Deliverable D4.3

Final Implementation and Software Release of the Technical Solutions on Content Placement and Delivery

Public deliverable, Version 1.0, 15 December 2014

Authors
- FT Ali Gouta, Yannick Le Louédec
- A-LBELL Danny De Vleeschauwer (editor), Chris Hawinkel
- IMDEA Nicola Bui, Joerg Widmer, Foivos Michelinakis
- TSP Ángel Cuevas, Reza Farahbakhsh, Noel Crespi
- ALUD
- TUD Christian Koch, Stefan Wilk, Julius Rückert, Timo Thräm, David Hausheer, Wolfgang Effelsberg
- TI Fabio Mondin
- UCAM
- UC3M Ruben Cuevas, Juan Miguel Carrascosa

Reviewers Nicola Bui, Joerg Widmer, Chris Hawinkel

Abstract
The present deliverable D4.3 is the third and final deliverable of WP4. This document first briefly recaps the status of the work reported in deliverables D4.1 and D4.2 and then concentrates on new results obtained up to the end of WP4. The new contributions fall in four categories. In the first part some new results are provided for small TELCO caches, illustrating that the optimal caching strategies depend on the type of service that are offered: in particular, it is shown that the caching strategy for Catch-Up Television differs from the one for Video-On-Demand. In the second part, new developments for pre-fetching are presented. Specifically, the final architecture of the mobile social pre-fetcher is described and some of its components are studied in more detail. Realistic traces (for Facebook and YouTube) are obtained with an app that implements a subset of the components of the pre-fetcher and via measurements in a real mobile network. Based on this real data, usage patterns are identified and relevant social information is extracted. Moreover, the network resource allocation algorithm is further detailed and its performance is compared to the optimal and a naive algorithm. Also, the performance of the pre-fetcher is studied, with the Prefsim simulator described in deliverable D4.2, showing that social information can yield a gain in terms of hit ratio and overhead for content shared on Facebook. Thirdly, the P2P-social-assisted content delivery use case is further worked out. In particular, some building blocks were further developed to make this use

© Copyright 2014 eCOUSIN Consortium

Project funded by the European Union under the Information and Communication Technologies FP7 Cooperation Programme
Grant Agreement number 318398
case more user-friendly and secure. Finally, the social-enhanced content placement with social information use case is further explored (in which dissemination of content announced via Twitter is studied). Specifically, a more thorough performance analysis is presented and it is shown that in this case the performance of caching algorithms using social information can double the performance in terms of hit ratio. The final conclusion of the studies in WP4 is that the gain social information can yield in content dissemination (e.g., caching and pre-fetching) components depends on the use case studied: for use cases where the social information strongly correlates with content consumption and is explicitly available (e.g., Twitter, Facebook) the gain can be considerable, while for cases where the social information is less pronounced or needs to be extracted from user behaviour (e.g., Video-On-Demand, Catch-UP Television) it is much harder to attain a gain. The results obtained in WP4 are being transferred to WP6 where they will be used to tune the software components of the demonstrators.
EXECUTIVE SUMMARY

This deliverable completes the work committed in WP4. Starting from the work described in deliverable D4.1 [D4.1] and D4.2 [D4.2] we present extensions and new elements that show that the goals of the work of WP4 have been achieved. The contributions of the present deliverable D4.3 can be divided into four main parts.

First, the case of small TELCO (Telecommunication Company) caches is studied. The (small) gains obtained in the study of VOD (Video On Demand) (reported in deliverable D4.2 [D4.2]) are hard to reach in case of CUTV (Catch-Up Television). This illustrates that the gain that can be obtained with social information depends on the considered use case. In general, it turns out that, while in cases where the social information is explicitly available and strongly correlated with content consumption (see Facebook and Twitter below) a considerable performance gain can be attained, while in cases where the (implicit) social information needs to be extracted from usage patterns (and probably does not correlate that well with content consumption) (e.g., VOD and CUTV), the performance gain can be quite small.

Second, the components of the mobile pre-fetcher, associated with the Social-Assisted Time-Unconstrained Content Delivery use case, are detailed further. The final architecture of the mobile social pre-fetcher is described and some of its components are studied in more detail. To study the performance two (additional) data sets were gathered: a small one obtained via a mobile tracing app installed on mobiles of consenting users and a large data set obtained in large network of a European mobile operator (already announced in deliverable D4.2 [D4.2]). It is argued that for Facebook a large gain can be obtained in terms of hit ratio (i.e., correctly predicting what a user will consume) and overhead (i.e., avoiding that too many wrong predictions are made). Furthermore, the performance of the network resource allocation algorithm, introduced in deliverable D4.2 [D4.2] and described in more detail here, is assessed. It is shown that its performance is very close to the theoretically optimal algorithm (that has complete knowledge of the future) and considerably outperforms a naive one that just tries to keep the video play-out buffer as full as possible.

Third, the P2P-Social-assisted Content Delivery use case (for Facebook) was enhanced with more features. In particular it has been made more user-friendly. It relies on local storage from the client in the present version and on VLC in its server mode to stream from that local storage, while in the previous version Dropbox was used as "local" storage. It now implements a method to locally store shared YouTube videos. Also security features were added. Note that while this use case was not amongst the six originally described in deliverable D2.1 [D2.1], its main purpose is to illustrate that the architecture developed in deliverables D2.2 [D2.2] and D2.4 [D2.4] is generic and can support more use cases than the six originally envisioned.

Finally, the Social-enhanced Content Placement with Social Information use case is further developed. Caching algorithms for content announced in tweets, that make use of social information, in particular, the number of followers the producer of a tweet has, are compared with traditional caching algorithms. It is shown that the former can outperform the latter by a factor two in hit ratio if the parameters are tuned correctly.

The general conclusions of the studies performed in WP4 are that in specific cases using social information in the content dissemination process (e.g., caching and pre-fetching) is highly beneficial, while in other cases the gain is marginal. Specifically, in case the social network is strongly correlated with content consumption and is explicitly known (like in Twitter and Facebook) the gain can be considerable if the algorithm is tuned accurately, while in applications where this social information needs to be extracted and is the content consumption is more driven by global popularity rather
than social relations (like in VOD and CUTV), the gains turn out to be marginal. These results will be used in WP6 to tune the parameters of the use cases that will be part of the demonstrations.
# TABLE OF CONTENTS

EXECUTIVE SUMMARY.................................................................................................................. 3

TABLE OF CONTENTS .................................................................................................................... 5

INTRODUCTION .................................................................................................................................. 7

## 1. SUMMARY OF DELIVERABLES D4.1 AND D4.2.................................................................. 8

1.1 Content dissemination layer in the WP2 architecture ................................................................. 8

1.2 Building blocks discussed in D4.1 and D4.2 ............................................................................. 8

1.2.1 Caching using social information ........................................................................................... 8

1.2.2 Pre-fetching ............................................................................................................................ 8

1.2.2.1 Architecture ...................................................................................................................... 8

1.2.2.2 Network resource allocation optimization ......................................................................... 8

1.2.2.3 Prediction of content consumption .................................................................................. 9

1.2.3 P2P-Social-Assisted Content Delivery .................................................................................... 9

1.2.4 Enhanced Content Placement Using Social and Coarse-Grain Location Information .......... 9

1.2.5 Exploiting Social Relationship and Personal Background for Content Discovery in P2P Networks. . 10

1.2.6 Content Lookup in the Personal Sharing Clouds .................................................................... 11

## 2. REALISTIC TRACES ............................................................................................................... 12

2.1 Dataset ...................................................................................................................................... 12

## 3. DESIGN AND PERFORMANCE OF BUILDING BLOCKS .................................................. 13

3.1 Small caches .............................................................................................................................. 13

3.2 Mobile Pre-fetching Components ............................................................................................ 16

3.2.1 Architecture of the mobile social pre-fetcher ......................................................................... 16

3.2.1.1 Social Data Manager ......................................................................................................... 17

3.2.1.2 Content Pre-fetcher .......................................................................................................... 17

3.2.1.3 Network Analyser ........................................................................................................... 17

3.2.1.4 Illustrative Example ........................................................................................................ 17

3.2.2 Optimizing Mobile Pre-fetching by Leveraging Usage Patterns and Social Information ....... 18

3.2.2.1 Introduction .................................................................................................................... 18

3.2.2.2 Related Work .................................................................................................................. 19

3.2.2.3 Approach ......................................................................................................................... 19

3.2.2.4 Results and Discussion .................................................................................................. 21

3.2.2.5 Summary and Outlook .................................................................................................... 22

3.2.3 Mobile Tracing App ............................................................................................................... 22

3.2.3.1 Introduction .................................................................................................................... 22

3.2.3.2 Background and related work ......................................................................................... 23

3.2.3.3 Mobile App System Design ........................................................................................... 24

3.2.3.4 Description of a Dataset for Pre-fetching ....................................................................... 25

3.2.3.5 Analysis of the Mobile Facebook Dataset ...................................................................... 26

3.2.3.6 Discussion and Conclusion ............................................................................................ 33

3.2.4 Network resource allocation algorithm ................................................................................ 34

3.2.4.1 System model ................................................................................................................. 34

3.2.4.2 Resource allocation optimization with perfect forecast ................................................... 35

3.2.4.3 General forecast model ................................................................................................. 36

3.2.4.4 Resource allocation optimization under uncertainties ..................................................... 36

3.2.4.5 Results ........................................................................................................................... 40

3.3 P2P-Social-Assisted Content Delivery .................................................................................... 42

3.3.1 Design a user friendly interface ............................................................................................ 43

3.3.2 Keeping local videos in a real local server instead of dropbox .......................................... 45

3.3.3 Adding the capability to download shared YouTube videos and add to the local server ....... 46

3.3.4 Improve security features of the application and comply the Facebook privacy levels for each video and user .................................................................................................................................. 46
3.3.5 Efficient capacity management of local server ................................................................. 46
3.4 Social-enhanced Content Placement with Social Information ............................................. 47

4. MAPPING OF THE SECTION ON THE ARCHITECTURE AND USE CASES ......................... 49

5. CONCLUSIONS .................................................................................................................. 51

REFERENCES ..................................................................................................................... 53

ACRONYMS .......................................................................................................................... 57
INTRODUCTION

The work contained in the present deliverable D4.3 establishes the final steps towards the final implementation of technical solutions for Content Dissemination Layer of the eCOUSIN architecture described in deliverable D2.2 [D2.2], which supports the use cases described in deliverable D2.1 [D2.1]. It consists of five sections.

The first section summarises the status of the content dissemination components described in previous deliverables D4.1 [D4.1] and D4.2 [D4.2] of WP4. Two components, i.e., Exploiting Social Relationship and Personal Background for Content Discovery in P2P Networks and Content Lookup for Personal Sharing Clouds, required only minor updates, while four other components are described in more detail in Section 3.

Section 2 presents the statistical properties of a Facebook and YouTube data sets, already announced in deliverable D4.2 [D4.2]. Similarly to what was already performed for other data sets in that deliverable, the present deliverable provides those statistics of that Facebook and YouTube data set that are relevant for understanding the performance of content dissemination modules (in particular, the pre-fetching module).

As stated before, in Section 3, which forms the major part of the present deliverable, four use cases are explored in more detail. The caching algorithms for small TELCO (Telecommunications Company) caches, studied in Section 3.1 illustrate that the beneficial effect of social information on the performance depends very much on the use case. While in previous deliverables it was shown that for (VOD) Video-On-Demand there was a small gain in hit ratio, for CUTV (Catch-Up Television) a similar gain is harder to reach. This section shows that this is due to the nature of the data set. In further subsections it is shown that the gain for other types of services (i.e., Facebook and Twitter) can even be larger than for VOD. Section 3.2 deals with the pre-fetching studies. The final architecture of the pre-fetcher is provided. Data is acquired with an app implementing a subset of the components. This data and the data described in Section 2 is statistically analysed to produce guidelines for setting the parameters in the pre-fetcher. The performance of the network resource allocation algorithm is assessed and compared to the ideal algorithm (which has knowledge of the future) and the naive algorithm that tries to keep the play-out buffer full. Finally, the Prefsim simulator described in deliverable D4.2 [D4.2] is used to refine the overall performance analyses of pre-fetching. Section 3.3 describes new developments for the P2P-Social-Assisted Content Delivery use case. User-friendliness and security are improved. True local storage is now used in the present solution, rather than using Dropbox for that purpose in the version described in deliverable D4.2 [D4.2]. For that purposes a mechanism to stream content from that local storage to a requesting user had to be implemented. Finally, support for YouTube links shared via Facebook was also foreseen. Section 3.4 describes further developments of the Social-enhanced Content Placement with Social Information use case. This use case considers caching algorithms for content items shared via Twitter. It is shown that the caching algorithms that make use of social information (i.e., how many followers the producer of a tweet) considerably outperform traditional caching algorithms.

Section 4 relates the topics described in the present deliverable with the use cases of deliverable D2.1 [D2.1] and the architectural components in the Content Dissemination Layer of the eCOUSIN architecture finalised in deliverable D2.4 [D2.4].

Section 5 draws the main conclusion of the work described in the present deliverable and of WP4 as a whole.
1. SUMMARY OF DELIVERABLES D4.1 AND D4.2

1.1 Content dissemination layer in the WP2 architecture

In deliverable D2.2 [D2.2] the eCOUSIN architecture is detailed, with one of the layers being the “Content Dissemination Layer”. Broadly speaking this layer takes past content consumption and explicit and implicit social relations (e.g., in an OSN) and network status extracted from the social and network layer respectively, as input and gives advice on where to place, store or cache content items. The study of some of the building blocks in this layer was started in deliverable D4.1 [D4.1] where we also provided an initial performance analysis of them. In deliverable D4.2 [D4.2] we provided more details for these building blocks and presented a further performance analysis of them. In the present deliverable D4.3, we describe the final version of the components together with their completed performance analysis. The final results are being transferred to WP6 where implementations of these building blocks are integrated in the demonstrators.

1.2 Building blocks discussed in D4.1 and D4.2

1.2.1 Caching using social information

In deliverable D3.1 [D3.1] we described a method to identify leading users and in deliverable D4.1 [D4.1] we introduced a method to exploit these social relations between users and user groups. This idea is further explored in deliverable D4.2 [D4.2], where a modified version of the LRU (Least Recently Used) caching strategy is adapted so that it can take the fact that there are lead users and followers into account. Further results on a new data set are described in Section 3.1.

