enhanced COntent distribUtion with Social INformation

www.ict-ecousin.eu

Deliverable D6.2

Final Plan for System Integration and Assessment

Public, Version 1.0, 30 June 2014

Authors

FT	Bertrand Mathieu, Patrick Truong
AL-BELL	
IMDEA	Joerg Widmer, Nicola Bui, Foivos Michelinakis
TSP	Noël Crespi
ALUD	Ivica Rimac, Klaus Satzke
TUD	Christian Koch, Julius Rückert, Fabian Kaup, Timo Thräm, David Hausheer,
TI	Fabio Mondin
UCAM	
UC3M	Rubén Cuevas
Reviewers	Ivica Rimac

Abstract

The WP6 of the eCOUSIN project will demonstrate and assess the technical solutions developed in the other WPs. This deliverable assesses the finalization of the plan started in the deliverable D6.1 [D6.1]. Here, four different and integrated demonstrators show how to realize the overall project objectives and validate the project claims by means of thorough testing campaigns. In particular, the functionalities described by the WP2 functional architecture are integrated in four demonstrators, each covering a different use case scenario. The four demonstrators cover the innovative aspects of the overall architecture, and are used to showcase and assess the overall eCOUSIN results.

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

EXECUTIVE SUMMARY

This document updates the content of the deliverable D6.1 [D6.1] and provides the final descriptions of the various eCOUSIN's target demonstrators. It shows how they fit the overall architecture and drafts a program of high-level tests to check the system coherence during the entire project lifecycle. Demonstrators reflect use cases expressed by partners to fulfil their business related expectations. Demonstrators and platform components will be developed in parallel in order to speed up the prototyping phase and to setup a dynamic feedback process between functionalities required by the use case definitions and those realized in the platform. Also, this will finally make a solid match between demonstrators, architecture and the final platform. The main additions with respect to D6.1 are to be found in the module and the assessment sections: new modules have been added to better realize architectural functions; new test have been added and old test have been updated to further improve the integration and the completeness of the demonstration and assessment work.

TABLE OF CONTENTS

INTRODUCTION

The flexibility of content distribution models empowered by the viral effect of social sharing is having huge impacts on the network. We have proposed a set of novel techniques to address the associated challenges and we have selected several examples of use cases to demonstrate how our platform can dramatically improve network efficiency and/or users QoE. Furthermore eCOUSIN will be providing mechanisms to assist novel smart content delivery chains advancing the upcoming network infrastructure evolution, which is expected to bring more bandwidth at the edges.

In the following four demonstrators will be described covering the different functionalities included in the architecture defined in WP2. In particular the following aspects of the architecture will be addressed by the different demonstrators.

The content placement demonstrator will focus on showing the optimizations for the delivery of the long tail of the distribution of the content popularity (medium to low popularity) studied in WP4. This will be achieved using emulation platforms such as PlanetLab or Emulab configured so as to mimic characteristics of real user social patterns.

The second demonstrator is called Personal Sharing Clouds and addresses the interactions between users of different social networks by leveraging on unique user identities. This concept makes it possible to access and interact with contents and users of any social networks the user is connected to without the need for using different credentials in each social network.

The third demonstrator addresses the implementation of a Twitter-like social networks with live streaming support adopting a Content Centric Network based architecture and open source protocols. The ultimate goal of the demonstrator is to show the improvement on content delivery performance thanks to the adopted social-aware content centric solutions.

Finally, the fourth demonstrator will concentrate on the benefits provided both to the user QoE and to the overall network performance by leveraging the end user device capabilities to optimize content distribution in terms of time, network and content.

1. DEMONSTRATORS

This section will describe the demonstrators and how they will be realized, starting from modules developed by individual partners and followed by their combination into joint showcases and testbeds.

1.1 Content Placement

1.1.1 Motivation

On the one hand, content, and more specifically video, is responsible for more than half of the Internet traffic nowadays. This trend is expected to increase, and as reported by CISCO[1], video traffic will be responsible for 67% of the whole Internet traffic by 2017. As a consequence, different stake-holders such as Content Providers or Internet Service Providers (ISPs) seek for techniques to improve the content delivery process, and Content Delivery Networks (CDNs) are expected to carry more than half of the Internet traffic by 2017. Therefore, we expect an increase in both the volume of content to be distributed and the utilization of optimized content delivery techniques, in the next years.

[1]http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

On the other hand, it is well known that content popularity in the Internet follows a Zipf distribution with a few contents attracting a large number of downloads and a large subset of contents (the long tail) attracting a low number of downloads. Current caching and prefetching algorithms deployed in content delivery infrastructures (e.g., CDNs) use prediction techniques that have been proven very efficient for popular content. However, these techniques are not well suited for content placement for the long tail of the popularity distribution. Furthermore, in the recent years, the proliferation of User Generated Content (UGC) platforms such as Youtube as well as the creation of new channels to distribute content such as the Online Social Networks (OSNs) have increased the volume of mid/low popular content (i.e., the long tail of the distribution) and thus the impact of such content in the operational costs of different players such as Content Providers, CDN operators or ISPs.

In this context it is important to develop new algorithms and solutions to enhance current content delivery infrastructures for the optimized distribution of mid popular and/or low popular content. The most critical aspect in the design of the aforementioned algorithms is to properly predict which users are going to consume which content and where these users are located in the network. This data will serve as input to caching and prefetching algorithms, which decide in which servers (or datacenters) of the content delivery architecture mid and/or low popular content is stored. In this demonstrator we implement the solutions designed in WP4, which leverage social information to predict the most likely location(s) where a content distributed through OSNs is expected to be consumed. These techniques will be implemented in an emulation environment that will serve as a proof of concept and to evaluate the enhanced performance introduced by such solutions.

1.1.2 Description

Note that this demonstrator is based on the Use Case 1 named Enhanced Content Placement Using Users' Social and Coarse-Grain Location Information and described in the Deliverables D2.1 [D2.1] and D2.2 [D2.2]. Furthermore, the different algorithms and mechanisms implemented in this demonstrator are developed within the WP4, an initial description of which can be found in the Deliverable 4.1 [D4.1] and a full description in the Deliverable 4.2 [D4.2]. As part of the work developed in the WP4, the aforementioned use case and the developed algorithms have been extensively evaluated through data-driven simulations using real data collected from Twitter.

In WP6, we will complement the large-scale simulation-based evaluation conducted in WP4 with the evaluation of the implementation of some of the designed algorithms in an emulation environment. In particular, we will simulate a set of users that are connected through friend-follower connections effectively mimicking the relation model found in several OSNs such as Twitter or Google+. Note that the social graph formed by these users will follow the properties of real-world social graphs collected as part of the work developed in WP3. In our emulation, different users will be located in different topological (ISPs) locations, for which the emulation environment will consider a topological infrastructure formed by different ASes and ISPs. Furthermore, the demonstrator will emulate a distributed content delivery infrastructure with servers distributed across the different ASes.

In this scenario users will produce posts (e.g., tweets) with an associated content (e.g., a video). The content posted by a user will be consumed by a subset of the followers of the user. The posting rate and consumption rate emulated will correspond to realistic values obtained from our traces collected in the WP3 and data reported in the literature.

We will study the use of different emulation platforms (e.g., Modelnet or PlanetLab). After carefully considering the pros and cons of each one of them we will chose one to develop our demonstrator.

We will run different content delivery mechanisms on the chosen emulation environment: we will compare the performance of traditional caching and prefetching algorithms used by current CDNs with the algorithms designed in WP4 that leverage social graph information and user location. Using these social-enhanced algorithms we will emulate pull-based and push-based solutions.

1.1.3 Functionalities

In this section we describe the functionalities of the eCOUSIN infrastructure defined in the Deliverable 2.1 that will be implemented in this demonstrator. In particular we will address: Social and Content Information Collection, Data analysis, Mining and Aggregation, Social Prediction and Strategies for Content Placement.

1.1.3.1 Social and Content Information Collection

The tools for the collection of social information have been developed as part of the work in WP3. While these modules will not be fully integrated in this demonstrator they will be used to extract the properties of real-world social graphs and content posting rates in OSNs. This information will be used as input to run the social-enhanced content placement mechanisms designed in WP4.

The demonstrator will implement a module for obtaining content included in user posts/tweets, i.e., content inserted in posts or a links to external content.

Therefore, this demonstrator along with the different tools developed in WP3 will help to demonstrate to what extend it is possible to collect social and content information within the current Internet and Online Social Networks configurations.

1.1.3.2 Data Analysis, Mining and Aggregation

A fundamental function of our demonstrator relies on data processing and aggregation. Using this demonstrator we will exploit our work in WP4 and describe how to properly process the social graph as well as the different content information in order to provide useful information to the development of social enhanced content placement solutions. In particular there are two main tasks to be deployed as part of this function: 1) computation of the socio-geographical relationships for each user in order to derive the social graph and the geographical distribution of users, 2) parsing of the information including in the posts. We are specifically interested in identifying posts that include content such as photos and videos or a link to external sources of content (e.g., an external video).

1.1.3.3 Social Prediction

The main goal of this demonstrator is to show how the social information available in OSNs is useful for predicting where long-tail content distributed through OSNs is going to be consumed. To this end, we leverage the social graph and the location information derived from the Analysis, Mining and Aggregation function in order to predict where a content published by a user is likely to be consumed.

1.1.3.4 Strategies for Content Placement

Finally, we will implement social-enhanced content placement strategies for the delivery of non-popular content shared by users through OSNs. In particular we will implement the strategies designed in WP4 including social-enhanced pull-based caching solutions in which content is pulled from an origin server and stored in a local cache server after the first local request depending on the number of social links of the uploader of the content within that specific location. In contrast, traditional pull-based caching algorithms replicate the content in the local cache server after the first request without taking any further consideration (e.g., estimating the presence of other potentially interested users). Furthermore, we will study social-enhanced push-based prefetching techniques in which content is pushed from the origin server to those replica servers that serve a sufficient large number of followers of the content uploader.

We will analyze the performance of the different content placement algorithms in the emulation environment designed for this demonstrator.

1.2 Personal Sharing Clouds

1.2.1 Motivation

OSN users have been dramatically increasing in the last few years to a point where they even attracted new user to the Internet. As the amount of content and information shared by the users of OSNs is ever increasing, so are their concerns regarding privacy and data ownership. Where is content stored? Who is the content owner? How can my content be used? By clearly answering these questions Federated Social Network and Mobile Federated Social Network solutions aim at offering alternative platforms to current OSNs. The basic idea is to separate user identity and relationship from the kind of service they belong to: instead of having different user accounts on different online services, the aim of this proposal is to leverage on a unique user identity, for instance the phone number.

At the time of writing, the attention of public institutions and private companies about privacy issues and right-to-be-forgotten has been growing more and more, together with data portability issues. Most of OSN services ask users to give ownership over their data to the service provider, which ties the user to the service provider. The latter usually don't provide for any constraint on data portability.

1.2.2 Description

The final goal of this use case is providing a content sharing ecosystem where media centres and/or enabled (mobile) devices will communicate and share multimedia resources at different levels (defining Access Control Lists), using different technologies (such as UpNP and DLNA) and exploring a content sharing model based on social relationship.

A solution based on web standards for communications between providers, allowing a distributed architecture, solving data portability and data ownership issues, is a good way to address these issues and highlight the need for and the advantages of such a federated architecture.

The final implementation of the use case on Personal Sharing clouds is going to closely follow the description in the deliverables of WP2. While the federated social networks have been implemented in a very first phase of WP6, other modules of the use case have been implemented in a preliminary prototype covering the functionalities described in the previous work packages.

The Federated social network part has been implemented and tested as described in deliverable D6.1. In this deliverable we describe the steps we have started taking towards the final implementation.

1.2.3 Functionalities

1.2.3.1 Social Data Collector/Data Analysis, Aggregation and Mining

The Social Data Collector will be responsible for periodically monitoring the social networks related to social entities and their modifications. A unique user, e.g. Alice, should be able to link its social network accounts (Facebook, Twitter or compliant federated social network account) to the media centre. Automatic discovery of user linked to Alice using the same app on their media centres should be performed.

1.2.3.2 Content Lookup/Content Copy Selection/Content Dissemination

Content lookup is responsible for periodically discovering 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 such as, for example, File Sharing and UPnP Proxy. The former one supports a simple form of resource sharing; available content is provided as a simple list of shared files. The latter one supports the communication among UPnP devices deployed in different subnets at multi-hop distance; the content is provided to the set of enabled UPnP devices and services based on the relationship among content owners. Once the users are linked they will be able to browse remote resources as local.

1.3 Information-Centric Networking

1.3.1 Motivation

As multimedia services are becoming pervasive, current content distribution technologies, mainly based on content caching at the network edges, may rapidly become too inefficient and ineffective for guaranteeing scale, performance and flexibility needed to deliver content people want, when and where they want it. As a matter of fact, all the current techniques for content placement (where to place surrogate servers and which surrogates will replicate which objects) generally rely on content popularity statistics. However, such statistics are often difficult to estimate for new disseminated content. Media content actually ranges from very popular content (i.e. popular TV Shows, buzz videos, etc.) to user-generated content (UGC) whose audience is mainly limited to the social environment of the content owner. This means that a small portion of available content enjoys a high popularity while the majority is actually receiving few downloads/views, generating the so-called long tail distribution. Therefore, the proliferation of personal multimedia, and especially video, leveraged by the trivialization of OSN usage in everyday life, will most likely increase the heaviness of the long tail. In this context, there will be a large number of requests that are of interest to only small groups of users linked together by some social relationship.

Information-Centric Networking is considered one promising approach to close the gap between the growing content-centric Internet usage and its architecture based on the conversation model. As such, ICN supports network infrastructure evolution with named data chunks instead of IP addresses, by providing primitives to:

1. name any piece of content , independently on its location in the network;
2. access named content objects - not hosts;
3. using a name-based forwarding plane, route a user request identified by a destination content-name, towards the closest copy of the content (either original server or cache servers);
4. deliver the content back to the requester with native in-network caching along the reverse network path.

The ICN paradigm is based on a pull model in which users send interest messages for content and the data is delivered back if available. This model is related to the concept of OSNs which involve people expressing an interest for a given person or for specific content shared by other users (the so-called followers in Twitter).

As described in the deliverable D2.1 of the project, we propose a use case for deploying a social network, say Twitter, over ICN architecture. We propose to evaluate the benefits of using ICN as a network layer for efficiently delivering UGC disseminated in the OSN while also taking into account the relevant interactions inside the social graph between users. To this end, we will implement a demonstrator to illustrate the use case and also to serve as a basis for consistent assessment of our proposal of using ICN as a social-assisted and content-aware delivery layer for OSNs.

