implementations on the network-level of the eCOUSIN architecture.
REFERENCES

[ACST14] Activitystreams: activitystrea.ms/, retrieved December 2014-


[CHEN14] A Mechanism of Information-Centric Networking based on Data Centers, M. Chen et al., In Proceedings of 2013 International Conference on Advanced Cloud and Big Data


[NOX014] Both NOX and POX information can be accessed via http://www.noxrepo.org/forum/, retrieved December 2014


Copyright © eCOUSIN Consortium, December 2014
Conference, 2013.


### GLOSSARY

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Access Point</td>
</tr>
<tr>
<td>AS</td>
<td>Autonomous System</td>
</tr>
<tr>
<td>ASU</td>
<td>Arbitrary Strength Units</td>
</tr>
<tr>
<td>AVP</td>
<td>Attribute Value Pair</td>
</tr>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol</td>
</tr>
<tr>
<td>BSD</td>
<td>Berkeley Software Distribution</td>
</tr>
<tr>
<td>CBCB</td>
<td>Combined Broadcast and Content Based</td>
</tr>
<tr>
<td>CCN</td>
<td>Content-Centric Networking</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Distribution Network</td>
</tr>
<tr>
<td>CDR</td>
<td>Call Detail Record</td>
</tr>
<tr>
<td>CS</td>
<td>Content Store</td>
</tr>
<tr>
<td>DB</td>
<td>Data Base</td>
</tr>
<tr>
<td>D2D</td>
<td>Device to Device</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Table</td>
</tr>
<tr>
<td>DLNA</td>
<td>Digital Life Network Alliance</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DONA</td>
<td>Data-Oriented Network Architecture</td>
</tr>
<tr>
<td>eNodeB</td>
<td>Evolved NodeB</td>
</tr>
<tr>
<td>ESMN</td>
<td>Extended Social Media Network</td>
</tr>
<tr>
<td>FIB</td>
<td>Forwarding Information Base</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>FSN</td>
<td>Federated Social Network</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HSPA</td>
<td>High-Speed Packet Access</td>
</tr>
<tr>
<td>HT</td>
<td>Hash Table</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ICN</td>
<td>Information-Centric Networking</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISIS</td>
<td>Intermediate System to Intermediate System</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LAC</td>
<td>Local Area Code</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LDNS</td>
<td>Local Domain Name System</td>
</tr>
<tr>
<td>LL</td>
<td>Link Layer</td>
</tr>
<tr>
<td>LPM</td>
<td>Longest Prefix Match</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution</td>
</tr>
<tr>
<td>MCS</td>
<td>Module and Coding Scheme</td>
</tr>
<tr>
<td>MDHT</td>
<td>Multilevel Distributed Hash Table</td>
</tr>
<tr>
<td>MOS</td>
<td>Mean Opinion Score</td>
</tr>
<tr>
<td>NDN</td>
<td>Named Data Networking</td>
</tr>
<tr>
<td>NBRP</td>
<td>Name-Based Routing Protocol</td>
</tr>
<tr>
<td>NFC</td>
<td>Near-Field Communication</td>
</tr>
<tr>
<td>NRS</td>
<td>Name Resolution System</td>
</tr>
<tr>
<td>NS3</td>
<td>Network Simulator 3</td>
</tr>
<tr>
<td>OSN</td>
<td>Online Social Networks</td>
</tr>
<tr>
<td>OSPF</td>
<td>Open Shortest Path First</td>
</tr>
<tr>
<td>OTT</td>
<td>Over-the-top</td>
</tr>
<tr>
<td>PIT</td>
<td>Pending Interest Table</td>
</tr>
<tr>
<td>PMF</td>
<td>Probability Mass Function</td>
</tr>
<tr>
<td>POP</td>
<td>Point of Presence</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>QR</td>
<td>Quick Response</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PHP</td>
<td>PHP: Hypertext Preprocessor</td>
</tr>
<tr>
<td>PIT</td>
<td>Pending Interest Table</td>
</tr>
<tr>
<td>PMF</td>
<td>Probability Mass Function</td>
</tr>
<tr>
<td>RAMP</td>
<td>Random Access Minimal Parser</td>
</tr>
<tr>
<td>RRC</td>
<td>Radio Resource Control</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indicator</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-Time Transport Protocol</td>
</tr>
<tr>
<td>RTSP</td>
<td>Real-Time Streaming Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SAL</td>
<td>Service Access Layer</td>
</tr>
<tr>
<td>SDP</td>
<td>Service Discovery Protocol</td>
</tr>
<tr>
<td>SGW</td>
<td>Signalling Gateway</td>
</tr>
<tr>
<td>SID</td>
<td>Service Identification</td>
</tr>
<tr>
<td>SINR</td>
<td>Signal to Interference plus Noise Ratio</td>
</tr>
<tr>
<td>SNEW</td>
<td>Social Network Web</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TCP</td>
<td>Transfer Control Protocol</td>
</tr>
<tr>
<td>TTI</td>
<td>Transmission Time Interval</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>U2U</td>
<td>User to User</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UPnP</td>
<td>Universal Plug and Play</td>
</tr>
<tr>
<td>URI</td>
<td>Universal Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Universal Resource Locator</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WiFi</td>
<td>Wireless Fidelity</td>
</tr>
</tbody>
</table>