1.2.2 Pre-fetching

The pre-fetching concept was introduced in deliverable D4.1 [D4.1]. It described how historical content consumption and social links between users can be used to predict upfront which user is going to consume which content item in the near future. This knowledge together with the knowledge of the status of the network is used to transport relevant content items closer (even on the user device) before the user requests it. Additionally in deliverable D4.1 [D4.1] we discussed related work and identified potential social properties of content items shared over OSNs that could be used for content prediction and, thus base pre-fetching decisions on them.

1.2.2.1 Architecture

Furthermore, the basic architecture of the mobile pre-fetching application was presented. It consisted of two main building blocks, the Pre-fetching App and the Crawler. These building blocks were further studied in deliverable D4.2 [D4.2], which provided the blueprint of the Content Prefetcher and the Social Data Collector and Manager components. In Section 3.2 of the present deliverable the final architecture and its components will be detailed.

1.2.2.2 Network resource allocation optimization

In deliverable D4.2 [D4.2], we showed the benefits that could be obtained by means of a proper resource allocation when the knowledge of the future evolution of the bandwidth availability is perfect. In addition, we explored the impact of prediction errors on the basic algorithm. In Section 3.2.4 of the present deliverable, we will relax the need for perfect knowledge by accounting for standard prediction techniques for both short and medium/long term scales.
1.2.2.3 Prediction of content consumption

To study how well content can be predicted based on past content we have developed a simulator, referred to as Prefsim, which is described in detail in deliverable D4.2 [D4.2]. With this simulator we assess the initial performance of various prediction algorithms in that deliverable. Section Erreur ! Source du renvoi introuvable. will investigate the performance in more detail.

1.2.3 P2P-Social-Assisted Content Delivery

In deliverable D4.2 [D4.2] we introduced the general idea of P2P-social-assisted content delivery (where P2P stands for “Peer-to-Peer”). The idea of this solution is to locate videos which are generated/shared by a user U through an OSN (Online Social Network) in two locations Local and CDN (Content Distribution Network): (i) U’s local premises with a 24/7 connectivity like U’s set-top box in her home as a local copy of the video, and (ii) in the standard location where the video is located with current solutions (e.g., CDN node, centralized server, etc.) as the traditional CDN location. This will provide U’s social friends the ability to retrieve the video from two locations, i.e., locally or from a CDN node. Both locations can stream the videos. The video selection will be based on U’s friend physical location. The Social-peer-assisted content delivery will help to offload a considerable portion of user-generated video traffic by means of applying a P2P approach and also reduce the load on social network CDNs as well as the overall traffic load.

In addition we presented a first draft version of the Facebook application which will be a practical demonstrator of the idea. The application named SocialiVideo is implemented to demonstrate the use of the proposed solution in the context of the most popular OSN, i.e., Facebook. The application is able to collect the location information of U and U’ Friends from two available sources:

i) physical location information (IP address of the users)
ii) Social location information from location attributes in the Facebook profiles (current city, hometown).

The application will collect the mentioned location information for all users that are using the application and store them in its database. Later for each video request the application first checks the location of both parties (video location and viewer location) and if they are in the same location zone (can be considered same city or same country) the application will stream the video from the local link, otherwise the stream will come from the Facebook link.

This simple but efficient way of content delivery mechanism can reduce the load of the social network CDNs as well as the overall traffic load. In Section 3.3 we look at the application in more detail.

1.2.4 Enhanced Content Placement Using Social and Coarse-Grain Location Information

In deliverable D4.1 [D4.1] the optimal location of data centres (for content associated with tweets) was determined based on a Twitter data set. Deliverable D4.2 [D4.2] conducted a preliminary investigation on which caching algorithm should govern the caches of these data centres evaluating performance metrics as HR (Hit Ratio), distribution and access cost (in terms of traffic volumes). These algorithms have been refined, implemented and tested in an emulation environment as described in Section 3.4.
1.2.5 Exploiting Social Relationship and Personal Background for Content Discovery in P2P Networks

In deliverable D4.1 [D4.1] we presented a thorough demonstration on how social information can help in improving the content look-up in an unstructured P2P network. Our concept relies on the fact that if an unstructured P2P network based on the actual social relationships declared in real OSNs is constructed, the content of interest can be found using fewer queries, thus reducing the overhead, and finally improving the HR, meaning that end users increase a lot the probability of finding the objects they are searching.

In deliverable D4.1 [D4.1] we presented a formal introduction of the problem and thorough experiments that demonstrates that our proposal improves standard algorithms available in the literature.

We have extended the work introducing a new element related to our algorithm. We remind that our algorithm ranks the friends (i.e., peers of a user in the P2P network) based on the probability that they may resolve successfully the query of the root user. Hence, we wanted to evaluate whether our ranking mechanism is actually correct. This means, whether the users ranked in first position actually are more efficient on resolving the queries of the root user.

Therefore, with the ranked friends list generated by social-DRWR-P2P (with DRWR standing for “Distributed Random Walks with Restart”, see [D4.1]), we cluster each 10 successive friends from top to bottom into a group and compare the average number of hits of each group. That is to say, the top 10 friends are clustered into group 1, and the next successive 10 friends (i.e., top 11th to top 20th) into group 2. In this way, we generate 20 ordered groups from the top 200 friends. Note that the friends in n-th group have higher scores than friends in (n + 1)-th group. Figure 1 shows that, both for personal interests searching and popular interests searching, the friends with more knowledge and higher similarity (i.e., the ones in higher positions in the rank list) can achieve better performance. In this figure “r=1” means that only direct friends are consulted, while “r=2” means that also friends of friends are consulted (see deliverable D4.1 [D4.1]).

![Figure 1: Hits number of social-DRWR-P2P](image)

Therefore, these results extend the ones obtained in deliverable D4.1 [D4.1] and reinforce the efficiency and enhancement achieved by the proposed solution.
1.2.6 Content Lookup in the Personal Sharing Clouds

The module initially called “Resource Locator”, described in deliverable D4.1 [D4.1] was mapped to the eCOUSIN architecture (see [D2.2]) as “Content Lookup”.

Before going in depth into this module’s details we should briefly recall how the associated use case works. The “Personal Sharing Clouds” use case (see deliverable D2.1 [D2.1]) aims at creating a direct link among different media centres, so that remote resources can be seen as local resources by leveraging on local distribution protocols such as UPnP (Universal Plug and Play) or DLNA (Digital Living Network Alliance). The meeting points as well as the information exchange points are OSNs. Figure 2 shows the use case architecture mapped on eCOUSIN architecture [D2.4].

Figure 2: Personal sharing cloud use case mapped on the eCOUSIN architecture.

Except for the federated social network integration, this was the architecture which was prototyped and shown in the Turin Intermediate Review meeting.

The Content Lookup module is responsible for periodically discovering the files shared on the remote nodes and present them in the local area network where the media centre sits, in order to have them available on the final user interface, enabling easiness of implementation and alignment with the rest of the architecture.

In this specific case resources are actually remote (and local) files. In general we designed the module so that resources could be both:

I. sharing nodes’ services available on remote nodes, either performing service discovery requests for known services or directly requiring to remote nodes the list of registered services

II. content provided by a preselected small set of well-known services.
Even if we decided not to enable the generic service discovery in the use case to keep it more aligned with the overall eCOUSIN architecture, it will be possible to enable it in a very easy way in other scenarios.

The Content Lookup module strongly leverages on the naming scheme and on the modules which have been described in depth in WP5 Deliverables [D5.1, D5.2]. The naming scheme encapsulates a private resource indicator into a standard public IP (Internet Protocol) addressing, trying to “fool” the local discovery protocol (uPnP, DLNA...) making remote resources look as local at the media-centre side. This means that the local content lookup module does not filter out the remote “unshared” resources, which are instead, filtered by the remote access filter/discovery filter, both of which are described as well in WP5.

At the time of writing, we can successfully report a working implementation of the Content Lookup module into a first prototype, which has been demonstrated at the intermediate review meeting in Turin.

The next development phase will be the extension of the use case with the Federated Social Network part in WP6. This two-fold extension will include some code to be run on the web server and some modification to the Social Data Collector described in WP3 [D3.1]:

- Exploit “Opensocial” (http://opensocial.org/) to publish data (for example via Mobile) on the media centre: In this case the published files located in specific folders in the media centre will automatically become accessible as local resources from remote endpoints according to the normal behaviour of the use case.
- Exploit “Opensocial” to obtain information on how to connect to my friends’ remote media centre reading this information into a specific field into the profile data exposed to the user’s friend, according to the Opensocial specifications.

In both cases the integration needs an assessment plan for the module in order to check that resources are correctly located locally and remotely, with specific focus on the refresh procedure, since the process of notification for the extension will be different from the actual.

The integration with federated social network and the retesting of the environment will be the main topic from here on in this work package.

2. REALISTIC TRACES

In deliverable D4.2 [D4.2] we already described some statistical properties of data sets used in the eCOUSIN project in the details that are relevant for understanding results produced in WP4 (other statistical properties are discussed in WP3 deliverables). In this section, we discuss the overall properties of an additional dataset (already introduced in [D4.2]). We provide a more thorough analysis and findings on users’ behavior in the restricted version of the present deliverable.

2.1 Dataset

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Number of unique users</th>
<th>Number of unique videos</th>
<th>Number of requests</th>
<th>Service provider</th>
<th>Typical URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>3,179,206</td>
<td>10,076,156</td>
<td>64,722,755</td>
<td>Google CDN</td>
<td>r8–sn–lg97kue6.youtube.com/video/playback?id=...</td>
</tr>
<tr>
<td>Facebook</td>
<td>399,645</td>
<td>2,856,321</td>
<td>14,305,404</td>
<td>Akamai</td>
<td>16083277.750958311600430.982950231_n.mp4?...</td>
</tr>
</tbody>
</table>

Table 1: Properties of the two used datasets.

We rely on a large dataset gathered at all Gi interfaces of all GGSNs (Gateway GPRS (General Packet Radio Service) Support Nodes) deployed by a mobile carrier in France. The dataset consists of logs of video streaming sessions generated by all connected devices of the carrier’s subscribers. The logs
were collected from 8 January 2014 to 28 April 2014. Due to maintenance reasons, the monitoring infrastructure was disabled for 27 days, which makes the real period of data collection lasting about 94 days. These disruptions do not introduce any bias in the data analysis and simulations reported in this deliverable since they are not achieved over the whole duration of the dataset but over short time frames during which the collection was not interrupted. We limit our study to two subsets of video traffic: requests for YouTube videos and requests for Facebook videos. Table 1 gives an overview of both these subsets of traffic. Parsing the HTTP (HyperText Transport Protocol) header in Facebook and YouTube traffic flows enables to extract the unique video identifier (referred to as reference ID (Identifier) requested by the users. For illustrative purposes typical URLs (Uniform Resource Identifiers) from YouTube and Facebook are given in Table 1, the field in bold pointing to this reference ID of the video. To preserve confidentiality and privacy, our dataset is anonymized during an early stage in the collection process.

In the restricted version of the present deliverable we present a more detailed analysis.

3. DESIGN AND PERFORMANCE OF BUILDING BLOCKS

This section describes various components of the “Content Dissemination Layer” that will be either further used in the demonstrators of WP6 or of which the performance is further explored in this deliverable.

3.1 Small caches

In deliverable D3.1 [D3.1] the concept of lead users was introduced, referring to the subset of users of a video service that are early consumers of new video content items, which are likely to be consumed many more times in the near future. The idea behind it was that knowledge of lead users (if any) can be exploited for more efficient caching and recommendation systems. Lead users were identified for two distinct datasets, a VOD (Video-on-Demand) dataset and a CUTV (Catch-Up Television) dataset; different mechanisms were used for this, due to the rather different nature of the two datasets (see [D3.1]).

Deliverable D4.2 [D4.2] reported on the effect of exploiting the combination of knowledge about lead users with a variant of the MLRU (Modified LRU) caching algorithm, for the case of the VOD dataset. It was concluded at that time that improvement was limited if only information from the VOD dataset itself was used.

In the current deliverable we report on the results of a more in-depth analysis of the CUTV dataset, and on the impact of this on the possibility to apply the Lead User idea to this dataset. Whereas for the VOD case MLRU performed better than LRU, and dMLRU (Differentiated MLRU) led to a further small improvement (see [D4.2], Figure 39), this proved not to be the case for the CUTV dataset, since in all investigated cases LRU always performs as least as good as MLRU. In an attempt to understand the reasons for this, a closer investigation of the CUTV dataset was done, which led to a number of observations, which are detailed in the following.

First we looked at the relation between views on consecutive days. On one hand we see a rather strong correlation between videos watched on consecutive days – see Figure 3.
On the other hand, there is only a weak relation between the users that are active on consecutive days, as is shown in Figure 4.

Secondly, we can clearly identify different classes of content (see Figure 5), in terms of the nature of their popularity: viewed all the time – viewed for just one week – viewed sporadically (but then multiple times/day). This is in fact as can be expected in a CUTV environment, with weekly new episodes.

A third observation is that, despite the overall diurnal pattern, the viewing patterns rather strongly differ from day to day, as illustrated in Figure 6: the peaks for total views and different videos do not always coincide and vary significantly from one day to the other. Note that, from a network perspective upstream of a cache, it is the peak in different videos which causes peaks in network traffic.

In order to assess the influence of these strong variations, the CUTV dataset was split in a number of smaller datasets, each covering exactly one day. LRU, MLRU and dMLRU were then compared for a number of these smaller datasets. The result is shown in Figure 7 for LRU and MLRU, for two

Figure 3: Views on consecutive days.

Figure 4: Users active on consecutive days.

Figure 5: Different popularity patterns.

Figure 6: Viewing patterns on different days.
different days: 19/03 and 21/03. For 19/03, LRU performed slightly better than MLRU for all investigated Jump values; for this reason dMLRU was not further considered. For 21/03, MLRU performed slightly better than LRU, but there was no significant further improvement by dMLRU; hence dMLRU is not shown on the graph. The strong variations from day to day are reflected in the LRU performance: note that for 19/03 the LRU HR is, for each of the considered cache sizes, significantly higher than in the 21/03 case. This suggests that there is much less room for improvement, as indeed illustrated by the weaker MLRU performance.

![Figure 7: Comparison of LRU, MLRU on different days.](image)

Although the picture is different from one day to the other, it is clear that the effect of using MLRU/dMLRU instead of LRU is minimal. Hence the conclusion is that for this CUTV dataset, the lead user approach, based on information from the dataset only, does not bring any significant benefits for improving caching performance.