1.3.2 Description

1.3.2.1 Implementation of the OSN Application and the NDN Networking

The demonstrator addresses the use case Information-Centric and Social-Driven Content Delivery, which is described in the deliverable D2.1. The scenario depicts the live broadcast of user generated content (UGC) over an OSN such as Twitter. In particular, users can stream videos live to their friends, and store the videos in the cloud so their friends can watch them later.

As exposed in the deliverable D3.1 of the WP3, user locality plays an important role in social networking applications: people are very frequently connected to other people that are in the same town or region, in short in close vicinity (e.g. tweets are mostly distributed locally to local followers, users often send their tweets from the same location, etc.). This means Twitter messages exchanged between end-users are mainly addressed to local or close users, except for very popular accounts (e.g. a Twitter account having millions of followers). Having analyzed the Twitter networking workflow from traffic monitoring, we observe that the locality of end-users is not aligned with the Twitter network behaviour, which always incurs transfer of messages towards remote centralized servers, even if the real destinations of the messages are very close. The network architecture could be then optimized so as to better reflect the end-user behaviour. We propose therefore using Name Data Networking (NDN) architecture in the demonstrator to efficiently delivering UGC over the social network Twitter since NDN is the most advanced project in the ICN space.

The proposed NDN architecture for the content-centric and social-driven delivery of UGC in Twitter, as well as the networking call flow, are detailed in the deliverable D5.1 of the WP5 [D5.1] and will be implemented in the demonstrator. We suggest in particular using routing strategies based on local and non–local users. More precisely, for the publication of tweets, announcement messages are always sent to the Twitter servers for both local and non-local end-users. For the routing of the requests for the tweets, we make a distinction between local and non-local users:

- for non-local users: requests are sent towards the Twitter server (as current Twitter);
- for local users: requests are sent towards the followed end-user himself (who has previously announced his name-based route in the network accordingly).

At the implementation level, the demonstrator consists of two parts:

1. We will first implement a Twitter-like social network with video live streaming support;
2. To convey the networking interactions of our Twitter-like application over our proposed ICN-based architecture, we will modify the open-source NDNx[2] implementation of the NDN protocol to integrate our proposed content naming scheme and our locality-aware content routing strategies.

1.3.2.2 Extension with Dynamic Routing Configuration

The envisioned extension for the demonstrator is to add dynamic configuration of routing tables based on cached contents. The main idea is to dynamically configure the NDN network to route requests towards nodes having the searched contents.

[2] NDNx implementation: http://named-data.net//

As background we assume that some OSN-related content has previously been published and is stored in multiple caching nodes. These caching nodes keep track of cached content items, e.g., by employing a metafile format such as (<ContentID><IP><URI>).

As a starting point, we assume that any new content is also cached and the information on recently performed modifications on cached content is made available to the data management plane [D2.2]. As an example, a lightweight daemon could be running on the caching nodes sending a corresponding registration message to the control plane on modifications of cached content (e.g. about eviction from the cache due to decreasing local popularity).

The information contained in these messages are received by a logical controller and used as input information to a controller application performing re-calculations of the actual routing information based on caching information updates. The controller application makes use of the received messages to make updates to the forwarding information base (FIB) of the forwarding nodes located in the data plane. Finally, the controller pushes updated routing information from the controller application into the forwarding tables of the forwarding nodes.

We currently consider two options for pushing the routing information updates:

Option A - The routing information can either be used to make modifications to the FIBs of NDNx nodes running in the Orange testbed to support the scenario of using content-centric routing strategies for Twitter with automatic configuration of NDNx nodes instead of using hard-wiring. Updates to caching could be tracked by deploying daemons which send registration messages to the control plane.

Option B - Alternatively, we can use the routing information to make modifications to the forwarding/service tables of forwarding nodes deployed in an ALUD local testbed, in which caching node functionalities and caching node update events can be emulated in addition.

1.3.3 Orange Testbed

The demonstrator will be run in a testbed hosted inside an Orange specific network dedicated to perform functional tests. As this specific network is also used by Orange to assess its own services, there is unfortunately no remote access due to security and privacy issues. So we will mainly use the testbed for implementing and evaluating the proposed social-enhanced ICN architecture in a real network environment. A video of the system and of the demonstration will be prepared for presentation during the EC reviews as well as at any other dissemination event.

Figure 1 below depicts the testbed we envision for this use case. The testbed will include different regions (1, 2, 3 and 4) to illustrate the different locality-aware behaviours as identified in WP3. We will target a representative environment for the testbed with a real network structure (access, aggregation, core and interconnection). The access network will be in ADSL, FTTH or Ethernet. We will use a Linux PC for the Twitter server, and the Twitter clients will run on laptops, smartphones and tablets. The possible testbed configuration is as follows:

- a Twitter Server will be set up in a remote AS (similarly to the official Twitter server in US): region 4 in Orange's testbed structure shown on Figure 1;
- end-users from 3 different regions to show local (and non-local) activity;
- NDNx routers deployed in edge nodes;
- an ICN controller.

For obvious lack of a native content-centric network layer, i.e. whose forwarding decisions are solely based on named pieces of content (instead of IP addresses as it is the case for the current IP network layer), our ICN-based architecture for delivering content in Twitter will be deployed as an overlay network. Without global change to the current infrastructures, this deployment consists in carrying the NDNx payload over the current legacy IP network layer. The underlying network acquires then the requested content and delivers it to the user. This means that NDNx will not overhaul the existing network pipelines, but it will use them to restructure the way networks manage resources and distribute information. The NDNx routers will be hosted on Linux-based machines.

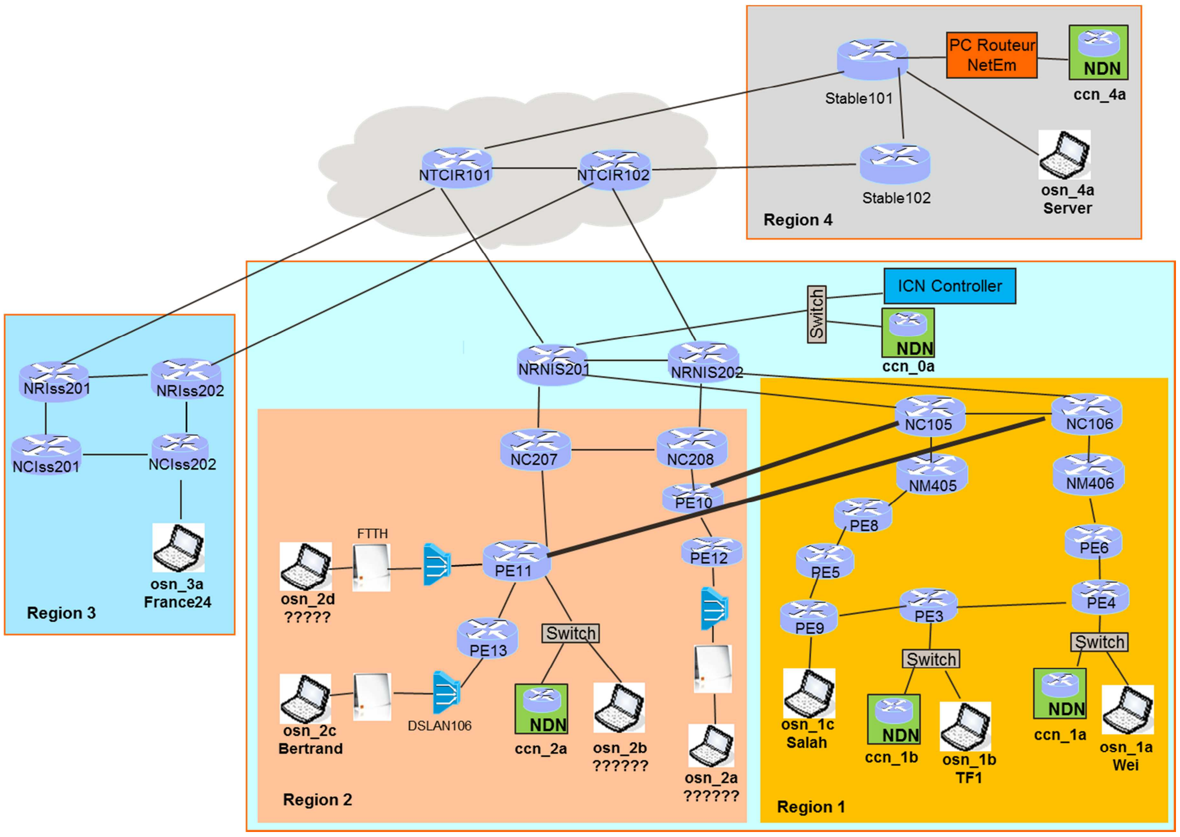

Figure 1 - Orange's testbed structure

1.3.4 Functionalities

Our information-centric and social-driven demonstrator will include the following functionalities of the eCOUSIN architecture defined in the deliverable D2.2 [D2.2].

1.3.4.1 Social-aware Content Naming

The demonstrator will include a social-aware naming scheme for content objects that can be later disseminated or retrieved within the ICN delivery infrastructure. The naming is then defined as follows:

- /Twitter/Local/UsersAAA/Tweet_BBB.

1.3.4.2 Content Dissemination Algorithms

The content dissemination layer provides capabilities to the demonstrator for delivering content to OSN end-users and includes a media streaming transport protocol over the NDN network layer so as to allow end-users to share and consume videos in live streaming. The demonstrator will show an improved delivery based on expected consumption of generated contents.

1.3.4.3 In-Network Content Routing and Caching

The ICN architecture that is implemented in the demonstrator for social-driven content delivery provides the following networking functionalities:

- Name-based forwarding plane based on our locality-aware naming scheme with an in-path resolution of content names into locators (hosts storing content) and a native integration of content caching functions;
- Traffic engineering for efficiently delivering content, with routing management and cache strategies to meet the time requirement of live content streaming. It can exploit social information related to user relationships, user locations, and user interests, etc.

1.3.4.4 Cached Content Update Message (Network Monitoring)

A daemon monitoring a caching node sends an update information messages to a controller. The implementation of the data path infrastructure and the caching systems is out-of-scope of this demonstration. Thus, we will resort to emulating the update functionality by implementing a light-weight daemon in caching nodes. The daemon tracks a local metafile for changes where the metafile would simulate, e.g., a cache index. Upon detecting changes in the metafile, the daemon sends an update message carrying information for newly added (or removed) content including at least the content identifier, the content URI, and the IP address of the node. The update is received by the ICN controller.

1.3.4.5 Content-Centric Routing Information Update (Network Configuration)

When the controller receives an update message, it re-calculates routes using topological and OSN information. If a newly calculated routing requires modifications to one or more forwarding elements, the controller communicates the changes to the controller clients running on the impacted forwarding nodes. A client on a forwarding node receives the instruction from the controller and manipulates the FIB tables of the forwarding nodes accordingly.

1.4 Content Offloading for Mobile Networks

1.4.1 Motivation

This demonstrator addresses the use cases defined in [D2.1, Section 3]: in particular the social-assisted time-unconstrained content delivery and the mobile content uploading scenario will be evaluated in this testbed.

The main idea of the testbed is to show the user-provider experience improvement obtained with the eCOUSIN solutions from two different angles: on the one hand we will visualize the benefit brought by the technical contributions of eCOUSIN on the network performance by means of simulations, controlled experiments, and user traces; on the other hand, the user experience will be evaluated through mobile applications on smartphones using the prefetching and bandwidth optimization functionalities.

1.4.2 Description

For the social-enhanced time-unconstrained content delivery use case, the technical solutions developed in WP4 and WP5 of eCOUSIN have the overall goal to reduce the load on the mobile cellular network and at the same time improve the user experience. This can be done by shifting a part of the data to be transferred to other access technologies, such as WiFi, e.g. when the user is at home. This is especially interesting for data-intensive content that is not time-critical and, thus, can be prefetched at the user device. By using a prefetching mechanism that is running directly at a mobile device, the use of energy and monetary costly mobile connections is expected to be reduced. This is especially important since, mobile data subscriptions are usually costly or limited in their monthly transfer volume.

Prefetching efficacy highly depends on the possibility to predict the data that, with a high probability, is going to be transferred in the future and, consequently, can be prefetched. Prediction can happen at different time scales: short-term to optimize the bandwidth usage of the mobile connection (e.g., to smooth traffic peaks in mobile traffic and to reduce the load of congested cells) or long-term to proactively load content items that the user might access in the course of a day. For the latter case, the goal is to show that an accurate prediction of access, for example, to video clips on YouTube can happen by using information from OSNs. For the former case, even more detailed information on the current activities of a user is likely to be needed.

The mobile prefetching related part of the demonstrator aims to show how an integrated mobile prefetching application could be implemented. Thereby the developed mobile application is meant as proof-of-concept implementation of the social-enhanced prefetching mechanisms designed in the context of the work of WP4 and WP5 of the eCOUSIN project. The target platform for the mobile prefetching demonstrator is Android. The assessment of the solution is planned to be done using collected usage traces of real-world users and through actual real-world trials, where users will be asked to test the application and provide feedbacks on the effectiveness of the solution. Furthermore, parts of the functionalities and findings related to the impact on the mobile network are planned to be visualized for demonstration purposes using graphical user or management interfaces. In this deliverable, an overview on the planned functionalities and modules of the application is provided. In addition, first module implementations are reported in more details.

For what concerns the mobility aware content delivery scenario, the demonstrator objective is to show the impact of technical solutions developed in WP4 and WP5, such as bandwidth usage optimization when future knowledge about channel condition is available. The demonstrator will address scenarios whose durations vary from tens of seconds to a few minutes or an hour.

The impact on the mobile network will be investigated by means of thorough ns-3 simulations performed on scenarios where bandwidth prediction is likely to be possible with good accuracy: for instance, commuter paths from home to work by car or using public transportation, or when user movements are easily understandable and predictable.

In order to show the outcomes both offline aggregate results and online simulation, animation will be realized in order to give both an overall evaluation and a runtime visualization of the benefits brought by the eCOUSIN solutions.

As a result, the demonstrator will evaluate the effectiveness of the eCOUSIN technical solutions in two parallel ways. The improvement on the user experience will be measured thanks to the development of an actual mobile application that will be tested on real-world traces and through actual trials. For what concerns overall network performance a graphical visualization tool will be used to show the network dynamics while the eCOUSIN solutions are applied.

In this way, the demonstrator can show the impact of the developed solutions on the two sides of the consumer-provider field at once, thus making clear what are the pros and the cons of applying the project results in real environments.

1.4.3 Functionalities

The described components and their functionalities are to be seen in close relation with the modules and functionalities as presented as part of the initial system architecture of eCOUSIN in [D2.2, Section 5]. For the demonstrator of the time-unconstraint content delivery (long-term prefetching) to mobile devices, three different functionalities are defined: Social data collection and analysis, Content access prediction, and the integrated Prefetching of content items. Their exact mapping to the modules in [D2.2] is described in Section 3.4. In particular, the following modules are involved for the presented functionalities: Social Data Collector, Data Analysis, Mining, and Aggregation, Social Predictor, Prefetching, and Network Monitoring. In a similar way Bandwidth Availability Prediction, Bandwidth Allocation Optimization and Network Visualization functionalities will be realized for the mobility-aware content delivery scenario (short-medium term prefetching). Finally, the demonstrator will showcase the interaction of all the functionalities in the overall scenario.

1.4.3.1 Social Data Collection And Analysis

Collecting and analyzing social information is a key functionality of the demonstrator and a prerequisite for all further functionalities of the mobile prefetching demonstrator. This functionality is going to be implemented in a Social Data Collector module that is described below. For further processing, the collected data has to be analysed and aggregated to perform content access predictions and prefetching decisions. This processing happens as part of the Data Aggregator of the prefetching application and thereby conceptually as part of the Data Analysis, Mining, and Aggregation module of eCOUSIN. To ease the analysis of the data, visualization functionality for some key properties of the collected and aggregated data is going to be provided as well. Furthermore, extended data collection and analysis functionalities are required to collect usage traces of individual OSN users for later evaluation and assessment purposes. This functionality, therefore, is also going to be part of the Data Analysis, Mining, and Aggregation module.

As the envisioned prefetching application is planned to be run by an individual user on his own mobile device, it is assumed to be able to access the OSN with the content access permissions of this particular user. It can retrieve structural information on the user's one-hop friendship graph, meta-data about the user and its friends, as well as the ongoing interactions of the user with its friends and the interaction history. Besides, the app can access the news feed of the user that aggregates the most recent activities that might be relevant to the user. The application has to access and retrieve a subset of this available information for any further analysis, processing, and aggregation. The vision is that the application can access multiple OSNs that the user actively uses such that content can be prefetched for these different networks together. For the proof-of-concept implementation, the functionality is assumed to be able to access a single OSN, but this could be extended later on.

1.4.3.2 Content Access Prediction

Based on aggregated social information, including structural information on a user's friendship graph, other meta data, and the current news feed, content items have to be determined that a user will access with a high probability in the near future. This functionality is part of the Data Management Plane of eCOUSIN and implemented by the Social Predictor module that is introduced in Section 2.4.4.

Content items that were identified as being relevant for the user are passed on as prefetching candidates to the prefetching mechanisms. The prefetching mechanisms then deal with the actual download of content items. To allow for prioritization in the prefetching process, it would be desirable to assign items numeric importance/relevance values on a yet to define scale.

The prediction functionality itself might consider rather static social properties of content items, e.g. whether they are part of the user's news feed due to group memberships or shared by direct friends. Besides, also rather dynamic properties could be used to decide on an item's relevance, such as the temporal importance of a relationship to a friend. The latter could be determined using the user's interaction history, e.g. how often a user liked/shared/commented posts by a certain friend. In addition, also input from global predictions could be used to influence the identification of prefetching candidates. Global predictions could come from a module of the eCOUSIN Data Management Plane that takes a broader view (in comparison to the limited view of the individual user) on an OSN into account to predict, e.g. content that is about to be viral and, thus, also relevant to the user.

1.4.3.3 Prefetching of Content Items

Based on the identified content items that are prefetching candidates, decisions have to be taken when and how to download/prefetch which of these content items. This functionality is part of the Content Dissemination Plane of eCOUSIN and implemented by the Prefetching module that is introduced in Section 2.4.7.

Some of the items might be more important than others as identified by the Social Predictor. Besides, the time to download the items might be limited by the user's connection to a WiFi network and by the available bandwidth of that connection. The prefetching functionality should account for these characteristics. To plan the downloading process, input from the Network Layer of eCOUSIN and in particular the Network Monitoring module is required on the network availability.

For an extended scenario, it is planned to investigate the potential of exchanging prefetched content items with other clients in proximity using WiFi Adhoc communication and, thus, further offloading traffic from the involved infrastructure-based networks. Here, also input from the Network Monitoring module is required to detect Adhoc partners in proximity of a device.

To round up this part of the demonstrator, the prefetching application requires a way to allow users to access prefetched content items on the mobile device. It would be desirable if this functionality could be integrated into an existing mobile OSN client, such as the Facebook App, in a seamless way. In particular, the app should act as usual but redirect the access to video to the locally stored video rather than, e.g. the Youtube App. How this could be done and, more importantly, how the view on the news feed by the official Facebook App and the Social Data Collector could be synchronized, is yet an open question. For the time being, this functionality is planned to be realized as a Facebook clone application that is under our full control and synchronization can be guaranteed. Due to the many open questions, this part of the demonstrator is going to be reported in a later version of this deliverable.

1.4.3.4 Bandwidth Availability Prediction

The system will predict the bandwidth availability based on statistical prediction on the user mobility and typical bandwidth availability on the predicted path. It shall be possible to provide bandwidth availability estimate from a few seconds up to minutes and tens of minutes in advance. The predictor shall provide a confidence range of the provided estimate. This function will contribute to the realization of the prefetching modules defined in D2.2 (section 5.2.1) through the network monitoring functionality.

1.4.3.5 Bandwidth allocation optimization

Given some future channel condition knowledge, it shall be possible to schedule transmission in order to optimize some objective function. For instance, for the case of real-time contents, the objective is to minimize the re-buffering time and the bandwidth cost (the bandwidth cost is higher when the capacity is lower), while for time unconstrained contents only the bandwidth cost optimization is needed. Again, this functionality is a component of the Network Resource Configuration functionality defined in D2.2.

1.4.3.6 *Network Visualization*

In order to graphically evaluate the network behaviour, simple graph based visualization is needed. For instance, it shall be possible to show link metrics, such as the link bandwidth cost and throughput graphically on the edge of a graph, where edges represent links between users and base stations. In addition, vertices (representing users, base stations and servers) shall provide graphical indicators of their status, such as the buffer state or the actual re-buffering time.

2. MODULES

The eCOUSIN functional architecture, presented in the Deliverable 2.2, is formed by three layers: the Social Layer, the Content Dissemination Layer and the Network Layer. Each of these layers includes different functional modules that constitute the basis of the modules implemented in each one of the demonstrators.

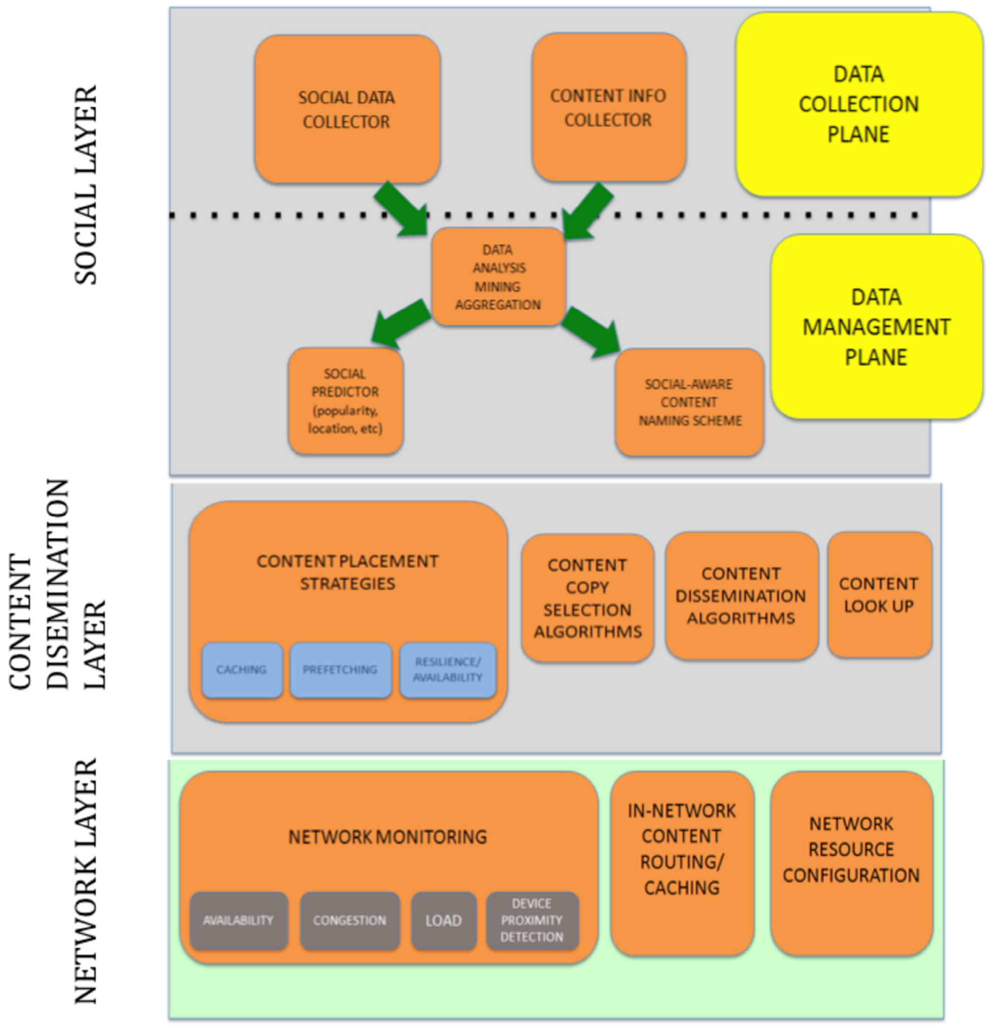
SOCIAL LAYER
SOCIAL DATA COLLECTOR
CONTENT INFO COLLECTOR
DATA COLLECTION PLANE
DATA ANALYSIS MINING AGGREGATION
DATA MANAGEMENT PLANE
SOCIAL PREDICTOR (popularity, location, etc)
SOCIAL-AWARE CONTENT NAMING SCHEME
CONTENT DISEMINATION LAYER
CONTENT PLACEMENT STRATEGIES
CACHING
PREFETCHING
RESILIENCE/ AVAILABILITY
CONTENT COPY SELECTION ALGORITHMS
CONTENT DISSEMINATION ALGORITHMS
CONTENT LOOK UP
NETWORK LAYER
NETWORK MONITORING
AVAILABILITY
CONGESTION
LOAD
DEVICE PROXIMITY DETECTION
IN-NETWORK CONTENT ROUTING/ CACHING
NETWORK RESOURCE CONFIGURATION

Figure 2 – eCOUSIN functional architecture

Next we describe each of the modules implemented by each demonstrator and describe the interdependencies between them.

2.1 Content Placement

2.1.1 Social Data Collector

The social data collector module has been implemented as part of our contribution in WP3 where we have designed and implemented tools that collect data from different OSNs (Twitter, Facebook and Google+) and store the information in a database. There are two main pieces of information of interest for this demonstrator: the user's social graph (that includes the links between this user and other users as well as the nature of that link (friend → follower or followed → friend) and the user's geographical location (if available). This database will be used in order to produce the emulation scenario of this demonstrator including a realistic social graph formed by the users considered in the emulation. Note that the information about the social graph will serve as input to the Data Analysis, Mining and Aggregation module.

2.1.2 Content Info Collector

This module is responsible to parse user posts in order to identify posts that include content (e.g., a video or a photo) to be distributed. Note that the content can be directly embedded in the post or the post can include a link to an external content (e.g., a link to a YouTube video). Furthermore, it will identify the user who uploaded a specific content and will collect other important information about the content such as: type (e.g., video vs. photo), size or location (e.g., external server in the case of YouTube videos).

This module will provide information regarding the content and uploader ids to the Data Analysis, Mining and Aggregation and the content id to the Social Predictor module.

2.1.3 Data Analysis, Mining and Aggregation

This module will receive as input the user-id of the uploader for a given content from the Content Info Collector module. Furthermore it will receive the social-graph of that user from the information obtained from the tools developed in the WP3 that form the Social Data Collector module.

It will process the received information and provide as a result a list of user-ids with their associated properties (i.e., geographical location). These user-ids correspond to the followers of the uploader who are more likely to consume this specific content. This module will provide to the Social Predictor module the uploader's user-id and location, the list of followers and their location and the associated content-id.

Note that this module needs to process information associated to multiple content/uploader pairs in parallel and in real time.

2.1.4 Social Predictor

The social predictor module will receive from the Data Analysis, Mining and Aggregation module the uploader's id and its location and the list of followers with their location and the associated content-id. Furthermore, using this content-id it will retrieve from the Content Info Collector the specific information about the content (e.g., type, size and location). Using this information the Social Predictor will implement the algorithms designed in the WP4 in order to predict the specific locations (and thus their associated cache servers) where the content is more likely to be consumed based on the social and location information of users.

This module will provide input data to the Content Placement module to implement the social enhanced content placement solutions.