The next step was to investigate whether other dataset characteristics, related to viewing behaviour and to the characteristics of the CUTV service itself, could be exploited. Earlier in this section, the rather strong relation between videos watched on consecutive days was discussed (ref Figure 3). So it can be assumed that ‘yesterday’s’ most popular movies will again be popular ‘today’. Also, in a CUTV context, it should be possible to predict reasonably well which planned videos are likely to be popular for some time, from the moment they are released in the system. Both these sets of videos (or the most relevant subsets of them) could be preloaded in the cache ahead of the peak hour. In a similar way, for a number of videos it is reasonably possible to predict that they will be requested very sporadically, or probably just once. For such videos, one may consider not to cache them when they are requested, in favour of other videos that may be requested (again) soon.

Figure 8 shows how the number of cache misses can be reduced when exploiting these elements during the caching decision process, for the same 2 days as in the first part of this section. The number of cache misses is chosen here, as they directly impact the traffic volume upstream of the cache. Both graphs compare the performance of LRU with the optimized approach (labelled “optim”), in which yesterday’s popularity, predicted popularity and predicted unpopularity are exploited. The figure shows that the reduction of cache misses at peak hour ranges from 6% (21/03) to 14% (19/03).
The conclusion from this study of the CUTV data set is that the caching performance can be improved, if the caching decision is based on a combination of information from the past with information based on prediction. In the cases presented in Figure 8, this prediction is based on specific characteristics of the CUTV service. In other environments (represented by other datasets) prediction information could be based on OSN data.

3.2 Mobile Pre-fetching Components

3.2.1 Architecture of the mobile social pre-fetcher

Figure 9 shows the architecture of the mobile social pre-fetcher. The components are first briefly discussed in the following subsections and some of them in more detail in later sections.
3.2.1.1 Social Data Manager

The Social Data Collector and Manager continuously monitors the user’s OSNs. For every OSN used, a special crawling module is used. This key module of the demonstrated software accesses the user’s OSN feeds to request information about video posts. Additionally, the on-going interactions of the user with his friends and his content-specific interactions are monitored. The Social Data Collector hands the information acquired by these modules to the Data Aggregator Module. Metadata such as timestamps and dates are stored in a uniform manner in a local database by the Social Tracer. To determine which items should be pre-fetched first the Social Predictor stores the candidates with a priority in the local Pre-fetching Candidates database, which is used by the Content Pre-fetcher. Prioritizing different videos is a topic of on-going research. For the demo, the priority is derived by a simple metric based on the video’s Facebook like count. Currently, the application supports Facebook, but will be extended with further OSNs in the future. In the scope of eCOUSIN, it is planned to extend the app to support YouTube as a further OSN.

3.2.1.2 Content Pre-fetcher

The Content Pre-fetcher includes the necessary functionalities to download the content. The Decoder service translates the OSN URL (Uniform Resource Locator) to the URL of the corresponding video file and passes it to the Download Client. The Download Client performs the content download, takes care of connection interruptions, and selects a video resolution appropriate for the smartphone’s display if multiple resolutions are available. The Download Scheduler allows download scheduling for the pre-fetching candidates. This enables postponing the download until certain conditions are met, e.g., until a WiFi (Wireless Fidelity) network is available.

3.2.1.3 Network Analyser

To be able to feed the Bandwidth Optimization module with predictions and their confidence, a bandwidth prediction module is used. This module considers statistical information about user mobility and the available bandwidth in a given mobile network cell. In order to combine all the statistical information, different predictors will be jointly used. In particular, we use different solutions for the short (e.g., tens of seconds, a few minutes) and the medium-long (e.g., tens of minutes, hours) term predictions.

The short term prediction is achieved by means of a simple ARMA (AutoRegressive-Moving Average) filter [BUI14]. The algorithm for applying the filter and its coefficients have been previously tuned according to user’s past information. Depending on the user’s movement speed, the prediction validity varies between a few ten of seconds (fast movements) up to some minutes (quasi-static scenarios). During the validity time, the ARMA filter predicts the future mobile throughput.

The medium-long term prediction is performed by statistical models [BUI14B]. These models account for the degradation on the accuracy of both the user position and the network cell congestion while the prediction is made in the future.

By combining the two prediction techniques this component is able to decide when it is best to pre-fetch a content item and, if the content has to be streamed from remote, what is the best way to allocate resources in order to optimize the cost.

3.2.1.4 Illustrative Example

In Figure 10, on the left, circles represent video events on the time line of a few users. These events are color coded: orange stands for publish time, green for pre-fetch time and blue for watch time. The area behind the dots is colored yellow or green if the mobile has mobile or WiFi access respectively. On the right video 5 is streamed from remote, since it had not been pre-fetched in
advance. The figure illustrates the bandwidth usage policies depending on the measured and forecast signal qualities.

![Figure 10: A simple example of pre-fetching (left) and resource optimization (right).](image)

### 3.2.2 Optimizing Mobile Pre-fetching by Leveraging Usage Patterns and Social Information

Mobile pre-fetching has to cope with high precision in the used prediction mechanisms due to limited resources on the smartphone. An approach which gives more flexibility and enables more efficient pre-fetching, is partial pre-fetching. Here, only the most promising parts of the video, which are likely being watched on a mobile, are pre-fetched. Partial pre-fetching can increase the QoE (Quality of Experience) perceived by users significantly since it allows a smooth playback without interruptions. How much of which content should be pre-fetched? In the following, investigations to answer this research questions have been performed [KH14].

#### 3.2.2.1 Introduction

Real-time entertainment content represents the largest single source of mobile data traffic in North America and Europe [SAN14]. The growing number of videos watched mobile leads to a heavy load on the networks, especially during peak-hours. In North America, real-time entertainment traffic accounts for one third of the mobile traffic during peak-hours, followed by traffic caused by OSNs and content portals. The most popular platform for UGC (User Generated Content) is YouTube. Six billion hours of videos are watched each month on YouTube. Currently, 40% of the contents are watched mobile. Besides YouTube, which is likely to remain the single largest source of UGC traffic, further applications such as Vine are expected to contribute more to mobile traffic in the near future.

It is expected that the per-month data volume will increase tenfold whereas the mobile bandwidth capacity will increase only twofold by the year 2018 [CIS14]. As a result, mobile operators face new challenges since the frequency spectrum is limited, while the data volume growth is not. Despite the increasing adoption of LTE (Long Term Evolution), the gap between supply and demand on mobile network capacities keeps increasing [SOL13]. Mobile carriers react to this by adding caps to LTE data plans. To this end, a new social- and context-aware approach to relieve overloaded mobile networks at peak-hours is proposed in this section. The approach leverages pre-fetching mechanisms to offload videos from the mobile network, mainly over WiFi. Less peak-hour traffic leads to reduced costs for mobile operators, since capacity over-provisioning can be avoided. Furthermore, the mobile network frequency spectrum is used less intensively. The proposed approach differs from existing work in this area by considering both, the network operator and the user needs, as well as using specific user-centric social information. This information is derived from the user’s OSN profiles (e.g., YouTube, Facebook or Vine), from the user’s watching behaviour (e.g., partial/repeated watching, source and topic), and the used smartphone sensors (e.g., connectivity, location, time, and movement). There is a huge number of potentially interesting videos for a user on content portals, which makes content consumption hard to predict. Using OSN information has proven to be predictive for a user’s watching behaviour [ZHA13].

The main contributions of the proposed work here include
(i) A mechanism which predicts the probability of videos being watched by a user. The prediction includes the portion of a video being watched and the number of times a user will watch it.

(ii) The user’s location and time are used to determine the optimal pre-fetching sequence.

(iii) The best time to pre-fetch is determined based on the time left until a user is likely to watch a video and based on a location-connectivity model. Initial results on the analysis of users’ preference patterns and partial video views show a high potential for category-based pre-fetching.

The remainder of this section is structured as follows. Section 3.2.2.2 discusses existing work. Section 3.2.2.3 describes the approach on mobile video offloading. Section 3.2.2.4 shows and discusses the results. Section 3.2.2.5 gives a summary and an outlook on possible further optimization.

### 3.2.2.2 Related Work

Content selection is the most important choice for pre-fetching systems. Zhao et al. [ZHA13] use machine learning to predict relevant videos from the user’s Facebook feed. They use post interactions, the number of private messages exchanged, the number of viewed videos from friends or pages, and the post’s Facebook “like count” to determine promising videos. To derive the user’s engagement, their own Facebook app needs to be used, which introduces a bias, since the post ordering and the look-and-feel of Facebook cannot be imitated. Our proposed approach works with the native Facebook view, while using an own video player and its focus is on the network operators. A key aspect in pre-fetching is how much shall be pre-fetched. An approach using a fixed chunk length of 1200 bytes is presented in [WSY11]. In [KZK11] the authors compare caching, pre-fetching, and a mixed approach implemented at a proxy or a client device. They are using prefix pre-fetching. Based on the available bandwidth, the video duration, and the video bit rate, the prefix size is determined. This approach increases the user experience, but does not respect the network operator needs. The authors of NetTube [CFL09] present a prefix-based pre-fetching mechanism. They aim to increase the user experience, which has shown to be sensitive to stalling events at the video playback start [KRI12]. Our approach determines the segment’s length with respect to the probability that it is watched. Therefore, the user perceives less stalling events. Energy saving is important for the user. The measurements conducted by [GRO13] show that transmissions over WiFi are about ten times more energy efficient compared to 3G (Third Generation (Mobile)). Thus, it is beneficial to download content when WiFi is available. In [MAN14], the behaviour of mobile streaming apps with iOS and Android in combination with the streaming players of YouTube, Netflix, and Hulu is compared. An interesting observation of the authors is that up to 15% of traffic in video streaming sessions is caused by redundant traffic. This is caused by video quality adaptation due to varying network conditions. The proposed approach circumvents this by offloading on stable network conditions in a fixed quality. Overall, our approach presented here differs, as it makes predictions based on the videos’ meta-data and the user’s engagement towards these aspects. Therefore, watching probabilities based on social information are leveraged in the approach.

### 3.2.2.3 Approach

Offloading videos from the mobile network requires accurate watching prediction, the estimation of how much is watched and a download scheduler. This problem is divided into multiple modules depicted in Figure 11, which is an extension of the concept stated in Figure 9 but with stronger emphasis on the machine learning models used for the predictions in this approach. As inputs, the Network Offloader uses information from the OSNs, the observed user interactions, and the network conditions. The proposed architecture consists of three monitoring modules, targeting the OSNs, the user interactions, or the environment. These modules provide input for the Pre-fetching Predictor. This module determines videos which are most relevant for a user. The module’s sub-modules
formulate, update, and leverage a model of the user involvement, the probability for a video being watched to a certain percentage, and the connectivity. The predicted videos are passed to the Prefetching Scheduler, which schedules the download based on time, location, and consumption patterns.

![Architecture of the Mobile Network Offloader.](image)

**OSN Monitor:** Next to popular content, content shared on the user’s OSN feed has turned out to be predictive as it is shared within a small social circle, e.g., family and friends, and therefore, is in many cases of high relevance for the user. This module provides a view on all contents that are presented to the user by OSNs. In OSNs, the publishing behaviour is assumed to be non-deterministic. Due to this, the module has to request for video updates regularly. Video-related data is stored in a local database.

**Interaction Monitor:** The Interaction Monitor senses interactions between the content and the user. Subscriptions, assignments, and friendships of a user are used as a baseline. Information like when, where, and for how long a user watched a video is stored in a local database. Compared to other works, our approach does not force a replacement of the user’s Facebook app. Instead, a video player which is able to collect detailed information about the video playbacks is used.

**Environment Monitor:** This module aims for the identification of user habits. Therefore, environmental data, e.g., location, time, and bandwidth are captured. The entries of the OSN Monitor’s database are enriched with this information.

**Pre-fetching Predictor:** Based on the monitoring module’s information, this module models the user’s key behavioural aspects. Not all aspects are equally predictive for different users. This module weights them in a user-specific manner. The Prefetching Predictor’s sub-modules build the core of this work and are described in the following.

**Involvement Model:** This module determines which video features are most predictive. Users are assumed to be interested in certain topics or in specific sources. A source is defined as the combination of OSN and publisher. Topics are identified by common words in the video name or description. Topics are especially interesting for predicting videos from unknown sources. This part of watched videos accounts for up to 44% of watched videos [GFK14]. Sources and topics cover interest-driven OSNs like Quora and source-driven OSNs like YouTube.

**Partial View Model:** Not all videos are watched completely [GIL08]. This module uses the portion watched as an engagement measure. Depending on the user’s context and the content’s meta-data, a prediction model is developed. This model is further used to decide how much shall be pre-fetched.
As shown by [CHE13], videos are watched repeatedly more often than others, depending on their categories. The proposed approach uses a cache management which leverages the user’s re-watch behaviour by keeping the amount of video which is pre-fetched longer or shorter.

**Location Model:** Most people follow a daily location pattern [DOG14]. This is assumed to apply for video consumption in many cases. The Location Model strives to model the user’s access patterns based on his location. At work, the user might be interested in significantly different topics, than in his leisure time. This information is important since the time between content recognition by the OSN Monitor and the user watching it can be quite short. Therefore, an optimal pre-fetching sequence should consider this information, especially when only a few videos can be pre-fetched timely.

**Pre-fetching Scheduler:** The time between the content publishing and its consumption allows planning the download for a certain time window. An energy-efficient scheduling uses WiFi whenever possible. Sometimes this is not possible in the given time window. In this case, the module uses a model which includes the daily movement pattern and the observed connectivity for an optimal scheduling. The network operator and the user have to agree on this scheduling plan. Therefore, a request, including video ID, time when the pre-fetching must be completed, and the videos size, is sent to the operator automatically. The reply message includes the video ID and when to start downloading.

**3.2.2.4 Results and Discussion**

A study is currently conducted on the large dataset of mobile YouTube video views, a subset of the set announced in Section 2.1. The large subset comprises 10M requests to 1.6M videos by 700k users over two weeks. The focus is on the portion of a video watched based on the video’s category. Figure 12 shows that only a minority of videos are watched fully. The overall average is shown as a solid line. The other four lines correspond to specific video categories. This information was retrieved from the YouTube Data API. The scale of the x-axis in Figure 12 goes beyond 100% because a user may watch videos multiple times or jump back in the video. Alternatively this may be due to packet retransmissions. For about 70% of the videos only 50% or less of it was watched. This heavily varies depending on the video category. The number of videos of which was watched less than 1%, indicates which categories are browsed more random. For instance, of music video is watched less than 1% in 38% of the cases, while of most how-to videos (over 50%) is watched less than 1%. In only about 20% of all samples, music videos are watched fully or repeatedly. After the 100% mark, a sharp incline is observed for some categories. This indicates that if more than 100% of the video is watched, it includes only a small part in most cases. A similar effect is visible at 200%. This result clearly shows the potential for pre-fetching by considering videos as a group of segments with different consumption probabilities. The video’s category is suited for building probabilistic models. These models can be used to optimize segment-based pre-fetching mechanisms. There are video categories which tend to be browsed more randomly than others. Categories which are randomly browsed are likely to be watched for only a few seconds, e.g., entertainment or comedy, because the user is not targeting certain content. Non-randomly watched videos are those where the user has already a certain video in mind that he is looking for. To this end, it is reasonable for categories, which tend to be browsed more randomly, to pre-fetch mainly a few prefix-segments of related videos than for other categories. For instance, music seems to be more specifically targeted by the users, so that greater numbers of prefixes or even the entire video should be pre-fetched. It is planned to evaluate the proposed approach with a segment-based HR and the CPR (Correct Prediction Ratio) introduced in deliverable D4.2 [D4.2]. These results will be reported in WP6.
3.2.2.5 Summary and Outlook

This section proposed an architecture for pre-fetching segments of videos on mobile devices. The related work has been investigated and a new approach has been proposed which leverages the user’s context and social information. The approach enables to relieve the mobile network from peak-hour traffic. Results on the analysis of users’ preference patterns and partial video views show a high potential for category-based pre-fetching. Possible extensions may consider internal and external smartphone sensors or new mobile network technologies.