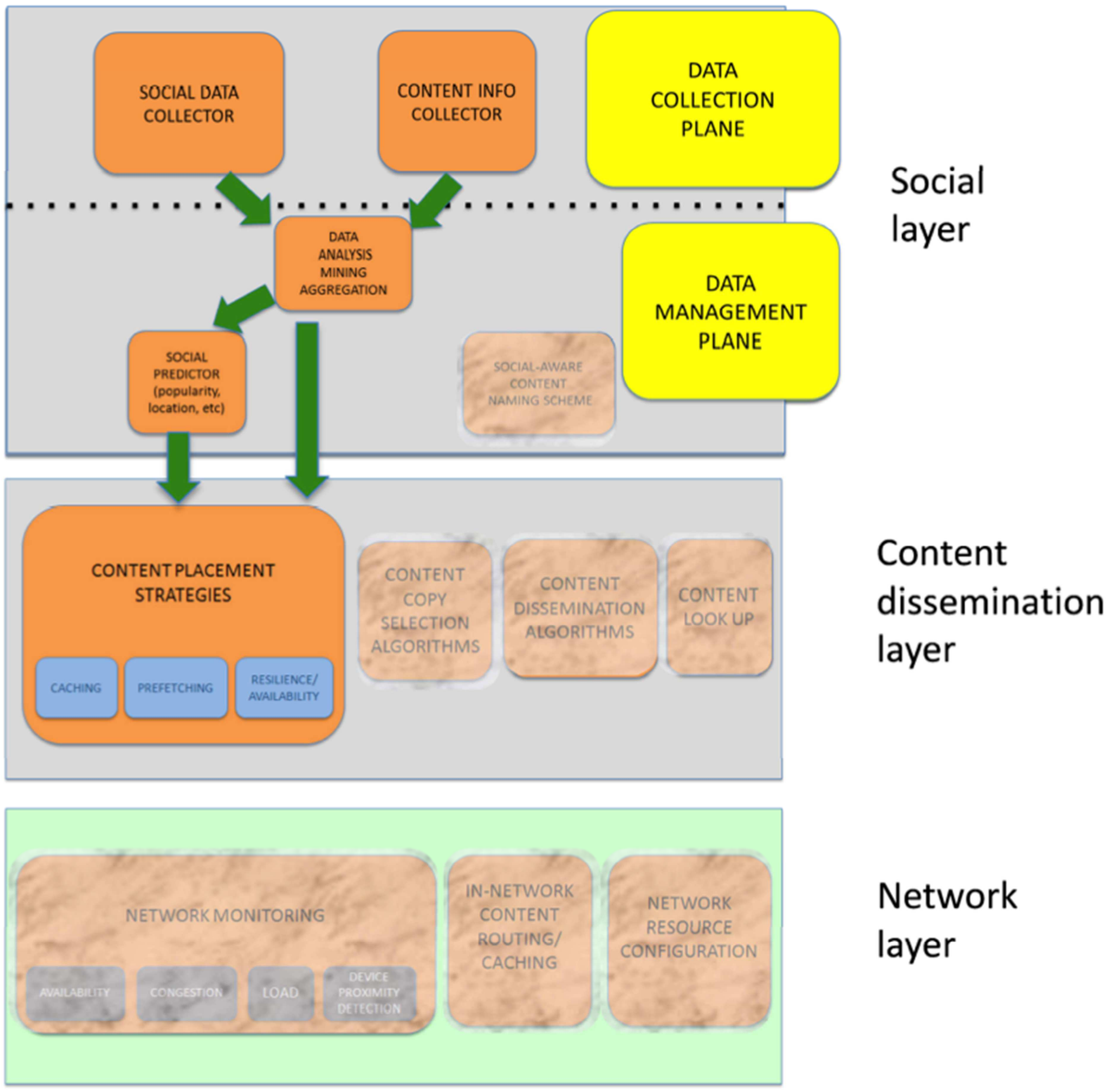
SOCIAL DATA COLLECTOR
CONTENT INFO COLLECTOR
DATA COLLECTION PLANE
DATA ANALYSIS MINING AGGREGATION
DATA MANAGEMENT PLANE
SOCIAL PREDICTOR (popularity, location, etc)
SOCIAL-AWARE CONTENT NAMING SCHEME
Social layer
CONTENT PLACEMENT STRATEGIES
CACHING
PREFETCHING
RESILIENCE/ AVAILABILITY
CONTENT COPY SELECTION ALGORITHMS
CONTENT DISSEMINATION ALGORITHMS
CONTENT LOOK UP
Content dissemination layer
NETWORK MONITORING
AVAILABILITY
CONGESTION
LOAD
DEVICE PROXIMITY DETECTION
IN-NETWORK CONTENT ROUTING/ CACHING
NETWORK RESOURCE CONFIGURATION
Network layer

Figure 3 - Modules implemented in the Content Placement Demonstrator and their interdependencies

2.1.5 Content Placement Strategies

This module is responsible for implementing both traditional content placement and social-enhanced content placement strategies.

Among traditional content placement strategies we will focus special interest to pull-based caching strategies since it is one of the most widely used strategies by current Content Distribution architectures such as CDNs. In this strategy a specific content is cached in a local cache server after the first request from a user located in the geographical area associated to that cache server. We will also consider push-based content placement strategies.

Furthermore, this module will implement social-enhanced content placement strategies that are designed in the WP4. These strategies include pull- and push-based algorithms that extend traditional solutions by taking into account the information provided by the Social Predictor regarding the most likely location where content is expected to be consumed based on the social graph and location information of users.

We will develop a mock-up Internet using a network Emulator. In this emulator we will implement a realistic scenario where different Autonomous Systems (ASs) will be represented as well as the connections and delays between them. Moreover, each one of the represented ASs will use a cache server as a gateway connection with the users.

This emulator will be the base where the users and the services will be connected.

2.2 Personal Sharing Clouds

The Personal Sharing Clouds use case will impact several functional architectural elements at each layer. In the following figure we perform a mapping of the impacted components. Those ones highlighted by continuous lines are the central ones, where the demonstrator aims at adding core functionalities. Those highlighted in dot lines are the ones the demonstrator aims at reusing and possibly side contributing to the functionalities list.

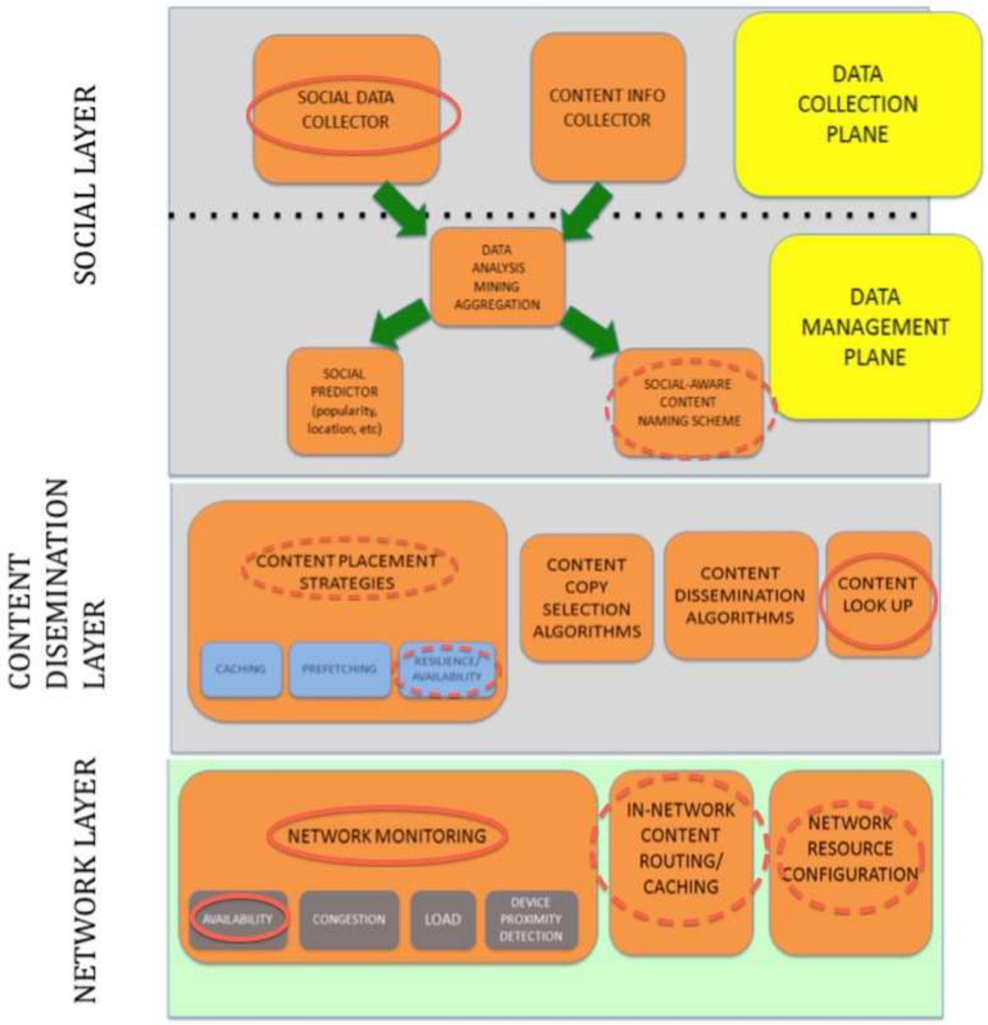
SOCIAL LAYER
SOCIAL DATA COLLECTOR
CONTENT INFO COLLECTOR
DATA COLLECTION PLANE
DATA ANALYSIS MINING AGGREGATION
DATA MANAGEMENT PLANE
SOCIAL PREDICTOR (popularity, location, etc)
SOCIAL-AWARE CONTENT NAMING SCHEME
CONTENT DISEMINATION LAYER
CONTENT PLACEMENT STRATEGIES
CACHING
PREFETCHING
RESILIENCE/ AVAILABILITY
CONTENT COPY SELECTION ALGORITHMS
CONTENT DISSEMINATION ALGORITHMS
CONTENT LOOK UP
NETWORK LAYER
NETWORK MONITORING
AVAILABILITY
CONGESTION
LOAD
DEVICE PROXIMITY DETECTION
IN-NETWORK CONTENT ROUTING/ CACHING
NETWORK RESOURCE CONFIGURATION

Figure 4 - Modules implemented in the Personal Sharing Clouds Demonstrator.

The overall use case is described by the Figure 5. The main idea of the use case is to link social identities to media centres, in order to create a real network among them, exploiting the user's social relationships. As of Deliverable D6.1 just a small part of the overall demo was implemented, which was the federated social network part. The final objective of this use case is to build a dynamic network leveraging on Online Social Networks as a meeting point. Since the mission of the project is to bring value to open solutions, besides using Twitter and Facebook we decided to implement an opensocial/ostatus protocol stack to leverage on.

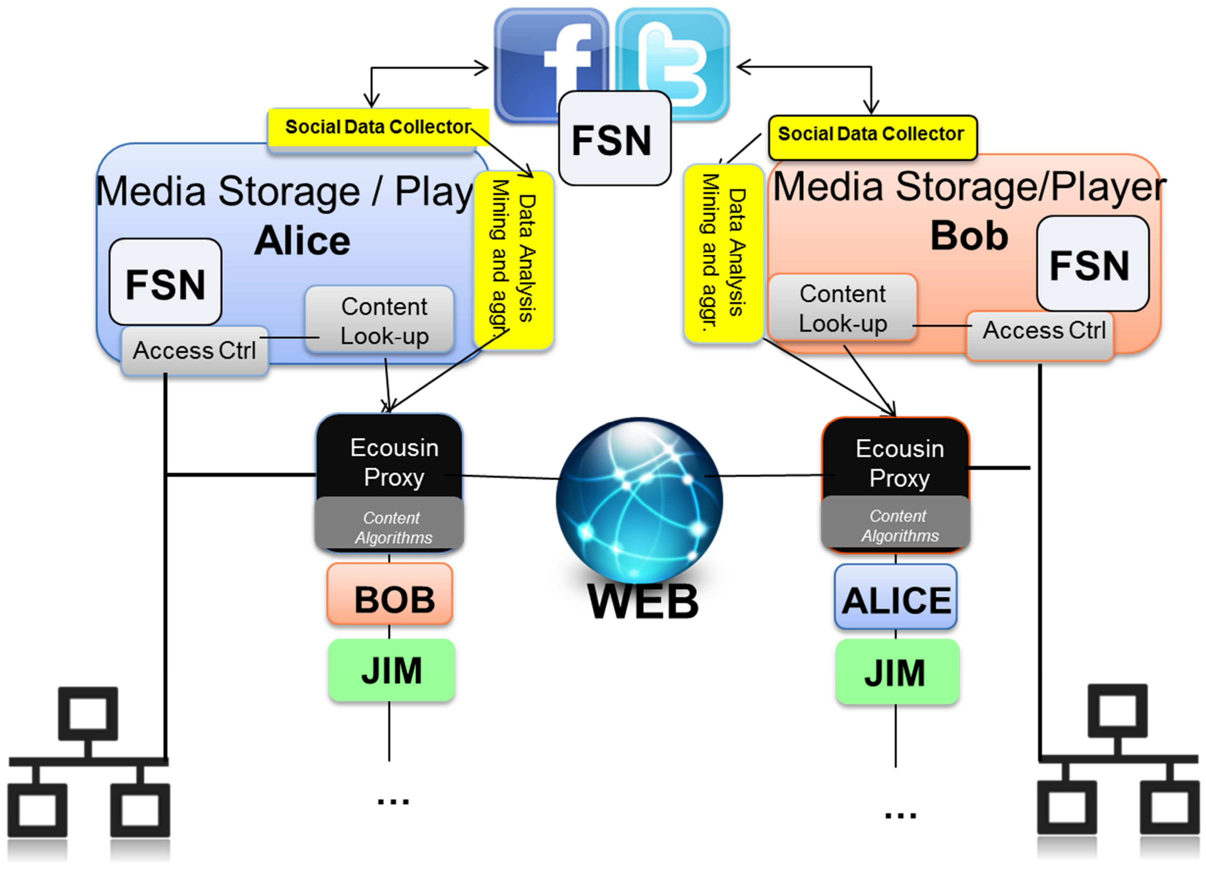

Figure 5 – Overall Personal Sharing Clouds picture

In the next subsections we describe the updates with respect to the status of the modules reported in Deliverable D6.1 [D6.1].

2.2.1 Social Data Collector

The social data collector implementation of D6.1 just took into account the Federated Social Network part, implementing interfaces to link users belonging to different social networks, compliant to the federation standard. Interfaces and data exchanges have been already described there.

The Social data collector is responsible for exploring and keeping updated the social relationships among users, which will be used to build a social based peer-to-peer network among users with the application installed on their media centre. This module acts as a social observer: when a user links his/her account to a media centre, it grabs online and federated social network data in order to understand who in the user's social graph is willing to be linked to the network.

Social information is then processed and passed to the Data Analysis and Mining module which will be responsible of actually setting up the network resources.

2.2.2 Data Analysis and Mining

The Data Analysis, and Mining module will be responsible for actually controlling the merging of the local sharing nodes with remote ones.

The information provided by the Social Data Collector is passed to this module and specifically it will know:

- If and how users having this app installed on their media centres are linked among them;
- The type of social relationship between them;
- A unique way to reach each Social Merger endpoint remotely.

The local Data Analysis and Mining module interacts with remote Data Analysis, Mining and Aggregation modules to generate a symmetric key and provide one another the references to be connected (e.g. public IP address and ports). By means of the key and the type of social relationship each sharing node knows which filtering rules should be applied on packets coming from the Internet.

The Data Analysis and Mining module forwards the required information (including the IP addresses, ports, key and filtering rules) to the eCOUSIN proxies, which are actually responsible of giving access/move/place content.

The Data Analysis and Mining module is orchestrating the way in which the system puts the different users in contact. This is the point in which the user's social actions start influencing the communication network; this is the point in which having a clear picture of social relationships becomes a requirement.

2.2.3 Content Lookup

The personal sharing clouds demonstrator has a content lookup component by introducing a support for Social p2p sharing of contents structured over two layers: sharing services and shared contents. Once the system determines whether a content can be shared to other nodes, the content lookup will be able to determine which service (e.g. UpNP or simple file sharing) is the most appropriate to share it. The basic idea is to decouple the content with the mechanism to share it, this will give the infrastructure the capability to be flexible and extensible thus able to support the unconstrained growth and evolution of sharing mechanisms together with clients embedded in commercial devices. The Content Lookup module acts in this use case as a Resource Locator. It is responsible for periodically discovering:

- Services shared on remote nodes, either performing service discovery requests for known services or directly requiring to remote nodes the list of registered services;
- Content provided by a preselected set of well-known services.

From a user's perspective the content lookup contributes to allow the access to physical resources shared by other users. Once the network resources have been set, based on social information, this layer will help in discovering remote media, acting as an interface to show them as local resources.

2.2.4 Content Copy Selection/Content Dissemination Algorithms

Network resources are configured based on social information in order to reduce traffic load. This module chooses the right delivery method to serve a content to a particular client (e.g. direct delivery or with a peer-assisted video delivery protocol). These algorithms are responsible of placing

the content in the best place, where best means the place maximizing globally the level of user experience.
The Content Dissemination Algorithms module takes into account social information and network traffic/geographic information. Considering a specific content, if we know it will be shared over a social network on a specific territory the content should be placed in the node which can be accessed more easily over the network in that specific territory. Content dissemination algorithms will take into account network/hardware capabilities of nodes.

In the case where a peer-assisted video delivery protocol is used to help offloading the end users' sharing nodes, this module selects the most adequate node among the different options (sharing node or peer-assisting server) to serve the requested content, once again to place the content so to maximize the overall level of user experience.

2.3 Information-Centric Networking

This section provides a high-level view of the interactions between the functional modules of the eCOUSIN architecture (as defined in the deliverable D2.2) that are involved in the information-centric and social-driven demonstrator, as depicted in Figure 6.

2.3.1 Social-Aware Content Naming Scheme

The social-aware content naming scheme developed in our demonstrator allows managing and addressing pieces of content by names independently from their location. Our social-driven solution is based on the differentiation of end-users depending on the degree of their popularity in the OSN graph. The locality-aware property implies that the OSN application needs to determine when and how to change the status of a user. This can be done with information of the social graph of the user from the module Data Analysis and Mining in the Social Layer of the eCOUSIN architecture. A reconfiguration announcement in the network for the adaptive popularity change needs then to be defined for the user and the history of some past messages could be imported in this new profile for being remotely accessed.

2.3.2 NDN Content Dissemination Algorithms

The real-time media streaming case study is put forward in the demonstrator for proving the validity and testing the performance of the proposed content-centric and social-driven delivery layer in a realistic ISP application context, presenting stringent QoS/QoE requirements. To support the timely delivery requirements of media streaming services over the NDN architecture, the following model will be developed in the demonstrator. Media files (i.e. videos) are decomposed into identifiable and named chunks according to our social-driven naming scheme, and then a suitable transport protocol over NDN requests the fetching of content data at the granularity of chunks, specifying also required delivery time.

The time requirement for live streaming might be turned into priorities for treating the data packets in the network according to the time elapsed and the network conditions. Appropriate deadline-aware link scheduling schemes might then to be in place at network nodes. Those relevant interactions between the modules Network Monitoring and Content Dissemination of the eCOUSIN architecture will not be implemented for the demonstrator.

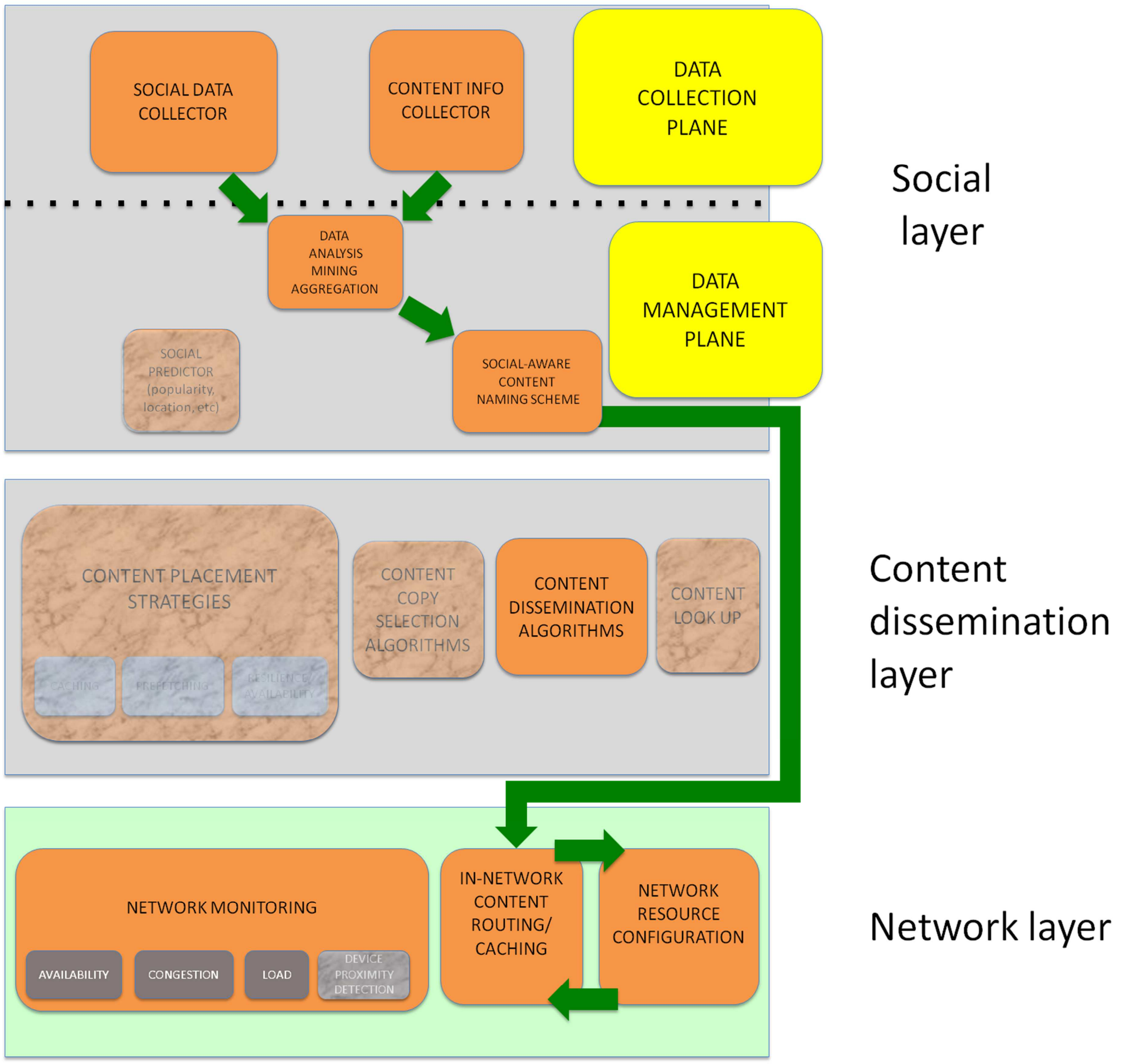
SOCIAL DATA COLLECTOR
CONTENT INFO COLLECTOR
DATA COLLECTION PLANE
DATA ANALYSIS MINING AGGREGATION
DATA MANAGEMENT PLANE
SOCIAL PREDICTOR (popularity, location, etc)
SOCIAL-AWARE CONTENT NAMING SCHEME
Social layer
CONTENT PLACEMENT STRATEGIES
CACHING
PREFETCHING
RESILIENCE/ AVAILABILITY
CONTENT COPY SELECTION ALGORITHMS
CONTENT DISSEMINATION ALGORITHMS
CONTENT LOOK UP
Content dissemination layer
NETWORK MONITORING
AVAILABILITY
CONGESTION
LOAD
DEVICE PROXIMITY DETECTION
IN-NETWORK CONTENT ROUTING/ CACHING
NETWORK RESOURCE CONFIGURATION
Network layer

Figure 6 - Modules implemented in the Information-Centric and Social-Driven Demonstrator and their interdependencies

2.3.3 In-Network Content Routing and Caching

This module will be based on our naming scheme defined at the Data Management Plane of the eCOUSIN architecture. Following the NDN architecture, it will contain functions necessary for the treatment of named content from Data Management Plane. Specifically, it will provide functions for the handling and forwarding of the INTEREST and DATA packets defined in NDN for requesting and retrieving named content.

It will also include control functions for the management and maintenance of the Pending Interest Table (PIT) and Forward Information Base (FIB). The routing of data will follow the rules declared in the FIB module of the NDN nodes. The FIB will be filled via an on-path routing protocol such as OSPFN or LSRN. This will allow end-users to inform about their reachability for contents they offer. Typically, it is mostly valid for local end-users who will provide the content themselves. This FIB configuration is one option. Another one is via the ICN controller, which will have knowledge of the network topology and social network graph and can dynamically control and configure the NDN routers via a configuration protocol (see next sections). The ability to know if an end-user should behave as a local one or a non-local one could be on a declaration-basis but could be also known from the Data Analysis Mining Aggregation module.

Additionally, as pervasive caching is a native feature, NDN nodes are also enhanced with functions for nodal caching and nodal cache control. Cache control encompasses the required intelligence for controlling the caching functions in terms of a) caching decisions at the node level (which requested objects to consider for storing in the cache) and, b) cache replacement policies (what to drop in order to meet the needs of live streaming delivery).

Having also knowledge of cached contents, the controller might dynamically configure the routing tables so that requests are forwarded to appropriate nodes.

2.3.4 Module IN1: Logically Centralized ICN Controller

The ICN controller is part of the eCOUSIN Network Resource Configuration functional block. When the controller receives a cache update message, it re-calculates routes using topological and OSN information. It then forwards the updated information to the ICN controller client in the In-network Routing/Caching functional block accordingly.

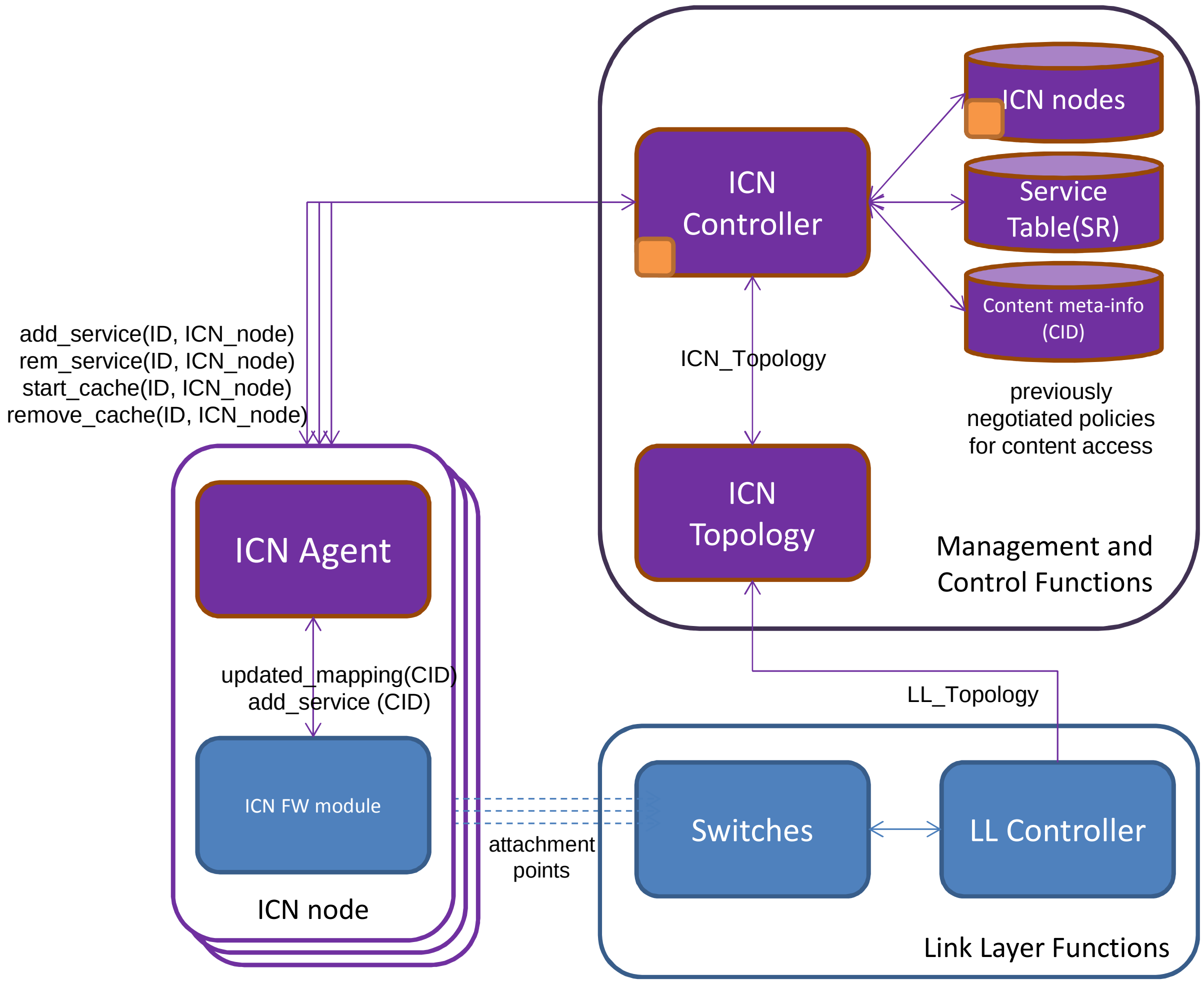

Figure 7 - ICN controller architecture

The ICN controller component manages the mapping of service names to ICN nodes and the launch/removal of caches on facility nodes.