3.2.3 Mobile Tracing App

In this section, results and insights gained from the data collected by the Mobile Tracing App, also referred to as SonNet, are described. These results are meaningful with respect to the development of future pre-fetching algorithms running on a user’s smartphone. The special focus of this work is on videos posted on the Facebook feed of a user. The Mobile Tracing App is a modified version of the Mobile Social Pre-fetcher (see Figure 9) designed for a user study performed in 2013 (see deliverable D4.2 [D4.2]). Further results of this study are reported below.

3.2.3.1 Introduction

While the idea of pre-fetching sounds straightforward, realizing an effective pre-fetching mechanism is challenging as it requires precise predictions on which content items are likely to be consumed in future. As content access is highly related to the use of OSNs, deriving prediction information from these networks seems to be a promising approach. Content items that an OSN presents to a user, e.g., on its individual newsfeed, can be used as candidates that, with a high probability, are accessed by the user in near future. Thus, they should be considered for pre-fetching. Yet, some of the items might be more interesting for the user than others and, consequently, also accessed more or less likely. Facebook offers a rich set of metadata, which might be helpful to derive the relevance of an item for the user. Properties, e.g., related to the social graph or the number of likes and comments could influence this relevance.
To this end, this section presents an analysis of content properties on Facebook on their potential to be used for user-based content prediction. As base of the analysis, an initial pilot study including 14 subjects has been conducted, spanning several weeks. All subjects agreed on using a specialized app on their mobile devices that mimics the functionality of the native Facebook app but allows collecting media- and OSN-related metadata in a privacy-conserving way. The results indicate that mobile video consumption on Facebook is very diverse and that general prediction rules are hard to obtain. Yet, a limited number of attributes exist that could very well help to design effective video pre-fetching algorithms for mobile devices. The insights provided by this work can help to better understand the potential of content prediction for mobile pre-fetching based on OSN information.

3.2.3.2 Background and related work

Mobile video pre-fetching is beneficial for two main reasons. Firstly, it allows leveraging WiFi connections to pre-fetch content at reduced energy consumption and secondly, video start-up delays can be reduced. Regarding the energy consumption in cellular networks, Huang et al. [HQG12] showed that LTE network traffic is 32 times more energy consuming in comparison to WiFi. For the second reason for video pre-fetching - the reduction of video start-up delay - Krishnan et al. [KRI12] show for short video clips, which are predominately shared in OSNs, that a start-up delay of two seconds already leads to many users abandoning a video streaming. Thus, a decrease of the video start-up time is an important goal to enhance the overall quality in video streaming.

Kaafar et al. [KBD13] present a recommendation-aware content placement strategy for CDNs. This approach shows sources for designing social-aware pre-fetching strategies, but aims at improving caching and thus the delivery to thousands of users by placing the right information at the right place within the network, whereas pre-fetching concentrates on loading content to a device. Bai et al. [BJS13] investigates OSN-related caching mechanisms for Yahoo and Facebook.

Pre-fetching in P2P video streaming systems is a well-researched topic. The work of Wang et al. [WSY11] investigates RenRen, a popular OSN used in China, which works similar to Facebook, on its suitability for P2P-assisted video streaming. Pre-fetching of the first video chunks helps in this domain to reduce the video start-up time. The proposed pre-fetching strategy utilizes RenRen users’ preferences, their past video access patterns and the social closeness of the viewer and the source of a video to predict the likelihood of an unwatched video being consumed. Their results indicate that social relationships and users’ preferences are suitable predictors for OSNs similar to RenRen. One of the findings elaborates on the correlation between two users with the same interests and their historical and future video consumption.

A similar approach has been followed by Cheng et al. [CFL09] for YouTube. The author’s aim is to propose a system which scales well, reduces the server workload, and improves playback quality. During long-term measurements they show, that 99.6% of all YouTube videos have less than 700 seconds duration. As YouTube videos are likely to be consumed in the order of the search results or the related video lists of a currently played back video, the developed NetTube system utilizes ties between videos as a source for their predictor. In addition, social ties between users are integrated into the scalable P2P-assisted video system. Video sharing sites are the focus of Khemmarat et al. [KZK11], who propose pre-fetching based on related videos, and search query results. Again, the main goal is to reduce start-up delay for stationary devices. Therefore, the pre-fetching schemes only load prefixes of a video. The introduced schemes are evaluated with web browsing pattern traces. The HRs of 80.88% are achieved by neglecting storage space and network bandwidth.

Li et al. [LIW13] add insights on video dissemination in OSNs and identify very short life times of videos with low number of likes. The resulting SocialTube [LSW12, LCL14] system demonstrates the efficiency of a P2P-based OSN that is driven by social relationships and less by interests. In their analysis they showed that 0.4% of the videos already cause more than 80% of the traffic and the
remaining 99.6% only cause 20% of the network traffic. But most, around 90%, of the video views of a user can already be explained by shares of their 1- and 2-hop friends, but only 25% of the users watch 100% of the videos, 33% of users watch 80% of videos and all viewers watched at least 20% of the videos on their newsfeed.

None of these studies and pre-fetching strategies consider the effect of mobile devices or behaviour of mobile users. With the work by Gautam et al. [GPN13] a mobile application pre-fetches whole video clips based on arbitrary sources such as OSNs or newsfeeds. Goal of the work is reducing the energy consumption of a mobile device, but pre-fetching algorithms are not described in detail. The most promising approach is presented by Zhao et al. [ZHA13, DOZ14] in which a custom mobile Facebook application has been developed that integrates social network based algorithms for pre-fetching. They conduct a study on Facebook, as our work does, too. The findings are limited and have to be supported by simulative experiments. The results indicate a significant energy gain, if the number of videos shared in OSNs would be higher. But, the designed application, their sorting of the posts, mapping of the real Facebook newsfeed and privacy-awareness are not described in detail.

The work presented in this section addresses these issues by a detailed explanation of the mobile application, its suitability to preserve the privacy of users, a detailed description of the framework, and a study including the results and giving an outlook on future sources for media prediction.

3.2.3.3 Mobile App System Design

To gain valuable insights to media content consumption in OSNs, we conducted a user study with 14 participants. Each participant was asked to use an Android app, referred to as SonNet, on his own mobile device(s). Additionally, the participants had to sign a privacy consent form before the activation key for the app was handed out. The app’s main functions are tracking of the posts shown on the user’s Facebook feed, keeping track if the user watches the posts, and uploading the collected information to a database server for later analysis. The information was anonymized before uploading to ensure the privacy of the user. This was done by hashing all personal data-related fields with a hash algorithm (SHA-1). This way, e.g., multiple datasets by the same user could be related, without revealing any information on the user identity itself.

The architecture of SonNet is shown in Figure 13. SonNet collects information about all posts on the user’s feed. This information was retrieved using the Facebook Graph API, located in the Facebook Crawler, which is embedded in the Social Data Collector component. As the aim is to identify posts that the user is most interested in, a set of features of the post are collected as well. To this end, the Facebook Crawler fetches the number of likes and comments of the individual posts. Furthermore, the source of the post is identified to tell if direct friends of the user or friends of friends published or interacted with the posts. Additionally, the information whether or not the post was posted in a group the user is assigned to, is tracked. With this distinction, it is envisioned to distinguish between different persons or groups to be more or less important for the user’s interest in a post than other sources, which could be beneficial for a more personalized prediction of content consumption.
The participants of the study access Facebook with the SonNet Facebook App instead of the native Facebook client. This allows to precisely keep track of interactions the user makes with the posts, e.g., watching, liking, or commenting. The posts presented to the user are the ones which were previously retrieved by the Facebook Crawler using the Facebook Graph API. This information is collected and filtered by the Social Data Collector and accessible from the SonNet Facebook App. The Social Data Collector regularly queries for new posts on the user’s newsfeed. It is important to note that some limitations regarding the collection exist. First, the collected posts do not include any sponsored posts as they would appear in the native Facebook app. Second, the order of posts as retrieved by the Facebook Crawler is not necessarily the same as the one Facebook presents in the native app. The latter is a limitation imposed by the Facebook Graph API as it does not include any specific ordering information. As the sorting algorithm used by Facebook itself is not publicly available, the SonNet Facebook App follows a simplified approach and presents the posts ordered by creation time. Another difference to the native Facebook app is that the SonNet Facebook App allows users to mark video posts as being relevant. This is done to account for scenarios where, e.g., the user would have liked to watch the video, but did not do so due to its limited mobile data plan. Marked videos are treated in the following similar to actually watched videos.

The Social Data Collector passes the received post and user interaction information to the Data Aggregator, which filters and locally stores the data. The anonymization of all personal data-related fields takes place directly before storing the data. In a later step, the local database is transferred to the database server, where the datasets of all participants are stored.

In this work, the main focus is on understanding which features have an influence on the relevance of a post for a user and, therefore, might be predictive. Based on the results, it is planned to develop a Social Predictor component, which is ongoing work and was not part of the mobile app as used during the study. This predictor would take the user’s newsfeed as an input to train a machine learning model. The model is planned to be used to identify relevant newsfeed posts and use videos as well as images contained in these posts as pre-fetch candidates. These media files are then planned to be downloaded by the Content Pre-fetcher component, which is also not yet available. It is envisioned that if the user accesses a video that has been pre-fetched, it plays locally instead of using the mobile network to access the online version.

### 3.2.3.4 Description of a Dataset for Pre-fetching

This pilot study contains the data gathered from mobile usage of Facebook of 14 participants over a time span of eight weeks. Volunteering participants agreed on using a Facebook-mimicking app.
called SonNet. The data includes interactions of users with the posts, especially with those containing photo and video as well as social graph information such as social closeness of users or their interests. Only data on the media consumption of media posts are gathered. The participants had no possibility to like, comment, or share posts from SonNet. In total 2138 posts including 202 video posts have been gathered. Video posts contain Facebook’s original video posts, link posts to YouTube videos or other video sharing sites as well as the category ‘flash’ video.

The data gathered has to map real usage of the OSN as close as possible. Thus, all participants were motivated NOT to change their Facebook usage behaviour. All users were asked to use their cell or smartphone. Tablet devices or laptops - despite their mobility aspects - are not included in this study. Users participating did not know that this work concentrates on the video consumption behaviour of users. No users or posts were filtered and no data clearance steps have been taken.

Figure 14: Overview on posts gathered in the study over all users - ‘video’ includes Adobe Flash videos and links to video sharing-sites.

An overview on the quantities of different posts can be seen in Figure 14, whereas Table 2 lists numbers on the gathered dataset. To ensure privacy and comply with national privacy laws, all data has been anonymized using a hashing algorithm on the mobile device.

![Figure 14: Overview on posts gathered in the study over all users - ‘video’ includes Adobe Flash videos and links to video sharing-sites.](image)

<table>
<thead>
<tr>
<th>Table 2: Overview on the mobile pre-fetching dataset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Male:</td>
</tr>
<tr>
<td>Female:</td>
</tr>
</tbody>
</table>

| # Friends | 217.5 | 160 |
| # Posts | 152.71 | 61.5 |
| # Posts Watched | 15 | 3 |
| # Friends list | 72.28 | 77 |
| # Groups | 31.29 | 13 |
| # Interests | 10.78 | 6.5 |

3.2.3.5 **Analysis of the Mobile Facebook Dataset**

The newsfeed within the OSN Facebook has been analyzed according to the questions: (1) How much media – especially video – that is being distributed in the OSN Facebook is watched by the participants of the study? (2) Can the global number of Facebook likes of a video be used to tell, whether or not one of the participants is going to consume media? (3) Is the origin of a post a suitable predictor for media consumption? (4) Is video consumption significantly different for individual user or user groups and can those groups be modelled?

**Crucial questions of the study**

The previous section describes the overall properties of the dataset and the propagation of content in Facebook for this study. As a result, the first question elaborates on the proportion of photo as well as video posts being consumed by the mobile users. While YouTube is a highly video-centric...
platform, this is questionable for Facebook as videos do only account for 9% of the posts observed. Photos, in contrast, account for a large proportion of Facebook posts. Furthermore, an analysis of the share of watched versus non-watched content gives a first indication on the potential of media pre-fetching strategies for mobile devices. On the newsfeed, different content types can be found such as status updates, links, photos, and videos. The focus of this study lies on the pre-fetching potential of media content, including photos and especially videos. In this context, consumption is seen as the interaction with the distinct post on the newsfeed by clicking on the thumbnail of a video or photo.

The second question investigates the origin of a post. The hypothesis of this work is that the origin of a video or photo post significantly influences its consumption probability. To be more specific, this work postulates that video shared by close friends and/or family members are more likely to be watched in comparison to videos from other sources.

The third question addresses likes, comments, and shares associated with a post. It investigates whether those interactions with a post have a significant influence on the probability of a single video being consumed. To be more precise, this study follows the findings of Li et al. [LiW13], in which the global number of Facebook likes of a video increases its probability to reach one specific user in an OSN. Thus, the underlying assumption is that high numbers of likes, comments, or shares in an OSN result in a high probability for a single user to consume a post.

In general, users of OSNs are very diverse. Specifically, some users tend to use specific functions of an OSN, others do not. Therefore, the fourth question analyzes the potential of clustering the participants of the study according to their media consumption behaviour into distinct groups. Those groups can help to tune pre-fetching algorithms by applying individual strategies in order to achieve a higher precision.