The ICN Controller learns about new mappings from observing the service announcements of the controlled ICN nodes. Moreover, it processes incoming content requests and further information from the topology component to calculate a path across the network from requestor to the appropriate service.

The resulting overlay path is associated with the corresponding resources and their overlay connections, and the corresponding routing information is pushed to the registered ICN agents which in turn perform the required modifications to the forwarding tables of the concerned ICN forwarding nodes.

The service table entries at the controller and on the ICN nodes are provisioned without a lifetime definition. As a side effect of this mechanism of deploying the overlay path the controller is automatically informed about local service de-registration events communicated through the ICN client whenever a service gets removed and can update his service table immediately.

2.3.5 Module IN2: ICN Controller Client

The ICN controller client is part of the In-network Content Routing/Caching functional block and is deployed on ICN forwarding nodes. The ICN controller client is able to communicate out-of-band with the controller IN1 and instructs the forwarding node.

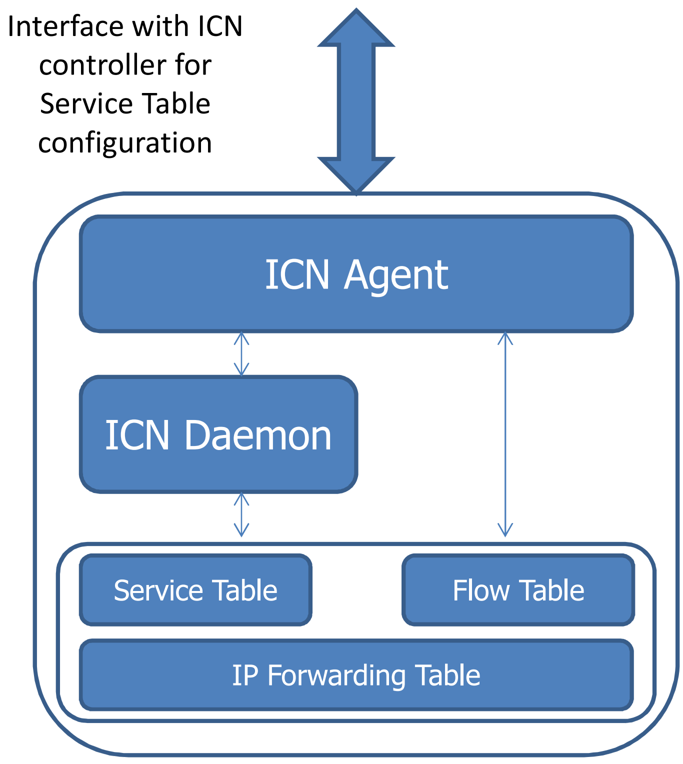
Interface with ICN controller for Service Table configuration
ICN Agent
ICN Daemon
Service Table
Flow Table
IP Forwarding Table

Figure 8 - ICN node configuration

The ICN controller client offers an interface to the centralized ICN controller to enable the configuration of forwarding tables and to receive routing information for adding, modifying and deleting forwarding rules.

The ICN controller client receives instructions from ICN controller in a meta-format, which describe the corresponding modifications independent from the implementation of the forwarding nodes with different implementations. Therefore, the ICN controller client will translate the routing information updates provided by the ICN controller to make the corresponding modifications also to the Forwarding Information Base (FIB) of NDNx based forwarding nodes.

As an option, forwarding table updates related to caching can be tracked by daemons running on the ICN nodes that send back registrations to the ICN controller via the ICN interface (see Figure 8).

2.4 Content Offloading for Mobile Networks

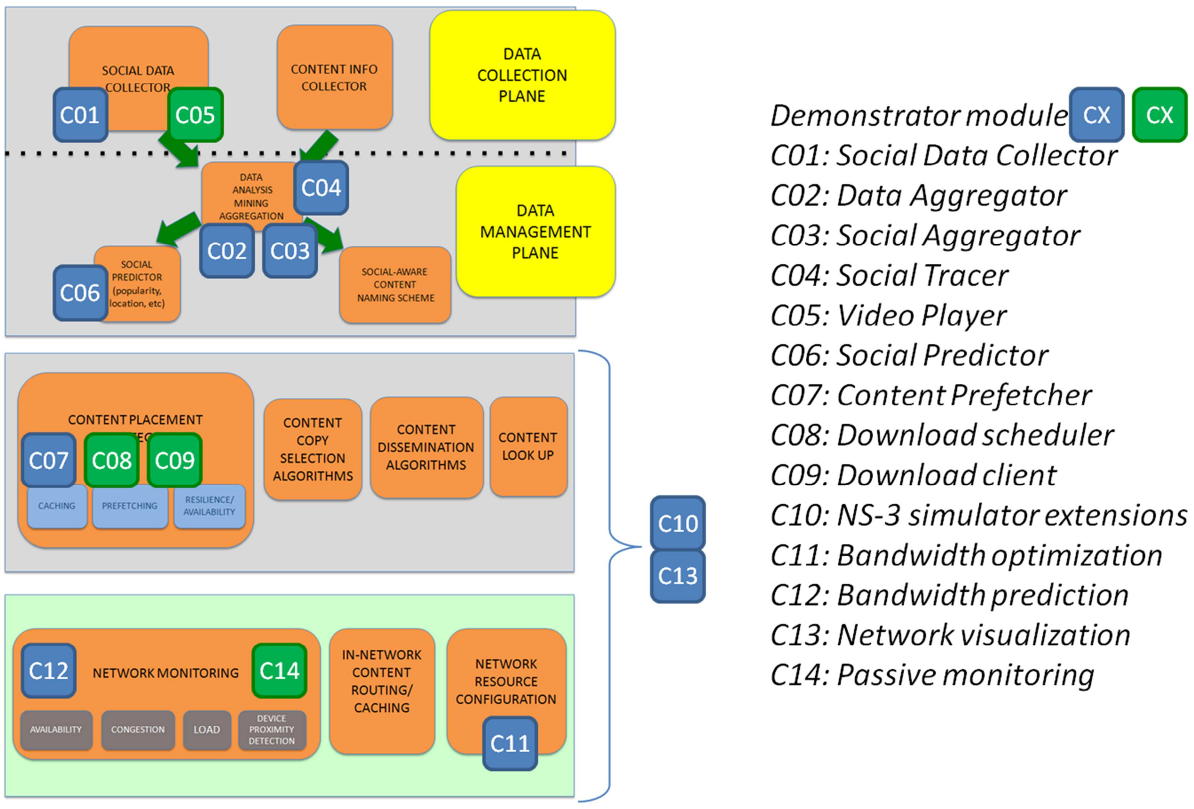

Figure 9 - Mapping between the demonstrator modules and the functional architecture.

The high-level architecture of the mobile prefetching application is shown in Figure 2 while Figure 9 provides details on the mapping between the demonstrator functionalities and the high-level architecture. Note that modules C01-C06 act on the data (collection and management) plane of the architecture, while module C07 provides a hook in the content plane, and modules C11-C12 and C14 provide networking functionalities. Finally, modules C10 and C13 are depicted as external since their role is ancillary to the architecture itself as they aim at simulating and visualizing the results of the system.

From a different perspective, modules C01-C05 and C07 focus on selecting what content to prefetch, while modules C06, C08, C09, C11, C12, and C14 address when and how to prefetch a given content. Modules C10 and C13 are to visualize the results of the other modules.

Finally, the mobile phone application will show the enhanced user experience by leveraging on modules C01-C09, C11, C12, and C14, while the overall network performance improvement will be shown as a result of the interactions among modules C07-C14.

The shown modules are a subset of the modules presented in the overview on the eCOUSIN modules for the time-unconstraint content delivery use case as presented in [D2.2, Section 5.2.1]. In the following subsections, the core modules of this architecture and their implementations are introduced.

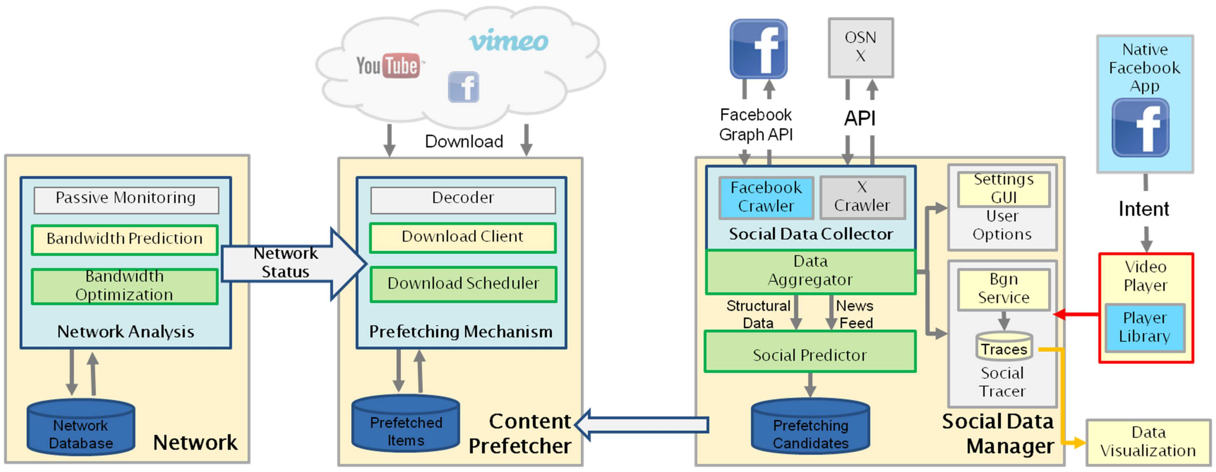

Figure 10 - The core mobile prefetching architecture and its modules

In Figure 10, the overall architecture of the Demonstrator and its modules are presented. We have there high-level components, the Network, the Content Prefetcher and the Social Data Manager (and Collector).

The Network component continuously analyze the network status in order to provide the other components with numerical indicators related to the effectiveness of prefetching at any given time in the future. This component exploits a Network database, where typical network behaviours and how to deal with them are stored.

The Content Prefetcher includes the functionality modules which are necessary to download the content. Here prefetching candidates are scheduled to be downloaded by the Download Scheduler module. This allows to postpone the download until certain conditions are met e.g. a WiFi connection is available. The Download Client module performs the actual download and takes care of connection abortions and also chooses the quality of the video if multiple qualities are available. The decoder service helps the Download Client to find the URI of the mp4 video file. Therefore it gets a video id as an input and outputs a URI or a list of URIs to the mp4 representations of the video hosted at the content provider or a CDN.

The Social Data Collector & Manager continuously monitors the OSNs used, which is at this point in time Facebook only. For every OSN used, a special crawler is used. The Social Data Collector hands the information acquired by the crawler module to the Data Aggregator Module. Here the date is brought into a common representation which is stored in a local database by the Social Tracer module. To determine which items should be prefetched first the Social Predictor module stores the candidates with a priority in the local Prefetching Candidates database, which is used by the Content Prefetcher.

The user of the App is using the native Facebook App. If a YouTube video is accessed the Video Player module is called by intent and plays the video. Additionally the user behaviour observed by the Video Player is send to the Social Tracer module and stored.

2.4.1 Module C01: Social Data Collector

The Social Data Collector module was implemented, so far, to access the OSN Facebook only. Further crawler subcomponents for other OSNs could be added later on.

For the access to Facebook, the Facebook Graph API[3] was used. In particular, two different methods are currently used to access the Graph API: the Facebook Query Language (FQL)[4] and the Facebook SDK for Android[5]. Compared to the previous Version of the Social Data Collector, where RestFB was used, FQL allows for more sophisticated queries which move data processing from the client side to facebooks' side. By using FQL and the Facebook SDK for Android, JSON objects are returned from the APIs that are processed in the Social Data Collector and converted into Java objects. The Social Data Collector module collects structural data and metadata, such as the one-hop friendship graph of a user, its membership in user groups, or its interests. Besides, information on content items that are presented to the user on the so called news feed is retrieved. Using the Social Tracer App, this data can actively be used to trace how a user interacts and consumes content. Furthermore, the history of a user's interaction with content items can be analysed. Here, the information on videos that were watched by the user in the past would be of particular interest for the work on prefetching. While this detailed information is not available as such, it is possible to retrieve all content items that the user interacted with, i.e. liked, re-shared, or commented on in the past. In addition our Video Player module keeps track of fine-grained events while the user is playing YouTube videos. This allows us to gain in-depth information about the user's video consumption behaviour, e.g. how long and when a video was watched as well as if the user paused or skipped the video during the playback.

To access all the previously described data using the Facebook Graph API turned out to be very time consuming if done in a single-threaded way. The reason is the fact that the response times of a single query can be very high. We observed response times in the order of almost two seconds to fetch 25 posts from a sample user's news feed. Due to the required nested requests that are necessary to retrieve, e.g., the likes and comments of all the post, the overall response time can easily add up to several minutes. This is especially a problem if a user front-end is connected to the data collector and waits for input to be rendered for the UI. Facebook monitors the average API response time live on a webpage[6], but does not state its absolute value. The graph suggests that the response time seems to be rather stable and only changing gradually over the day. Together with our observations over two months, we assume that response times in this order are the usual case, not an exception. In the previous version of the Social Data Aggregator, we relied on a highly multi-threaded approach to deal with this fact. It turned out that this greatly helps to reduce the overall time to retrieve the data. However, we figured out a more elegant way to get our hands on the data. By using FQL queries we are able to retrieve the same data in a much faster manner. Limited pre-processing like filtering is possible and helps to reduce processing load on the client. For example the likes of a comment can be requested, as well as the likes that were given from the users friends only.

2.4.2 Module C02: Data Aggregator

The Data Aggregator module uses the inputs from the Social Data Collector module, which are essentially Java representations of the retrieved content items and metadata, and stores them in a local SQLite database. The Data Aggregator module, thereby, can do an initial filtering and aggregation of the data to avoid duplicate entries in the database and combine the input from different OSN crawlers. As for now we only use a single crawler/data collector for Facebook, the current aggregation part is rather simple.

[3] Facebook Graph API: https://developers.facebook.com/docs/graph-api/

[4] FQL: https://developers.facebook.com/docs/reference/fql/

[5] Facebook SDK for Android: https://developers.facebook.com/docs/android/

[6] https://developers.facebook.com/live_status/

The structured and relational representation of the data in the local database eases the work with the data for all further steps, using the build-in features and capabilities of the database engine. The input and output representations and the described interactions with other modules are depicted in Figure 11.

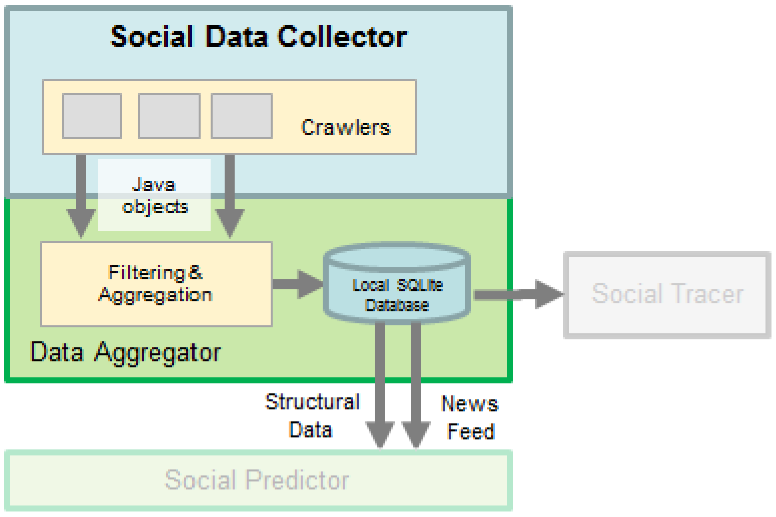

Figure 11 - The Data Aggregator module and its dependencies.

2.4.3 Module C03: Social Aggregator

The Social Aggregator module is a desktop application implemented in Java to allow the analysis of a single users' Facebook feed to study and derive content to be prefetched from the incoming items. The analysis is based on past interactions between the user running the software and his/her friends (the history). For this, the Facebook feed is fetched from Facebook servers to allow a more comprehensive analysis of the users' interactions. This is necessary, as the Facebook API heavily restricts the users in this respect. The collected information is used to analyse and visualize the relations between a user and his friends. Visualizations currently implemented are a user interaction graph and a modification of the FB stream allowing filtering the displayed posts based on the content type.

2.4.3.1 Information Retrieval

As stated before, the implementation makes use of the RestFB library as part of the Social Data Collector module and, thus, uses the Graph API to connect to Facebook. After an initial aggregation and filtering, the retrieved data is available as local SQLite database as output of the Social Aggregator module.

By issuing SQL statements to the database, filtering, sorting and combining table entries becomes possible. Such it is possible to return videos appearing on last week's FB feed, which were commented or liked by friends. These are then candidates likely to be pre-fetched. This selection process can then be improved by including past interactions between the user and his friends to weigh the likeliness of future interactions. This weighting can be based on the number of exchanged messages, the percentage of liked or commented videos posted by this user, or the general number of comments by friends, who watch similar videos.

The Social Aggregator is structured as detailed in Figure 12. All components except the Visualization are written in Java and triggered by a central component. The authentication is implemented as a Java WebView, as it is not easily possible to retrieve the generated access token from the browser without user interaction.

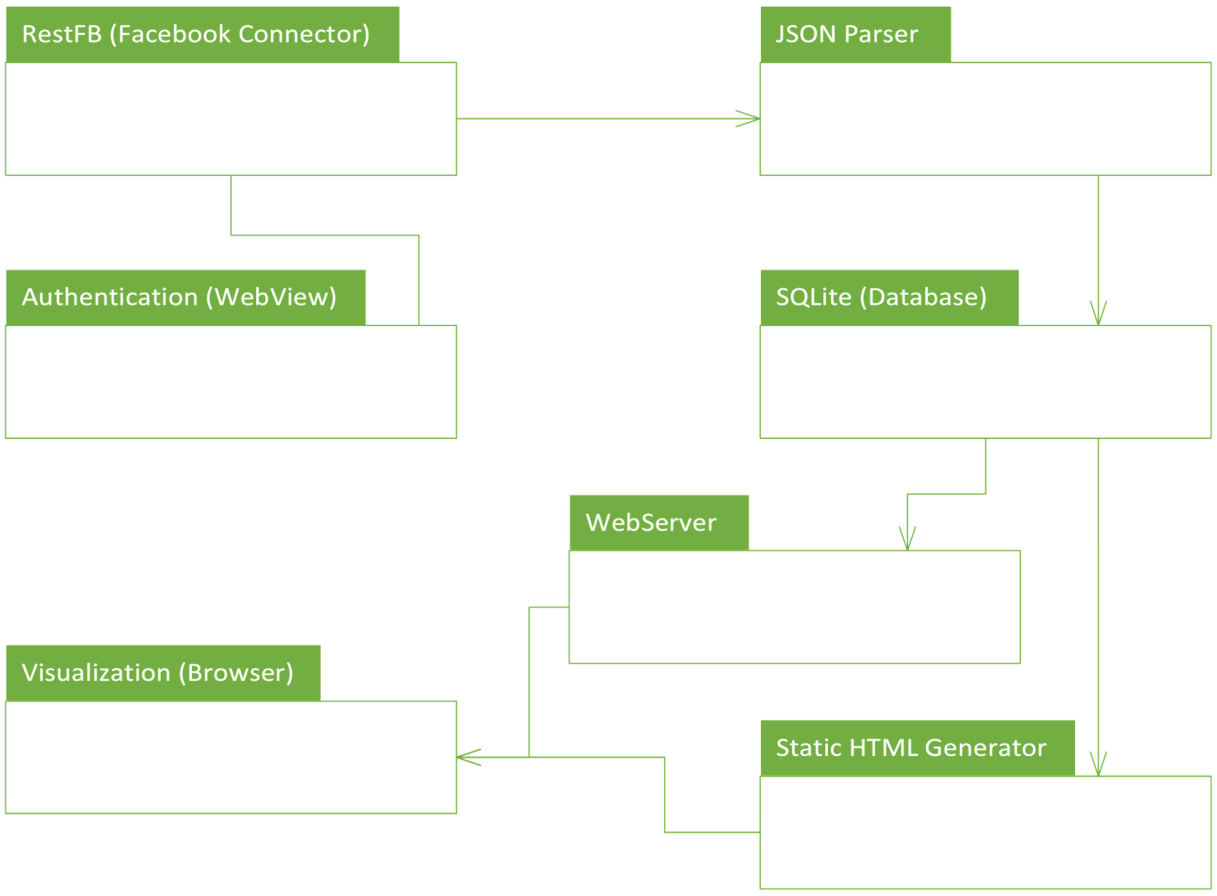

Figure 12 - Preliminary architecture diagram of the Social Aggregator

After the successful authentication, the Facebook Client retrieves the friends list, the wall posts and the interactions on these from FB. The information retrieved is:

- List of friends
- Posts visible on the wall
- Posts visible by friends

For each post, additional information is loaded:

- All comments
- Number of likes

2.4.3.2 Data Visualization

The collected data can be visualized in different ways. Currently implemented is a static graph, visualizing the user's interactions with his friends. The collected data can also be used to generate a FB feed, which is filtered and ordered by different criteria.

The interaction graph is generated by counting the comments or likes visible to the user of the Social Aggregator from each of his friends. The resulting list of friends with the interaction count is then sorted by the number of comments and visualized using D3[7]. An example of the visualization is given in Figure 13. In the resulting graph, ten friends with the highest interaction count are highlighted.

[7] JavaScript based plotting library: http://d3js.org/

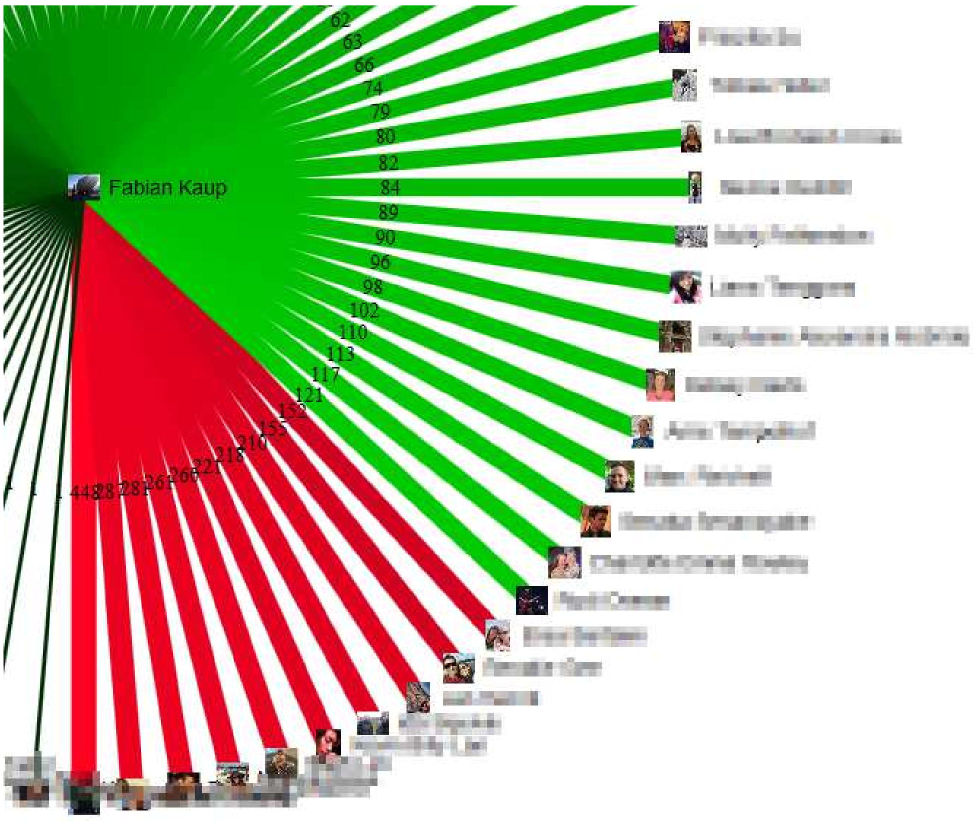

Figure 13 - Visualization of the user's interaction count

This information can then be used to weigh or filter incoming posts before any interaction has occurred. This promises to be useful for the selection of items to prefetch to local devices, in particular video content.

A first implementation of the history-based FB feed filtering is also implemented. Currently it is possible to filter the feed based on the last update time and content type (i.e. all/photo/video). The remaining posts are then sorted by the number of interactions on the post. In the first implementation we count the comments and likes on each post, and sort by comment count first, then by like count. An example of the FB feed filtering and sorting is given in Figure 14. The left is a screenshot of the actual FB stream, while the screenshot on the right is the filtered and sorted feed. On the left, the user must scroll down a bit until the first video is displayed. On the right, the video as defined by the filtering and sorting is displayed at the first position.

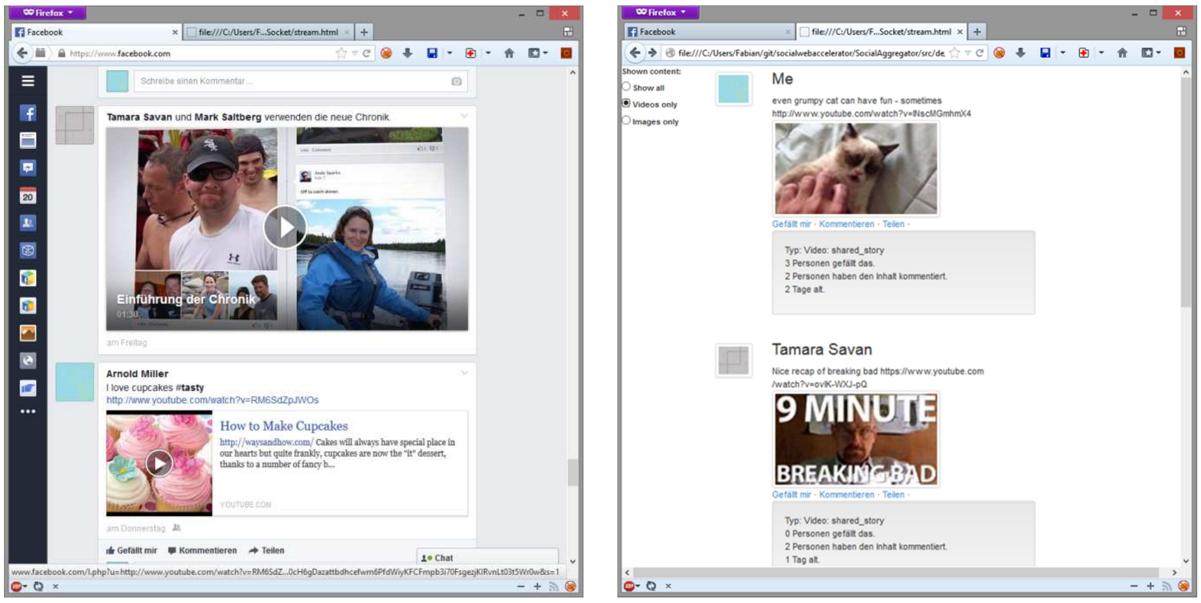
Facebook
file:///C:/Users/F...Socket/stream.html
Shown content:
Show all
Videos only
Images only
Me
Tamara Savan
Gefällt mir · Kommentieren · Teilen

Figure 14 - Comparison of the original and filtered FB feed

The static visualization of the interaction count is statically generated by the Java code. The feed filtering is implemented using a simple Java based web server providing a web socket to allow the page to refresh itself.

2.4.3.3 Deployment

The Social Aggregator is written in Java to allow deployment on a variety of different systems. As an analyser, the software can be run on different desktop systems. After the user has authenticated, the data is fetched from the FB server and the visualizations are prepared. Then the user can connect to the provided web server or open the generated HTML file to visualize his/her social network and his/her interactions.

Another deployment option is to run the Social Aggregator on a centralized machine for a number of users. This allows retrieving the information over a longer time interval without affecting the user. The Social Aggregator can then be used to give recommendations for items to prefetch, based on the collected history. To allow this, an API must be implemented to allow configuration of the Social Aggregator and retrieval of the suggestion. The only information needed for operation is the `userId` and an access token of the user requesting the service.

2.4.4 Module C04: Social Tracer

The Social Tracer module has two goals: first, it is used to collect usage traces for real mobile OSN users and, second, it is used to test and further develop the functionality of the Social Data Collector and the Data Aggregator module that were presented above, as well as the Video Player module. The collected usage traces are essential for the analysis of the content access behaviour of real users to design and implement advanced prediction mechanisms for the Social Predictor module.

To collect the user traces, the Social Tracer uses the output of the Data Aggregator, presents the news feed posts to the user and logs the user's interaction with the items. In particular, the access to videos that are shared using the OSN are relevant for the study on prefetching and the design of prediction mechanisms that are going to build the core of the Social Predictor module. The user accesses its Facebook wall like common over the native facebook app or the browser on his smartphone.