**Mobile media consumption & influence of content type**

The first research question addresses the specifics of the gathered mobile Facebook media consumption dataset. For the consumption rate, touch/click events for photos or videos are monitored. For photos, the touch event maximizes the photo to full-screen size, for videos it starts the playback. In total, the average consumption rate of posts in total is 9.268% (Stdev.: 14.72%; Median: 0%). Videos exhibit on average a larger probability for consumption per user (Average: 19.15%; Stdev.: 28.5%; Median: 0%) in comparison to photos (Average: 7.755%; Stdev.: 14.53%; Median: 0%). This may have various reasons. First of all, the total number of videos is significantly lower in comparison to photos. In this study, nearly half of the posts belong to the category photo. The mass of photo content may influence consumption negatively, whereas the exceptional occurrence of a video on the newsfeed may influence the rate in a positive manner. Additionally, photos, in comparison to videos, are easier to be consumed directly from the newsfeed without any interaction. Small versions of photos, so called thumbnail versions, are presented on the newsfeed as preview. Even though these thumbnails have a reduced quality and do not allow assessing details, in some cases it might be sufficient to understand the context and content of the photo. In this case, some pictures might have been relevant for the user but not counted as consumed due to the missing touch event. Videos have a temporal dimension that cannot be obtained by solely watching the thumbnail. In the latest version of the Facebook website and mobile application this is different. Video key frames are automatically played back when the user focuses the video post, allowing capturing the temporal dimension of the video as well. This feature was not available during the presented study.

The consumption of media is very diverse, as the median of 0% and the standard deviation (photo: 14.53%; video: 28.5%) for both video and photo consumption show. Especially for videos, a group of users within this study did not watch content at all, whereas some other participants consumed a
large share of the videos. This diversity of the users is discussed in detail in Section 3.2.3.6 in the subsection on user diversity.

The results obtained in this study are with a consumption rate on mobile devices of 19.15% for videos significantly higher than those obtained in studies for stationary OSN access, e.g. by Li [LCL14] with around 14%. Users tend to watch more videos on their mobile device, but still the available video content is very limited, which reduces the potential for video pre-fetching on mobile devices. Still Facebook is transforming into a multimedia-affine OSN, as in combination with photos over 50% of all posts belong to either the video or photo category, resulting in a huge pre-fetching potential.

Origin of a post

The origin of posts is grouped into posts shared by a friend and posts shared by other sources. Shared in this context refers to any interaction of the friend that resulted in showing the post on the user’s newsfeed, e.g., the creation of a post, commenting or liking a post may bring it to the newsfeed. One of Facebook’s core competitive advantages is its algorithm to create the newsfeed according to the interests of its users, activities of his/her friends, and personalized advertisement by featured posts. This algorithm shapes the newsfeed of each user to its individual needs. This study based requests on the offered Facebook API which does not integrate the correct order of the algorithm and omits advertised posts, e.g., by advertisement partners or those of global interest. Thus, for observing the OSN newsfeed of the official Facebook app, the proportion of posts from other sources can be slightly higher.

For this study, the majority of all posts are shared by friends and only 32.45% of all posts stem from other sources. Specifically, this rate was significantly higher for videos with 44.56% and 40.9% for photos. Thus, in comparison to other post types photos and videos are increasingly being distributed by non-friends.

Table 3 shows the fraction of content being consumed versus non-consumed for photos as well as videos. There is no significant difference resulting from the source of a video or a photo on the consumption probability, if solely the average is evaluated. A slightly different observation can be gained from Figure 15 which again shows, the origin of a video post but as a box plot including the median. As the observed data suffers from significant variance, median offers a better insight. For videos a higher median for the proportion of watched videos can be observed for friend posts.
Table 3: Overview on the percentage of a video and photo being watched in relation to the origin of the media post.

Global OSN popularity

Previous studies [LIW13] show a significant impact of OSN like count of content on its consumption. Content on video sharing sites follows a so called Zipf-like distribution in which a large proportion of the videos is watched only by a very small amount of users, whereas the top percent has enormous access rates. The like count in the OSN Facebook is represented by the total number of interactions with the post. Interactions evaluated are likes, comments, as well as shares of posts. Please note, due to technical limitations, the maximum number of captured likes and comments was limited to 1,000 in this study. While this is a limitation that was removed for future studies, it only affects 1.1% of the comment and 6.1% of the like counts on posts in the dataset as they exceed this threshold. In the dataset, it can be observed that only a small share of posts reaches high numbers of likes, comments or shares.

Table 4: Summary of the influence of comments, shares, and likes on the consumption of video and photos.

Figure 16 shows the effect of likes and comments on the consumption of videos. No significant difference can be observed for the watched and non-watched videos influenced by likes and comments. Only 10% of the multimedia posts have high numbers of shares. This indicates that globally popular content that is being shared from user to user accounts only accounts for a very small proportion of the posts.

Regarding photos, the median of the number of likes is 102 and 10 for comments. A difference between the values of watched and non-watched posts can be observed for photos. No similar pattern exists for the video posts. The median number of likes with 5 (Stdev.: 339) for watched videos and 53 (Stdev.: 402) for non-watched video indicates that especially videos with a low number of interactions are of interest to a user. Figure 16 shows the small impact of the number of likes measured by likes, comments and shares for a large share (more than 80%) of the videos. Table 4 summarizes the impact of likes, comments and shares on media consumption. For the remaining 20% the Facebook like count has an impact, as the CDF for watched videos is significantly less increasing until around 900 comments. For this small proportion of posts a high number of shares gives additional potential for prediction. All video posts with more than 10000 shares are watched, indicating that globally relevant content, even though seldom available, is consumed by users. Interestingly, the CDF on the effect of increasing number of likes on the video consumption rate indicates that with an increasing number of likes the consumption rate of a video decreases. A closer
look on the effect of the number of likes shall be gained by Figure 17. It shows that watched videos have major jumps in consumption at around 300 and around 700 comments, whereas for photos especially near 1000 and more comments explain around 15% of consumption. In contrast to this non-watched content reaches nearly 100% at lower comment counts.

![Figure 16: (Left) Effect of number of comments, (Centre) effect of number of likes and (Right) effect of number of shares on the consumption rate of a video](image1)

![Figure 17: CDF on the impact of comments on the consumption rate of photos and videos – CDF limited to explain the upper 35% of the consumption.](image2)

Multiple explanations for the findings gained by Figure 16 and Figure 17 exist. First of all, videos being consumed are mainly shared within small groups and thus range. When creating a post on Facebook, a user has the opportunity to restrict the visibility of the post to herself only, or a specific group of friends, managed in friend’s lists or public. Another explanation would be that the users very much consume videos shortly after being published. As a result for this study, one finding is that for the majority of posts available on the newsfeed, the OSN popularity expressed by like, comment, and share numbers is not affecting the consumption probability. Nevertheless, a discussion of an inverse OSN popularity as predictor is interesting, as possible explanations include that the content is shared in small groups. Secondly, the audience to which it is shared is very diverse. Different interests in the set of friends and different using behaviours of Facebook may result in a low number of views and, thus, potential interactions such as likes or comments. A third reason is that the participants of the study consume the videos shortly after propagation. As the displayed list of posts on a mobile device is very limited especially in comparison to the classical desktop browser UI (User Interface), this is a very likely cause. This theory can be supported by findings by Li et al. [LIW13]. They reported that the average time span where a video’s view count increases significantly in an OSN is very limited. Thus, they can disappear before likes and comments gain a significant scale. Sharing reaches a far larger audience scale.
To better understand the results gathered, we compared those findings to the CDFs of photos shared on Facebook. The result can be obtained from Figure 18. The figure clearly shows a stronger influence of high number of likes and comments on the probability of a photo being enlarged. Still, new content or available content not able to reach large numbers of shares and likes exhibit the strongest probability to be watched. As global OSN popularity is not a well-suited predictor, the OSN popularity within the participant’s friends is evaluated. The different forms of interactions by friends with watched and non-watched posts can be seen in Table 5. It shows that the amount of interactions is very limited. On average only every fourth post on the newsfeed has an interaction by a friend. This significantly limits the prediction capability of likes, comments, and shares.

<table>
<thead>
<tr>
<th></th>
<th>Watched</th>
<th></th>
<th>Non-watched</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Posts</td>
<td>Avg. # per user</td>
<td>Posts</td>
<td>Avg. # per user</td>
</tr>
<tr>
<td>Photo</td>
<td>42</td>
<td>0.39252</td>
<td>190</td>
<td>0.247</td>
</tr>
<tr>
<td>Video</td>
<td>7</td>
<td>0.11475</td>
<td>10</td>
<td>0.07092</td>
</tr>
<tr>
<td>Total</td>
<td>49</td>
<td>0.2917</td>
<td>200</td>
<td>0.2169</td>
</tr>
</tbody>
</table>

**Table 5: Interactions by friends with a post including like and comment.**

In addition, the dataset shows that an approximation between likes, shares, and comments can be made based on having only a single information source. Especially for videos, the Pearson’s correlation between shares and comments reaches 0.9263, whereas it drops for pictures to 0.17864. In contrast to that, the correlation between comments and likes for videos reaches 0.6287 and around 0.79986 for pictures. The correlation for shares and likes ranges between 0.1409 (photos) and 0.43549 (video).

![Figure 18: (Left) Effect of number of comments, (Centre) effect of number of likes, and (Right) effect of number of shares on the consumption rate of a photo.](image)

![Figure 19: (Left) Box plot on photos being watched in percentage of total pictures depending on post originated from friend or other sources, and (Right) plot with same properties for videos – Subset of high consuming users.](image)

As the collection of number of likes and comments was limited to a maximum of 1000, the posts with a higher number are excluded. By this, the correlation converges to 0.7404 (photo) and 0.7848 (video), respectively.
OSN popularity is a highly accurate prediction mechanism for caching, but at least for this study only a limited impact on pre-fetching precision is observable. The consumption, especially of video, seems only to a very small portion being affected by the number of likes and comments. Due to a significant correlation of shares, comments and likes counts, the OSN popularity can be estimated relying on a single property.

**User diversity**

As previous sections show high variances, one of the potential opportunities is the exploitation of better knowledge on the users. Some users, for example, do not have any non-friend posts on their wall, whereas others have some and watch these videos with a higher probability than direct shares by friends. Table 6 illustrates the users in the study and their photo as well as video consumption rates. Table 6 allows the distinction of users into a group of users consuming a large share of multimedia posts and those who do not. The group of high consuming users includes U2, U3, U7, U8, U10 and U11. Note that a user is considered to be a heavy user not only based on the pure number of posts, but on the share of posts which he consumed compared to all posts which appeared on his feed. It already illustrates the diversity in willingness to consume media content on the mobile phone. In addition, Figure 19 shows that especially videos shared by friends are more likely to be consumed by high consuming users in comparison to any other origin. Even though the results still suffer from variance, it can be observed that the box plots of friend and non-friends can be more clearly separated in comparison to the previously discussed in Figure 15. This may give already a first indication on how to subdivide users according to their behaviour in an OSN. Profiling of users can be easily conducted in an OSN and pre-fetching prediction can thus be tailored to users consuming a large share of available content, whereas pre-fetching could be disabled for users with only a small proportion of multimedia content on their newsfeed.

<table>
<thead>
<tr>
<th>User</th>
<th>Photos watched</th>
<th>Videos watched</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Friend</td>
<td>Total</td>
</tr>
<tr>
<td>U1</td>
<td>0%</td>
<td>15</td>
</tr>
<tr>
<td>U2</td>
<td>20.8%</td>
<td>24</td>
</tr>
<tr>
<td>U3</td>
<td>0%</td>
<td>2</td>
</tr>
<tr>
<td>U4</td>
<td>0%</td>
<td>56</td>
</tr>
<tr>
<td>U5</td>
<td>0%</td>
<td>10</td>
</tr>
<tr>
<td>U6</td>
<td>0%</td>
<td>35</td>
</tr>
<tr>
<td>U7</td>
<td>0%</td>
<td>64</td>
</tr>
<tr>
<td>U8</td>
<td>0%</td>
<td>3</td>
</tr>
<tr>
<td>U9</td>
<td>0%</td>
<td>7</td>
</tr>
<tr>
<td>U10</td>
<td>22.33%</td>
<td>103</td>
</tr>
<tr>
<td>U11</td>
<td>25%</td>
<td>148</td>
</tr>
<tr>
<td>U12</td>
<td>0%</td>
<td>1</td>
</tr>
<tr>
<td>U13</td>
<td>0%</td>
<td>1</td>
</tr>
<tr>
<td>U14</td>
<td>0%</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6: Overview and comparison on users’ consumptions of video and photos based on the origin of a post.

Another idea focuses on the friend’s interests. Thus, a comparison of the interests of users anticipating in this study and the interests of their friends is done. Due to the anonymization of data transferred to our servers, only the interest categories could be compared. To better understand the influence of common interests, both the friends’ interests who shared posts and the users’ interests are compared for each individual post. The Euclidian distance between the number of interests in the categories, i.e. music, television, movies, books, and games is calculated. For both photos as well as videos, a lower distance indicates that user and friend have more interests in common. For all posts, a mean distance of 17.9 is calculated for watched, in comparison to 21.4 for non-watched content.
This already indicates that interests could play a significant role for predicting multimedia consumption. This difference shall be calculated for the high consuming group of users.

In Table 6 a reduced distance can be observed with an increased difference between the means of watched and non-watched videos. In contrast to our expectations, it is not a valid classifier for high consuming users, but still valid for those watching videos or photos only seldom.

From Section 3.2.3.6 the phenomena is known that for a large share of video posts, watched as well as non-watched, only small numbers of likes or comments exist and that a small number of likes and comments does not mean that the videos are not watched. It shows that many videos are consumed quite quickly after publishing or that they are distributed in only small groups of friends. This can be supported by Table 7, which investigates friend’s lists. The distance to a given user is approximated by the membership in a friend’s list. Table 7 shows the subset of posts that has been shared by users within a friend’s list. Despite the anonymization of the traced data, standard Facebook groups such as ‘family’ and ‘close friends’ can still be identified. For both videos and for photos, posts by close friends and family are preferred. The information on the social distance to a posting user could, thus, easily be identified and leveraged for pre-fetching mechanisms. The videos shared by close friends or family are predominantly those with low numbers of likes or comments. None of the videos re-shared by these friend list members has more than one thousand or more likes. This demonstrates that videos are either (1) consumed quickly after being shared, or (2) the videos watched from friends do not have the necessity to be very popular in the social network.

<table>
<thead>
<tr>
<th></th>
<th>Photos</th>
<th>Videos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Friends</td>
<td>70.1%</td>
<td>85.7%</td>
</tr>
<tr>
<td>Family</td>
<td>82.6%</td>
<td>50%</td>
</tr>
<tr>
<td>Other lists</td>
<td>9.2%</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

Table 7: Friend lists and their impact on consuming video and photos. The table shows the percentage of media shared by members of a friend list that is being clicked.

3.2.3.6 Discussion and Conclusion

In this section, an analysis on features available in the OSN Facebook has been conducted in order to test them according their suitability to predict media consumption. The results indicate that multimedia content is increasing on Facebook, as more than half of all captured posts were images or videos. From this pilot study with only a limited number of users, no single predictor explaining media consumption can be retrieved. Pre-fetching mechanisms solely based on the pure number of likes and comments are not adequate for predicting whether a video is going to be watched or not. Likes and comments may predict quite well the OSN popularity but most of the content is watched when it has only a local OSN popularity.

Possible prediction sources can be the interests and the social closeness. While social closeness is mostly measured by hops in the social graph, this study gives insights that being friend alone is not enough. Friend lists seem to be, at least for those users who use them, one abstraction layer that increase prediction precision.

The findings gained have to be validated in the upcoming large-scale study. By being independent of a custom app and allowing users to use their smartphone web browsers to access Facebook or the official app, more reliable data is expected to be gathered. This will be reported in WP6.

One of the major limitations stems from the fact that the video content - which would be most beneficial for pre-fetching accounts for less than 10% of the total posts in the gathered dataset. With this limited number of video posts, one of the simplest strategies would be to pre-fetch any video on
the user’s newsfeed as soon as a WiFi network is available. The energy savings achieved within a WiFi network are beneficial and the required storage is, due to the short durations of the clips, sufficiently available. A potential to enhance the robustness of pre-fetching algorithms would be analyzing the content itself, e.g., by investigating the metadata hosted on the video sharing site that is associated with a video and integrate the information into the prediction module. Comparing interests and genres or tags of videos could be an interesting future approach for pre-fetching modules.

### 3.2.4 Network resource allocation algorithm

Mobile networks are increasingly constrained by limited spectral resources, while at the same time user traffic demands are growing steadily [WAN14]. Researchers are addressing this challenge from a variety of perspectives including massive multiple-input multiple-output communications, heterogeneous networks combining femto, micro and macro cells, device-to-device communication, and the concept of exploiting knowledge about user behavior and the network itself for performance optimization.

Recent studies [PAU11] highlight how network dynamics [SHA11] can be understood, predicted and linked to human mobility patterns [GON08]. This ability to predict user and network behavior allows optimizing resource allocation [VEC13, ABO14]. As such, it is a highly interesting approach to increase the efficiency of mobile networks and deal with future traffic growth.

In this section, we propose a resource allocation algorithm for mobile networks that leverages link quality prediction and prediction reliability. We first design an optimal resource allocation algorithm that assumes perfect knowledge of the future, as well as a general user capacity forecasting model for mobile networks. We then combine the two to obtain an iterative algorithm achieving near-optimal performance by taking into account the forecast reliability. While some prior work dealt with prediction-based optimization and uncertainty [RAS12, AKK11, JOK13] our approach is the first to combine multi-scale forecasting with a resource allocation scheme that accounts for imperfect throughput prediction in mobile networks.

#### 3.2.4.1 System model

In this section we address the downlink from a base station of a mobile network (eNodeB) to a single receiver (referred to as UE (User Equipment)). To simplify the description of the problem, we consider slotted time with slot duration $t$ and thus the quantities discussed in the paper are discrete time series. The quantities of interest are:

- **Position $p_i$** is the distance between UE and eNodeB in slot $i$ and $P$ is whole time series.
- **Active users $n_i$** is the number of active users that are in the same cell as the UE. It reflects the congestion level of the cell.
- **SINR (Signal to Interference plus Noise Ratio) $s_i$**, where $s_i = s_0 p_i^{-\alpha} f_P$. Here, $s_0$ is a system constant, $\alpha$ is the path loss exponent and $f_P$ is a random multiplicative term to account for fast fading.
- **User cell capacity $c_i$** represents the average capacity obtained by the user in the slot. We compute $c_i = c_0 g_c(s_i, n_i)$, to express the link between capacity and signal quality and congestion. $c_0$ is a system constant and $g_c$ is a technology dependent function which models system level variables such scheduling policy, congestion, spectral efficiency, etc. In what follows we consider LTE as the mobile network technology and we adopt the model in [OST11]
- **Receive rate $r_i$** is the rate at which the base station sends data to the UE
- **Download requirement $d_i$** is the data consumption rate needed to satisfy the user QoE. If at any time the user receives more data than required, the excess can be stored in a buffer for later use.
• Buffer state $b_i$ is the buffer level and $B_M$ is the buffer size.
• Buffer under-run time $u_i$ is the fraction of time for which no data was available to satisfy the download requirements.

The aforementioned quantities are linked as follows:

$$
b_{i+1} = \min(\max(b_i + r_i - d_i, 0), B_M)$$

$$
u_i = \begin{cases} 
\max(d_i - r_i - b_i, 0), & d_i > 0 \\
0, & d_i = 0 
\end{cases}$$

In what follows, we refer to function $y = g_y(x)$ as $g_y$. Similarly, we refer to the probability density function and the CDF of a random variable $X$ as $f_X(x)$ and $F_X(x) = \int_{-\infty}^{x} f_X(y) \, dy$ and with $\mu_X$ and $\sigma_X$ to its mean and standard deviation.

### 3.2.4.2 Resource allocation optimization with perfect forecast

The resource allocation problem aims at finding the optimal rate time series $R$ that satisfies the download requirements $D$ by using the available capacity $C$ in the best way. We define the following objective function:

$$O = \{o_i = r_i / c_i \in [0,1]\}$$

where $o_i$ is the fraction of the available capacity used and represents a cost. Note that the same rate has a usage cost which is inversely proportional to the available capacity. We obtain the following optimization problem:

$$\begin{align*}
\text{minimize} & \quad \sum_i o_i \\
\text{subject to} & \quad \sum_i u_i = \sum_i u_i^* \\
& \quad 0 \leq b_i \leq B_M
\end{align*}$$

where $u_i^*$ is the minimum feasible buffer under-run time. To minimize this cost function, the base station should send more data when the available capacity is high and use just the minimum rate required to avoid a buffer under-run when the capacity is low.

The solution of the problem above is the optimal resource allocation strategy $R^*$ that achieves the minimum buffer under-run time $\sum_i u_i^*$ at the lowest cost $\sum_i o_i^*$. If the sequence $C$ is known a priori, various offline algorithms can be used to determine the optimal resource allocation. We use a simple WF (Water-Filling) algorithm, which is able to achieve optimality using the following rules:

1. define the break-point $e_l$ as the last slot for which all previous rates are finalized (i.e., no more rate can be used in slots up to $e_l$) if either $b_{e_l} = B_M$ or $r_k = c_k \forall e_l < k \leq e_l$;
2. define an optimization window $[e_l + 1, m]$;
3. starting from $l = 0$, $e_l = 0$ and $m = 1$ the algorithm accounts for the slots in the set $\{e_l + 1, \ldots, e_l + m\}$ to satisfy the requirements up to slot $e_l + m$; the algorithm chooses a slot if it has the highest capacity among the unused ones in the set;
4. the algorithm either increments $l$, updates $e_l$ and resets $m = 0$ if a break-point is found or increments $m$ otherwise.

This WF algorithm obtains an optimal solution for the minimization problem above. We omit the optimality proof and the algorithm code here, but the interested reader is referred to [BUI15] for further details.
3.2.4.3 General forecast model

In this section we summarize the model we developed in WP5 to account for forecast reliability (see deliverable D5.3 [D5.3]). 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 [SAD04, QIA04]. It is characterized by the reliability time $\tau_p$, which defines how many slots of the sequence can be predicted.

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 [NIC08] can usually compute the likelihood of visiting a given cell, while trajectory-based predictors [FRO08] 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 availability based on the steady state distribution of the user position in the network.

3.2.4.4 Resource allocation optimization under uncertainties

The objective of this section is leveraging the concepts of the previous ones to design a network resource allocation algorithm which takes into account imperfect forecast and that we called ICARO (Imperfect Capacity prediction-Aware Resource Optimization). ICARO aims at minimizing the communication cost while avoiding buffer under-runs. In particular, we exploit the water-filling algorithm in an iterative way. At each iteration the WF algorithm makes a single decision about which rate $r_i$ to use by exploiting both predictors described in deliverable D5.3 [D5.3].

Using an iterative algorithm allows ensuring that the optimization algorithm is only making decisions about actual capacity values, but taking into account the future evolution of the sequence.
Before describing the new algorithm, we first have to combine the aforementioned tools into a single general capacity prediction. In order to account for the three time periods we proceed as follows (see Figure 20):

1) The short term prediction $c^{(F)}_i$ is obtained from the past capacity information collected, for example, by means of lightweight measurements [DOV04] and choosing the filter order and coefficients based on the user speed $\nu$.

2) The medium term model $f_{C,i}(x)$ is computed as the superposition of the cells $j$ that the user is likely to visit in the $i$-th time period, each of them accounted for according to their user position $f_{P,i}(y)$ [FRO08] and active user number $f_{N,i}(z)$ [PAU11] statistics. Similarly, the duration of the $i$-th time period $\tau_i - \tau_{i-1}$ is obtained as a weighted sum of cells traversal time $T_j$ related to cell $j$ using the probability of visiting cell $j$ as weight.

3) During the $i$-th time period $D_i = \sum_{j=\tau_{i-1}}^{\tau_i} d_j$ bytes has to be downloaded to avoid buffer under-run. The maximum cell efficiency is achieved when only the slots with the highest capacity are used.

4) The highest threshold $c_{T,i}$ is computed so that the average amount of data obtained by selecting only the slots with larger a capacity than $c_{T,i}$ is larger than $D_i/\left(\tau_i - \tau_{i-1}\right)$:

$$c_{T,i} = \max_p \; \text{s.t.} \; \int_{-\infty}^{\infty} x f_{C,i}(x)dx \geq D_i/\left(\tau_i - \tau_{i-1}\right)$$

5) The $i$-th time period is modeled as a sequence of $\tau_i - \tau_{i-1}$ values

$$c^{(M,i)}_j = \begin{cases} c_{T,i} & j > \left(1 - F_{c,i}(c_{T,i})\right) \left(\tau_i - \tau_{i-1}\right), \\ 0 & \text{otherwise} \end{cases}$$

where $F_{c,i}(c_{T,i})$ is the probability of the capacity being lower than $c_{T,i}$, thus $\left(1 - F_{c,i}(c_{T,i})\right) \left(\tau_i - \tau_{i-1}\right)$ is the average number of slots with larger capacity than the threshold.
6) Steps 2 to 5 are repeated and new time periods are added in the sequence if their reliability is sufficient (two cells, if Markovian predictors are used [NIC08]).
7) Compute $\tau_0$, as the offset time when the user first entered in the cell.
8) Obtain the predicted capacity sequence as the concatenation of the previously computed time period sequences (mixed prediction equation):

$$\bar{c}_i = \begin{cases} 
    c_0 & i = 0 \\
    \bar{c}^{(F)}_i & 0 < i \leq \tau_p \\
    \bar{c}^{(M,1)}_i & \tau_p < i \leq \tau_1 \wedge \tau_1 > \tau_p + \tau_0 \\
    \bar{c}^{(M,2)}_i & \max\{\tau_0 + \tau_p, \tau_1\} < i \leq \tau_2 \\
    \vdots & \\
    \bar{c}^{(M,n)}_i & \tau_{n-1} < i \leq \tau_n 
\end{cases}$$

where $\tau_n$ is the duration of the whole sequence, $c_0$ is the known present capacity, $\bar{c}^{(M,1)}_i$ is modified by removing slots from the beginning if the past capacity values and those predicted with $\bar{c}^{(F)}_i$ are lower than $c_{\tau,1}$ or from the end if the opposite is true.

Figure 20 shows an example of a mixed model sequence: in the upper part, the thin solid line represents the ground truth available capacity $C$, the thick solid line is the short term prediction $\bar{c}^{(F)}$ and the dashed line represents two cells through their statistics by means of $\bar{c}^{(M,1)}$ and $\bar{c}^{(M,2)}$, respectively. The lower part of the figure represents a map where the user is moving from the left to the right following the central horizontal solid line. The shaded area highlights the area where the user is likely to be. The dashed circles represent the coverage areas of different cells. Finally, dash-dotted lines crossing the two parts of the figures mark $\tau_p$, $\tau_1$ and $\tau_2$ instants.

ICARO, which is detailed in the following Algorithm 1, uses the WF algorithm to allocate rate iteratively based on the mixed forecast sequence of step 8 above. The algorithm ends if the total number of remaining required bytes is smaller or equal than the current buffer level.

**Algorithm 1: ICARO**

<table>
<thead>
<tr>
<th>Imperfect Capacity prediction-Aware Resource Optimization (ICARO)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT</strong>: requirement $D$, user speed $v$ and position $p$, $\tau_p$ past capacity samples, capacity statistics $f_{C_i}(x)$ and traversal time $\tau_i$ for the next $i$ time periods.</td>
</tr>
<tr>
<td><strong>OUTPUT</strong>: allocation policies $R$, associated cost $O$ and buffer under-run time $U$</td>
</tr>
<tr>
<td>$s = 0$ // set the starting point</td>
</tr>
<tr>
<td>$b_s = B_0$ // set the starting buffer</td>
</tr>
<tr>
<td>$\tau_s = 0$, $R = \emptyset$ // set the starting allocation</td>
</tr>
<tr>
<td>while $\sum_{i=s}^{[s]} d_i \geq b_s$</td>
</tr>
<tr>
<td>compute $\bar{C}$ as per step 8 above</td>
</tr>
<tr>
<td>run $\bar{R} = \text{WF}(\bar{C}, D, b_s)$</td>
</tr>
<tr>
<td>$\tau_s = \min{\tau_1, c_s, B_M - b_s}$ // policy for the current slot</td>
</tr>
<tr>
<td>if $c_s &gt; 0$</td>
</tr>
<tr>
<td>$a_s = \tau_s / c_s$ // policy cost</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>$a_s = 0$</td>
</tr>
<tr>
<td>endif</td>
</tr>
</tbody>
</table>
if \( d_s > 0 \)
\[
    u_s = \max\{d_s - b_s + r_s, 0\}/d_s \quad \text{// under-run time}
\]
else
\[
    u_s = 0
\]
endif

\[
    b_{s+1} = \min\{\max\{b_s + r_s - d_s, 0\}, B_M\} \quad \text{// next buffer}
\]

\[
    s = s + 1
\]
\[
    D = \{d_i, s < i \leq |D|\} \quad \text{// remove the first requirement}
\]
\end{algorithm}

The rationale for using the WF algorithm on the mixed forecast sequence is that its operational principle, that selects which slot to use in descending order, still works under uncertainties and provides a solution which is conservative (as the highest capacity slots are placed last) to avoid under-runs, and aggressive (as the allocation priority is given to the most reliable slots) to optimize allocation costs. In the following, we provide a few examples of the algorithm:

**Ordering the short term forecast:** the elements of the short term prediction sequence can be assumed to have the same order of those of the actual sequence. Thus, the WF algorithm can be used on the short term prediction, because its order is likely to match that of the actual sequence.

**Ordering between short and medium term forecast:** the \( i \)-th medium term period is represented as a sequence of \( \left(1 - F_{C,i}(c_{r,i})\right)(\tau_i - \tau_{i-1}) \) slots with no available capacity while the remaining slots are equal to \( c_{r,i} \). Hence, if the short term prediction is lower than the threshold only the minimum rate is used, since from the statistical model, slots of higher capacity are expected to come later. Conversely, if the short term prediction is larger than the threshold, it is more likely that the remaining slots will be lower than the threshold (compare with step 8 of the sequence creation).

**Buffering:** the algorithm will always try to use the slots above threshold in each time periods and bridge those by using the buffer. By positioning the slots with highest capacity at the end of each time periods we ensure that the algorithm is conservative. Finally, the maximum buffer size \( B_M \) limits the optimization horizon of the algorithm.

Figure 21 shows an example of ICARO’s performance compared to the optimal boundary (OPT) obtained with perfect forecast and to the trivial (FULL) solution which maintains the buffer as full as possible. The left part of the figure shows three plots of the used rate \( R \) of the three algorithms: ICARO in the topmost part, OPT in the center and FULL at the bottom. In all three plots the shaded area represents the used part of the total available capacity which is drawn as a solid line.
Figure 21: ICARO algorithm output is compared to the optimal allocation (OPT) and the most conservative approach (FULL) on the left. On the right the differences between the buffer state and the cost evolutions of the three algorithms.

The right side of the figure shows an example of ICARO’s cost compared to the optimal boundary (OPT) obtained with perfect forecast and to the trivial (FULL) solution which maintains the buffer as full as possible.

The main difference among the three solutions is that FULL continues to fill the buffer during the low quality period at about $i = 25$, while OPT just use the needed quantity to be able to harness the best part of the second cell ($i = 50$), while ICARO, being more conservative than the optimal solution, accumulates more in the beginning and needs to make some suboptimal decisions (i.e.: $i = 80$). In the rest of the trace, FULL continues downloading just the needed to maintain the buffer full, OPT is able to use the best slots only, while ICARO performs very close to OPT.

Similar considerations can be derived from the right side of the figure: the upper part shows the buffer variation for the three schemes, while the lower part reports the cumulative cost. We can remark that ICARO performs very close to OPT, but it is always slightly more conservative as the buffer is always a bit fuller earlier on. Also, even though the cost is higher than the optimal, the two algorithms perform quite the same.

### 3.2.4.5 Results

In this section we provide an analysis of the overall performance of our algorithm. Since, to the best of our knowledge, no other solution is able to compute resource allocation while accounting for the impact of prediction uncertainties, we compare our solution with the optimum offline allocation (OPT) computed with the optimal WF algorithm on the exact capacity time series and the most conservative approach (FULL) which just fills up the buffer as soon as possible and maintains it as full as possible until the download requirements are satisfied.

The main performance metrics we are interested in are the objective function $O$ and the buffer under-run time $U$. In order to be able to mix the results of every tested configuration, we adopt the average cost $\xi = \sum_i q_i / |O|$, the average cost saving $\eta = (\sum_i q_i^{\text{FULL}} - q_i^{\text{ICARO}}) / \sum_i q_i^{\text{FULL}}$ obtained by our algorithm, and the average buffer under-run time increase $\zeta = \sum_i u_i^{\text{ICARO}} - \sum_i u_i^{\text{OPT}}$.

We analyze two LTE network scenarios: a suburban environment with users moving at moderate vehicular speed and a pedestrian urban environment ($\mu_v = 5$ and 1 meters per second). Both scenarios have been simulated by generating random networks of 100 adjacent cells with average
distance between base stations of 500 meters and 1.5 kilometers respectively. Results obtained in the two scenarios are shown in Figure 22 and Figure 23.

Figure 22: Overall performance comparison among ICARO, OPT and FULL algorithms in the suburban scenario. $\xi$, $\eta$ and $\zeta$ for different operational conditions are plotted on the left, centre and right respectively.

To generate capacity traces we use the Hata model [HAT80] for the path loss, the Rayleigh distribution for the fast fading and we follow the analysis of proportional fair scheduling in [OST11] to obtain the capacity distribution of a UE at a given distance from the eNodeB, when there are $N$ active users uniformly distributed in the cell.

Each simulation group is defined by a network deployment and a reference path which the user follows to cross the network. The time it takes to traverse the path is $2000 t/u$ seconds. To train the system we study 50 other paths following random trajectories within a cell coverage range from the reference path. We use trajectory based predictors in suburban simulation and Markovian estimators for the urban ones. Subsequently, we validate the system on 25 other paths generated from the same reference while using the information gathered from the training set for the predictions.

For each tuple $(\nu \in [0.5, 5], t \in [0.5, 5])$ we generate 20 groups of simulations. Finally, during the validation we vary the requirement over capacity ratio $(\sum_i d_i / \sum_i c_i) \in [0.1, 0.9]$ and the normalized buffer size $(B_M \sum_i c_i / \sum_i d_i) \in [1, 200]$.

Figure 22 (left) shows the average cost $\xi$ of the three algorithms (OPT, ICARO and FULL as solid, dashed and dash dotted lines, respectively) varying the buffer size (x-axis) for $(\sum_i d_i / \sum_i c_i) = \{0.1, 0.5, 0.9\}$ (upper, center and lower plots). In the upper plot the download requirements are moderate and both OPT and ICARO are able to obtain a normalized cost lower than 0.08 (corresponding to 80% of ratio between requirement and capacity), while FULL needs 95% of the resource. The performance is coherent in the other plots and ICARO is always better than FULL and close to OPT. As expected, ICARO performance improves when the buffer is larger and the requirements are lower. Notably, when the buffer is very small the three algorithms perform the same. However, the performance degradation for large buffer size has to be ascribed to the simulation length: in fact, in order to fully exploit a large buffer a proportionally longer time is needed.

The central figure shows contour plots of ICARO’s efficiency $\eta$ using $B_M \sum_i c_i / \sum_i d_i$ as abscissa and $\sum_i d_i / \sum_i c_i$ as ordinate: the curves are labeled according to the cost savings achieved and the area is colored with a darker shade if the saving is lower. Again the best results are obtained for medium buffer and small requirements where ICARO is more than 25% cheaper than FULL. On average, ICARO is 8% worse than OPT.

The figure on the right shows how close is ICARO to the optimal buffer under-run time obtained by both OPT and FULL. We plot $\zeta$ using the same coordinates as those of the previous figure. Here the
white part of figure highlights where ICARO is able to achieve optimal performance, while other darker areas correspond to slightly worse performance. Notably, for no parameters the buffer under-run time was larger than 0.01 and very rarely larger than 0.001, chiefly for high requirement and small buffer size. Also, the 0.0001 contour is noisy due to the system sensitivity to input parameters in that particular region.

Figure 23 provides results equivalent to those of the previous set of figures, but obtained for the urban scenario. Here, ICARO performance is slightly worse than those obtained in the suburban scenario. This is due to two main reasons: first, the urban environment shows a lower time correlation due to a more scattered fast fading and, second, to the lower accuracy of Markovian estimation compared to trajectory-based predictors.

Nonetheless, ICARO maintains, in the urban scenario, the positive outcome obtained in the suburban: the buffer under-run time (right) is most often optimal and, when it is not, the increase never reaches 1% of the buffer under-run time obtained by OPT; the cost savings (center) are mostly larger than 10% and as good as 20% in a limited parameter region.

Since ICARO gives priority to avoiding buffer under-runs, it can obtain higher cost savings when the ratio between requirements and available capacity is lower. Thus, since ζ is always lower than 1%, the algorithm is able to trade off cost efficiency for robustness effectively and it is able to achieve up to 25% cost reduction when the conditions are favorable, but it is never too aggressive when the future capacity estimation does not allow doing so.

Finally, when ICARO achieves the best results, the system is able to sustain the same quality of service while saving up 25% of the network resources or, analogously, 25% more users can be served with the same capacity. Conversely, where it obtains the worst performance, it is the system condition itself that offers small improvement margins: in particular for very small buffer sizes and/or high requirements.

3.3 P2P-Social-Assisted Content Delivery

As it is mentioned in Section 1.2.3 of this report, we have designed a beta version of a Facebook app (named “SocialiVideo”) to demonstrate the idea of P2P-social-assisted content delivery. Although the beta version was able to show the main concept behind the idea, there were several aspects that had potential to be improved and extended. In this section, as the next steps we improved the previous beta version of the application as well as the idea in several aspects as follows.
3.3.1 Design a user friendly interface

The previous version of the application had a naive interface which simply includes a list of users who are using the application and the videos inside each user account in a separate page.

Figure 24 shows the previous version of the interface.

The goal of the new interface is to provide a user friendly environment which clients of the application experience as something similar to the original Facebook wall page. The idea is to have a simple wall page that includes videos in each user account sequentially in order of the publishing time, and users can scroll the wall page down/up and easily stream the videos by clicking on it. In the
stream time of the videos, there will be a note which says from which location (Local or CDN) the video is streaming to the users on that time.

We have changed the previous interface with the mentioned idea and we designed a wall page style interface that is able to stream videos in the wall page similar to the actual Facebook wall page.

Figure 25 shows a snapshot of the implemented interface. As it shows we have videos of the users friend in order of their published times, and the top videos in each account shows the recently added videos from a friend.

Figure 26 shows the interface in the time that user click on a video and the video is streaming. As it shown the note shows that the video is streaming from a local server to the user.
3.3.2 Keeping local videos in a real local server instead of dropbox.

In the previous beta version of the application, we have used dropbox as our local server to keep the copy of the user generated videos as local videos. At that stage the goal was just presenting a general vision of the idea, but now we want to use a real local storage to keep the videos and stream them easily from it. The best approach is to locate local videos in the user’s local premises with a 24/7 connectivity to, e.g., the user’s set-top box in her home. But at this stage we are considering the local device storage which can be hard disc of a pc or laptop.

To this end, we need to implement the application in a way that connects to a real local server and stream the videos from there. We found two ways to do this procedure:

i) Network-based video streaming: one solution in this way is using VLC media player which can be used as a server and as a client to stream and receive network streams.


We have tested both approaches and evaluated their performance but for the application we implemented the network-based approach. In this approach, we considered a shared folder in each device as a local server of the user. A copy of the user generated videos will be located in the shared folder and the application will stream them directly from that folder via VLC media player. In this procedure, we need to install VLC in its server mode that provides us the functionality to connect to
it and stream the videos remotely. Please note in this approach we are considering that the users have a valid IP address which is visible for all friends.

3.3.3 Adding the capability to download shared YouTube videos and add to the local server

The previous version of the application was designed for videos that are generated by users and the application is able to fetch those videos from users profile or the users can upload the videos directly to the local servers.

As improvement, we added some new features to the application which can provide the ability to download videos that are shared from YouTube in the user Facebook wall page and keep a local copy of those videos.

To this end, we have implemented a feature that enables the application to download the shared videos from Youtube by using the Youtube API and by this we are covering a considerable portion of the shared videos in Facebook wall pages which mainly come from Youtube.

As future work, we are going to extend this feature for other video portals to see if we can do same procedure for some of other video providers such as Vimeo.

3.3.4 Improve security features of the application and comply the Facebook privacy levels for each video and user

Security features should be an important part of our application, which we did not consider in the previous version of the application.

To consider some security features in the application, first of all we have added an administrative account from which users can manage the videos. This will allow users deleting videos from the local servers. Also we added a feature that detects deleted videos from the Facebook wall page and deletes those videos from local servers as well.

In addition, we implemented some security consideration for each video which is aligned to the Facebook security features. The first and the important security concern is that, users just should be able to see only videos that are shared by their friends rather than all the videos in the application. Therefore we considered this concern in the implementation: based on the friend list of users the application is able to make a list of people that should have the access each of the videos.

3.3.5 Efficient capacity management of local server

As the application is going to keep a local version of the uploaded/shared videos per each user, the capacity limitation is an important issue which needs to be considered in the design. For this reason we implemented some policy which can be tuned by users in their administrative accounts. We have two types of capacity management in the application:

i) Based on the capacity

Users are able to select the capacity that they would like to share (e.g. 500M, 1GB, etc.) manually in the administrative accounts. When the size of the allocated folder is full it will delete from the oldest published videos.

ii) Based on the videos’ published time

Users can select how long they want to have the local copy of the videos in their local account (e.g., keep the videos from last two weeks).
3.4 Social-enhanced Content Placement with Social Information

Following the work on the design and implementation (see Section 1.2.4) of Social-enhanced Content Placement solutions the implementation of the different designed algorithms was investigated and validated in a real environment. This work has been conducted in parallel to the design of the related demonstrator in WP6 that includes the modules developed as part of WP4 work.

Moreover, in addition to the implementation of social enhanced content placement algorithms, traditional algorithms have been implemented as well so that a meaningful performance evaluation can be conducted.

In particular the traditional algorithms implemented are the following ones:

- **LRU**: This is the traditional LRU caching in which upon the reception of a new request from and end user, the cache server looks for the content locally. If the content is stored by the cache server, it returns it to the end user and updates the timestamp of the last access to that content. If the content is not available in the cache, it fetches it from the central server and forwards it to the user. Moreover, it caches the new content locally. In case the cache is full, it replaces the least recent used content (i.e., the content with the oldest associated access time stamp) with the new one.

- **FIFO (First In First Out)**: This caching mechanism is exactly the same as LRU, with the exception of the replacement strategy. In this case the oldest content (i.e., the one that was earliest cached among those in the cache) is the one selected to be replaced upon the reception of a new content from the central server.

These traditional algorithms (especially LRU) work well with popular content since they predict very well when a content will become popular and are able to delete those that are not popular anymore. However in a very dynamic context such as OSNs where most of the content is not popular, the previous strategies are far from optimal. For instance, a content item \( C \) uploaded by user \( U \) can be of interest to user \( V \), and at the same time user \( V \) may be the only interested user in that content associated to a given local cache. However, once \( V \) requests the content, the procedure explained above will be executed and thus \( C \) would be stored in the local cache server for a long time, despite no other user apart from \( V \) is going to consume it, consuming storage space that could be used for another content item that might be of interest for some other users associated to the local cache.

In order to avoid such situation we have implemented the following social-enhanced caching algorithms:

- **s-LRU (social LRU)**: In this case, the caching system would be provided with information about the social graph, and thus, the social connection between users. Therefore, using that information the caching system can determine the number of friends \( N \) that a given user \( U \) has in each local cache system. The decision of caching a content item in the local cache server is modulated by \( N \). In particular, when a user \( V \) (friend of \( U \)) associated to cache server \( CS \) requests the content, the system operates as in a regular LRU scheme. If the content is stored locally it is served to \( V \) and the timestamp of the last access to the content is updated. If the content is not available locally, it is fetched from the central server and forwarded to \( V \). However, the content is stored locally only if \( N > T^* \), where \( T^* \) is a threshold defined by the caching system. If the condition \( (N > T^*) \) holds and the cache is full, then a regular LRU algorithm is applied and the new content replaces the least recent accessed content.

- **s-FIFO (social FIFO)**: The only difference with s-LRU is that when the replacement of content needs to be done, a FIFO policy is followed and then the replaced content is the oldest one.
Moreover, we have implemented two versions of each algorithm: push and pull. In the push algorithm when a user U uploads a content item, it is automatically pushed to the caching servers of the distributed caching system (in the case of s-LRU and s-FIFO only if the defined condition holds). In the pull algorithm a content item is cached in a local cache only after a request happens and in the case of s-LRU and s-FIFO only if the defined condition holds.

The two versions (pull and push) of the 4 algorithms (LRU, FIFO, s-LRU and s-FIFO) have been implemented in Java. Moreover they have been integrated into one of our demonstration platforms developed within WP6. This demonstrator has been implemented on top of a network emulator software that allows us to emulate realistic network topologies with realistic delays, bandwidth capacity etc. We refer the reader to the future WP6 deliverables to find further details.

Using our demonstration platform we can run tests in a variety of scenarios: modifying the strategy used (pull or push), the caching policy (LRU, FIFO, s-LRU and s-FIFO) or the value of T* and the cache size. All these possible scenarios provide different information in order to understand the improvement that the social enhanced algorithm offers in front of traditional algorithms.

Our experiments reveal that pull technique and the variants of the LRU replacement strategy achieve the best results. Then, in the rest of the section we present a careful comparison of the performance of LRU versus s-LRU obtained from the real experiments conducted on our demonstration platform. Specifically, we ran the experiments using 5 cache servers, with a fixed number of users per cache (i.e., 200 users). The posting pattern was obtained from the Twitter dataset collected as part of our WP3 work (see deliverable D3.1 [D3.1]). Finally, the experiments were stopped when the number of requests for each cache was over 700 requests.

In Table 8 we show the configuration parameters of the different tests. We defined two cache sizes: 20 and 30 content items and T* ranges from 10 to 25.

<table>
<thead>
<tr>
<th>Cache #</th>
<th>Cache Size</th>
<th>T*</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache #1</td>
<td>20</td>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>Cache #2</td>
<td>20</td>
<td>15</td>
<td>200</td>
</tr>
<tr>
<td>Cache #3</td>
<td>20</td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td>Cache #4</td>
<td>30</td>
<td>15</td>
<td>200</td>
</tr>
<tr>
<td>Cache #5</td>
<td>30</td>
<td>25</td>
<td>200</td>
</tr>
</tbody>
</table>

**Table 8: Configuration parameters.**

The metric that we use to evaluate the performance of each solution (LRU and s-LRU) is the HR, which is, the fraction of users requests that found the requested content in the local cache server. It is obvious that a higher cache size would imply a higher HR. In fact, the effective average HR we get in the experiments for LRU was 26% (±2%) and 36% (±1.7%) for cache size equal to 20 and 30 respectively. Table 9 presents the relative improvement (in percentage) over these effective HRs provided by s-LRU in comparison with LRU.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Cache #1</th>
<th>Cache #2</th>
<th>Cache #3</th>
<th>Cache #4</th>
<th>Cache #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment #1</td>
<td>5.48%</td>
<td>19.25%</td>
<td>10.55%</td>
<td>6.83%</td>
<td>8.36%</td>
</tr>
<tr>
<td>Experiment #2</td>
<td>3.54%</td>
<td>25.51%</td>
<td>12.60%</td>
<td>12.42%</td>
<td>10.19%</td>
</tr>
<tr>
<td>Experiment #3</td>
<td>6.17%</td>
<td>26.83%</td>
<td>4.96%</td>
<td>19.86%</td>
<td>14.29%</td>
</tr>
<tr>
<td>Experiment #4</td>
<td>7.86%</td>
<td>12.85%</td>
<td>19.54%</td>
<td>9.21%</td>
<td>8.59%</td>
</tr>
<tr>
<td>Experiment #5</td>
<td>10.60%</td>
<td>8.47%</td>
<td>5.37%</td>
<td>4.76%</td>
<td>9.24%</td>
</tr>
<tr>
<td>Experiment #6</td>
<td>8.43%</td>
<td>15.66%</td>
<td>11.08%</td>
<td>11.83%</td>
<td>10.41%</td>
</tr>
<tr>
<td>Experiment #7</td>
<td>4.82%</td>
<td>18.10%</td>
<td>13.55%</td>
<td>6.62%</td>
<td>7.38%</td>
</tr>
<tr>
<td>Experiment #8</td>
<td>9.51%</td>
<td>9.33%</td>
<td>15.38%</td>
<td>6.12%</td>
<td>11.28%</td>
</tr>
</tbody>
</table>
Table 9: Results of the experiments using different configuration parameters.

<table>
<thead>
<tr>
<th>Average</th>
<th>7.05%</th>
<th>17.00%</th>
<th>11.63%</th>
<th>9.71%</th>
<th>9.97%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Deviation</td>
<td>2.44%</td>
<td>6.82%</td>
<td>4.88%</td>
<td>4.93%</td>
<td>2.15%</td>
</tr>
</tbody>
</table>

In total, we have conducted 8 experiments for each cache configuration. In all of them the social enhanced algorithm outperforms the corresponding traditional algorithm. Specifically, the improvement in the hit ranges between 4.76% and 26.83%. Moreover, Table 9 also shows the average and standard deviation for each cache configuration. First, we observe that, independently of the configuration, the average HR improvement provided by s-LRU is higher than 7% in all cases. Moreover, we observe that both parameters (cache size and threshold T*) have an influence on the performance of our solution. It seems that low values of T* leads to lower hit-ratios since as we observe the experiment using the smallest value (T*=10) provides the worst performance. Moreover, the improvement provided by our algorithm is higher for higher replacement rate required in the cache. If we focus on the results for Cache #2 to Cache #4 for a fixed value of T* our results show that a smaller cache size, that imposes a higher replacement rate, leads to higher improvements. Therefore, we conclude that the more demanding the considered scenario is, the more beneficial the use of our social enhanced algorithm in comparison with traditional algorithms is.

In summary, our results confirm that the utilization of social information can help to improve content placement strategies for unpopular content distributed through social media. As part of the work in WP6 we will develop further tests with different configurations and report the final results within that WP.

4. MAPPING OF THE SECTION ON THE ARCHITECTURE AND USE CASES

In this section we associate the topics discussed in the present deliverable to the use cases described in deliverable D2.1 [D2.1] and the eCOUSIN architecture finalised in deliverable D2.4 [D2.4].

Six use cases were defined in [D2.1]:

- Enhanced Content Placement Using Users' Social and Coarse-Grain Location Information
- Optimization for the Delivery of Premium Content
- Social-Assisted Time-Unconstrained Content Delivery
- Mobile Content Uploading
- Personal Content Sharing Clouds
- Information-Centric and Social-Driven Content Delivery

The content dissemination layer of the eCOUSIN architecture [D2.4] comprises the following architectural building blocks:

- Content Placement Strategies
  - Caching
  - Pre-fetching
  - Resilience
- Content Copy Selection and Look-Up Algorithms
- Content Dissemination Algorithms

Table 10 maps the topics described in the present deliverable on those use cases and architectural components. A short explanation for the mapping is as follows:
• Section 1.2.1 and Section 3.1 deal with small (TELCO) caches and are relevant for the Caching module of the Content Dissemination Layer of the eCOUSIN architecture. Such algorithms could be envisioned in the Optimization for the Delivery of Premium Content use case.

• Section 1.2.2.1 and Section 3.2.1 to 3.1.3 are targeted towards the Social-assisted Time-unconstrained Content Delivery and the Mobile Content Uploading use cases. The two sections consider functionalities related to the Social Layer and the Content Dissemination Layer of the eCOUSIN architecture.

• Section 1.2.2.2 and Section 3.2.4 are also relevant to the Social-assisted Time-unconstrained Content Delivery and Mobile Content Uploading use cases. Besides with the Pre-fetching module, the two sections also interface to the Network Layer: in particular they describe some part of the Network Resource Configuration and the Network Monitoring modules of that Network Layer.

• Section 1.2.2.3 and Section Erreur ! Source du renvoi introuvable. predict which content items to pre-fetch (using the Prefsim simulator). Hence, the results are relevant for the Social-assisted Time-unconstrained Content Delivery use case and in the tuning of the Pre-fetching module of the Content Dissemination Layer of the eCOUSIN architecture.

• Section 1.2.3 and Section 3.3 deal with the P2P-Social-Assisted Content Delivery use case. This use case was not part of the six use cases described in deliverable D2.1 [D2.1], but it is used to show that the eCOUSIN architecture detailed in deliverable D2.4 [D2.4] is generic enough to accommodate a whole range of use cases. The algorithm that decides whether to redirect users to local copies of the content or copies stored in central servers, CDNs, is part of the Content Copy Selection and Look-up Algorithm module that resides in the Content Dissemination Layer.

• Section 1.2.4 and Section 3.4 are directly related with the Enhanced Content Placement Using Users' Social and Coarse-Grain Location Information use case. In particular the modules implemented are the Social Predictor of the Social Layer that maps users to cache servers in the content distribution infrastructure and the Content Placement Strategies through the new social-enhanced content placement algorithms s-LRU and s-FIFO in both push and pull modes.

• The algorithm developed in Section 1.2.5 will be running in the module Content Copy Selection and Look-up Algorithm belonging to the Content Dissemination Layer. The proposed solution specifically defines a mechanism to find content in an unstructured P2P network.

• The Resource Locator described in Section 1.2.6 is part of the Content Lookup module in the content dissemination layer of the eCOUSIN architecture and plays a key role in the Personal Sharing Clouds use case.

• The data set described in Section 2.1 and the analysis thereof (in the restricted version of this deliverable) is relevant for the performance aspects of the Social-assisted Time-unconstrained Content Delivery use case and the pre-fetching module of the Content Dissemination Layer of the eCOUSIN architecture.

<table>
<thead>
<tr>
<th>Section</th>
<th>Use case</th>
<th>Architectural block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.1+3.1</td>
<td>Optimization for the Delivery of Premium Content</td>
<td>Caching</td>
</tr>
<tr>
<td>1.2.2.1+3.2.1+3.2.2+3.2.3</td>
<td>Social-assisted Time-unconstrained Content Delivery</td>
<td>Complete architecture + Interface of Social Layer and</td>
</tr>
</tbody>
</table>
Table 10: Mapping of components on use cases and architectural building blocks.

5. CONCLUSIONS

The present deliverable is the last one of WP4 that studied the Content Dissemination Layer of the eCOUSIN architecture. In the three deliverables associated with WP4, components of that Content Dissemination Layer were described. Although each of the deliverables builds further on its predecessor, enough information is given to make each one self-consistent. It was detailed how they are various components of the eCOUSIN architectures are strung together to support one of the six uses cases described in deliverable D2.4 [D2.4] and the performance of the individual components and the overall architecture was analysed, mainly via simulations and statistical analysis of traces (some of which were acquired with partial implementations of the proposed architecture and some of which were obtained in WP3). These (performance) results will be further used and refined (via experiments) in WP6 when fine tuning the implementations of the demonstrators.

Apart from these guidelines that are being transferred to WP6, the main conclusion from the work in the present deliverable (and its predecessors) is that using social information is beneficial to improve the performance of content dissemination. However, the gain that can be obtained depends on the use case considered. In cases where the social relations are correlated with content consumption and are explicitly known, using social information yields a considerable gain (e.g., Facebook, Twitter). In this case it does not matter too much whether the social graph (from an online social network) or
the similarities in tastes (expressed through profiles or acquired by observation) are used as social information: the performance is about the same, if anything the latter outperforms the former ever so slightly (see deliverable D4.2 [D4.2]). For cases where a library of content is made available to all users the social relations are only implicitly known, so that the social information needs to be extracted from content consumption, e.g., in the video-on-demand and catch-up television use cases, the gain of using social information is at most marginal. The reason is that most content in such cases is globally popular anyway and for the content items that are consumed in a specific social group it is not known (in a mobile scenario) by which cache these will be served.

The use cases in which a gain was observed are now further developed in WP6, where the gain will be shown in an actual usage environment via small scale experiments.
REFERENCES


ACRONYMS

3G            Third Generation (mobile)
API           Application Programming Interface
ARMA          AutoRegressive-Moving Average
CDF           Cumulative Distribution Function
CDN           Content Distribution Network
CPR           Correct Prediction Ratio
CUTV          Catch-Up Television
DNLA          Digital Living Network Alliance
DRWR          Distributed Random Walks with Restart
dMLRU         Differentiated MLRU
eCOUSIN       enhanced COntent distribUtion with Social INformation
FIFO          First In First Out
FNR           False Negative Ratio
GGSN          Gateway GPRS Support Node
GPRS          General Packet Radio Service
HR            Hit Ratio
HTTP          HyperText Transport Protocol
ID            Identifier
ICARO         Imperfect Capacity prediction-Aware Resource Optimization
IP            Internet Protocol
LRU           Least Recently Used
LTE           Long Term Evolution
MLRU          Modified LRU
MPMR          Most Popular, Most Recent
OSN           Online Social Network
P2P           Peer-to-Peer
QoE           Quality of Experience
SINR          Signal to Interference plus Noise Ratio
s-FIFO        Social FIFO
s-LRU         Social LRU
TELCO         Telecommunication Company
UE            User Equipment
UGC           User Generated Content
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>UPnP</td>
<td>Universal Plug and Play</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>VOD</td>
<td>Video-On-Demand</td>
</tr>
<tr>
<td>WF</td>
<td>Water Filling</td>
</tr>
<tr>
<td>WiFi</td>
<td>Wireless Fidelity</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
</tbody>
</table>