The collected traces can be uploaded to a trace server that is run by us. The upload is triggered periodically by the app.

Figure 15 shows the Social Tracer module, its interactions and related modules.

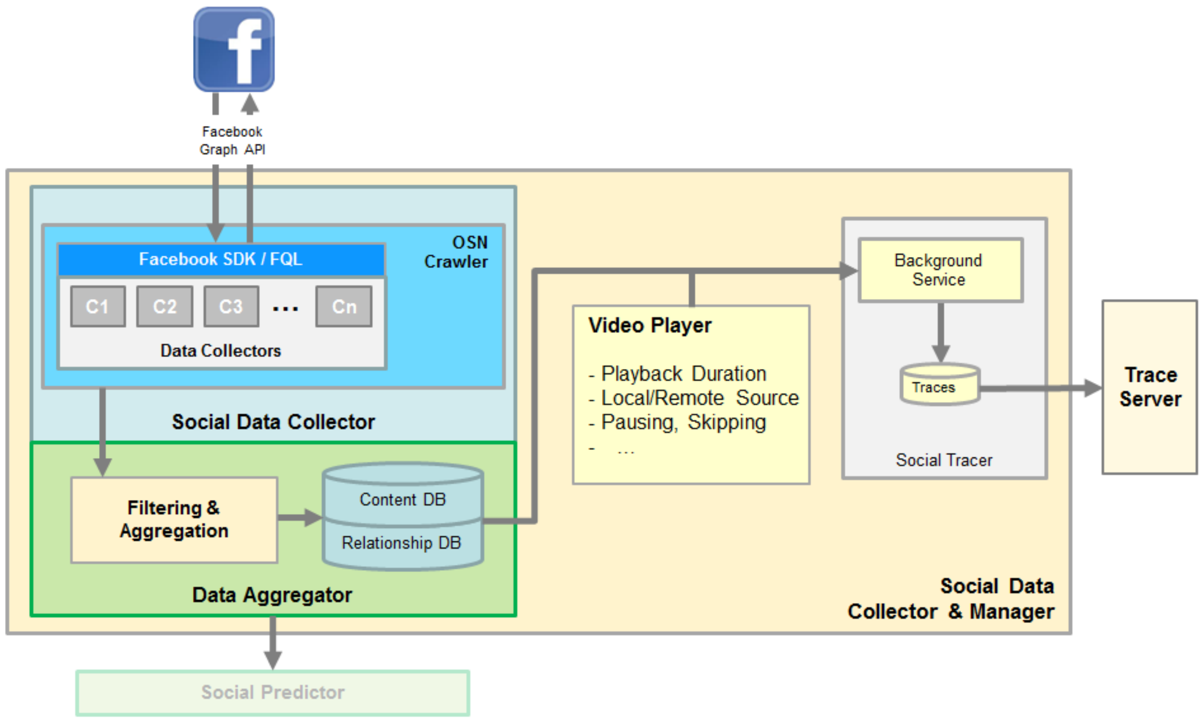

Figure 15 - The Social Tracer module and its dependencies.

For reasons of privacy and to be compliant with the German law, collected trace data is anonymized directly in the Social Data Collector module, when run in the tracing mode. Therefore, all personal information of a user is hashed, using the cryptographic hash function MD5. This way, we are still able to correlate content items as the hash function will always map a certain string to the same hash but we do not process any personal information of a user. Once a user study is finished and all users uploaded their traces to the trace server, the data sets are further anonymized by replacing hashed identifiers with enumerations (an integer value for each hash); where the same hash values can be assigned to the same items.

2.4.5 Module C05 Video Player

The Video Player module is going to be responsible for the mobile video playback. This module is only called when the user opens a video in his App. Therefore the user can use the native Facebook client without limitations but leverage on our prefetching app. If the user clicks on a YouTube video in his stream an event is raised from the Android system. The Video Player module has been registered on the Android system to handle such events. Therefore the user is allowed to choose between all video players he has installed. He can choose to always use our video player for video playback and won't be asked every time which player he wants to use.

Besides the playback, the Video Player module is going to record different events related but not limited to the user behaviour. These event logs are then passed to the Background Service, which stores the event logs in a local MySQL database. The following events are planned to be recorded:

- The system time, when the Video Player module has been started;
- The system time, when the playback starts;
- The system time and the playback time, when the video is paused by the user;

- The system time and the playback time, when the video is resumed by the user;
- The system time and playback time, when the video player is closed;
- The system time and playback time before and after the user skipped in the video.

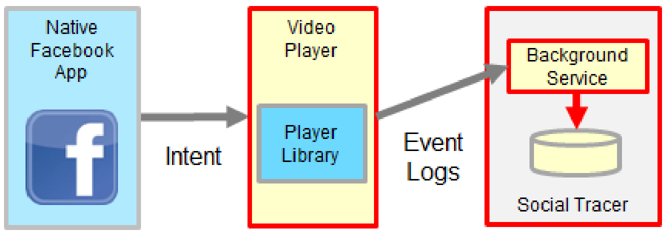

Figure 16 - The Data flow of the Video Player module

2.4.6 Module C06: Social Predictor

The Social Predictor module is going to be responsible to select/identify content items (e.g. videos) from the personal news feed that the user with high probability will access in near future. The module bases its prediction on the collected and aggregated structural and history information as delivered by the Data Aggregator module.

Besides identification, the goal is also to assign a relative importance value to the individual items. This information is then going to be used by the Content Prefetcher module to plan and execute the download of content items.

2.4.7 Module C07: Content Prefetcher

The Content Prefetcher module is going to be responsible for the planning and execution of the prefetching/download of content items that were identified by the Social Predictor module as being relevant for the user with a high probability in the near future. The module uses input on the network status to trigger the download process or learn when to schedule the next downloads. To do so, the module might use simple rule-based or learning models to identify the best time to download relevant items.

Downloaded items should then be opened and presented to the user once he accesses the item on his smartphone. It is yet to be investigated how this presentation to the user could work. While having a proprietary Facebook-clone application that is used by the user to access the OSN would be the easiest solution to control and trigger the access to the content item, it is rather hard to maintain such an application with the rich set of features that state-of-the-art OSN applications, such as the official Facebook App, provide. Therefore, a hybrid approach would be desirable, where the prefetching runs in the background as a service and intercepts the access of videos from the official Facebook application. We are currently investigating whether this is possible. All core modules of the mobile prefetching application that were described so far should be compatible with such an approach.

2.4.8 Module C08: Download Scheduler

The Download Scheduler module is going to determine, when videos are prefetched. A simple approach is to download only if a Wi-Fi connection is established to benefit from the energy-saving properties of Wi-Fi compared to cellular networks. Sometimes this might be insufficient since the user is not connected to a Wi-Fi or if the video is predicted by the Social Predictor module to be consumed in the near future. In this case a download has to be scheduled while the device has only connectivity to a cellular network. To define this decision metric is ongoing work.

2.4.9 Module C09: Download Client

The Download Client module is going to perform the actual download. Therefore it stores the prefetched files in a dedicated folder and marks the video as downloaded in the Prefetching Candidates database afterwards. The module is also responsible for handling failures like aborted downloads, e.g., when a TCP session is aborted due to a change or loss of network connectivity.

To download the videos we need a service which translates the URL of a YouTube video to the actual URL of the mp4 file which we want to download. Therefore we use a small Decoder Service running on a server. This server uses Youtube-dl[8] to retrieve the URL.

2.4.10 Module C10: NS-3 Simulator Extensions and Test Scenarios

We plan to test our use case described in deliverable D2.1 and realized by the modules described in the sections above by means of simulations. To this end, we intend to modify the Network Simulator 3 (NS-3), in order to both accept input from these modules and to be able to simulate properly the related use case scenarios.

Initially, we aim for modifying the existing LTE scheduler implementations of NS-3, which currently focus on a more conservative approach to the allocation of network resources. The modified versions of NS-3 LTE schedulers will allow for a more dynamic resource allocation, modelling more accurately the current commercial installations of LTE. This will be possible by taking into account when a UE does not have any more data at the related eNodeB queues and then excluding it, by subsequent scheduling decisions.

Also, we will create new modules that represent both the transmitting and receiving applications of the server and the client respectively. Finally, we will modify a big number of already existing files in order to generate custom traces, test scenarios that require core parts of the LTE network to behave differently and allow proper integration of the new modules.

2.4.11 Module C11: Bandwidth optimization.

This module is intended to dynamically vary the bitrate of a multimedia server in order to take advantage of the variations in capacity that a UE experiences during the consumption of multimedia content. It will take as an input:

- The predicted future capacity of a UE;
- QoE constraints specific to the content being consumed, i.e., in the case of a video, the user must not suffer pauses caused by rebuffering, despite the possible impact on the network;
- Further constraints, e.g., the module should not transmit more data than that can be stored in the application buffer at any given time instance;
- Feedback from the UE client application.

This module will be implemented on the server side, with only minor extensions to the client side, which will be limited to feedback generation.

The core concept of our approach is the following:

- Transmit at the maximum possible rate, allowed by the actual capacity and the constraints, when the UE is experiencing good capacity compared to the expected capacity of the near future;

[8] http://youtube-dl.org

- Avoid transmission when the UE is experiencing bad capacity compared to the expected capacity of the near future and instead rely on the data that has been transmitted at previous time instances;
- If the buffered data are not sufficient, transmit regardless of the possible impact on the network.

Since the prediction of future capacity and the actual capacity are different, it is expected that the actual rate of data being received will deviate from the predicted one as it has been modelled in [BUIE14]. Thus, the server module will request feedback from the client module in order to assess the magnitude of the deviation between the expected state of the UE and its actual state. If the deviation is bigger than a threshold, the server is expected to recalculate the future allocated transmission rates, in order to make the UE reach again an expected state.

2.4.12 Module C12: Bandwidth prediction

In order to be able to feed the Bandwidth Optimization module with predictions and their confidence, a bandwidth prediction module is in order. For instance, this module will take into account statistical information about user mobility and the bandwidth availability in a given cell.

In order to combine all the statistical information together, different predictors will be jointly used. In particular, we are planning to have at least three different stages of prediction:

- the short term predictor [operating in the order of seconds or tens of seconds]
- the medium term predictor [operating in the order of minutes]
- the long term predictor [operating in the order of tens of minutes]

The first time will be based on models such as ARIMA filters, tuned according to the actual information about the user position and the cell congestion information. This predictor is supposed to offer the most accurate prediction of the available bandwidth.

2.4.13 Module C13: Network visualization

This module is intended to be an overall network performance visualization tool. It shall accept both traces collected from simulation and real time output of network monitoring devices and web applications. The idea is to be able to show the evolution of the network behaviour while the eCOUSIN solutions are applied.

In particular, the visualization tool will build a dynamic network graph based on performance metrics: vertex of the graph will represent nodes in the network, such as users, servers, etc., while edges will represent relationships between any couple of nodes. With the term relationship, we intent to capture any possible information related to two nodes at once: for instance, edges can be used to represent the channel condition between users and base station or the social correlation among users. In the former case, the graph will represent the wireless connectivity map, while in the latter the graph will become the social connectivity map.

In addition, the network visualization tool will draw nodes and edges using different sizes and colours in order to be able to easily represent node and link metrics at glance: for instance, it will be possible to show the variation of bandwidth usage on a given link by varying the thickness of the edge connecting the two devices, as well as representing the QoE perceived by a given user by varying the colour of the vertex.

Finally, as a mean to evaluate the benefits of the eCOUSIN solutions, a split screen mode will be considered in order to show the system working with and without them at the same time.

2.4.14 Module C14: Passive measurement

This module aims at providing lightweight and frequent measurement in order to enable the prediction to work on a time scale small enough for the bandwidth optimization module to operate properly. Note that to efficiently run, the optimization module needs to be able to receive bandwidth estimation samples at a frequency of about one per second. When mobiles are used intensely (video playing, gaming, etc.), this is not a problem, since a passive monitoring application can measure the throughput achieved by the device without performing any advanced operations.

Conversely, when the device is not so intensely used (browsing, email checking, etc.), it is important to be able to estimate the bandwidth availability from very few communication samples. This module achieves that by inferring the steady state throughput of a TCP connection by computing the median bandwidth achievable by considering group of packets at one, thus providing the system with a frequent estimation of the bandwidth availability.

3. ASSESSMENT

This section will describe the assessment of the different demonstrators.

3.1 Content Placement

3.1.1 Test CP1 (Data Collection)

In this test we will address the social and content information retrieval functionality. In particular we will evaluate the Social Data Collector and the Content Information Collector modules. We plan to test this functionality first on real OSNs (Twitter and Google+) and afterwards in our emulation environment as part of the demonstrator. We will evaluate the capacity of these modules to capture the information regarding the social graph associated with a user as well as the information about a specific uploaded content, and properly store the derived information in a database for further use.

3.1.2 Test CP2 (Data Analysis and Social Prediction)

In this test we will address the data analysis and social prediction functionalities. In particular, we will evaluate the Analysis, Mining and Aggregation and the Social Prediction modules. Again we will evaluate these modules using both real data collected from Twitter and Google+ with the tools implemented in the WP3 and controlled experiments in our emulation platform. In particular, we will check the correct functionality and the coordination of these modules. They are expected to process the social and content information and to produce a social-based prediction regarding the most likely locations where a specific content is expected to be consumed.

3.1.3 Test CP3 (Emulation Environment)

In this test we will check whether the emulation environment allows us to reproduce the interaction among users in a social network system. To this end, we will check the functionality of our mock-up social network using the Analysis, Mining and Aggregation and the Social Prediction modules as a social network and content provider services and we will simulate the existence of users in the system publishing and consuming content.

3.1.4 Test CP4 (Content Placement Strategies)

In this test we will evaluate the correct functionality of the different content placement strategies implemented within the Content Placement Strategies module. In particular we will develop different scenarios for each of the implemented strategies in order to confirm that the placement is done correctly.

3.1.5 Test CP5 (Integrated Demonstrator Test)

In this test we will evaluate the correct integration between the different modules of this demonstrator and thus it involves all modules. We study the performance of the social-enhanced content placement algorithms designed in WP4 in our emulation environment. Furthermore we will compare the performance of these algorithms with the ones offered by traditional content placement techniques.

In a first phase we will consider simple emulation set ups that allow us to carefully analyze the correct integration of the different modules and the proper communication between them. In this phase we will use the data obtained with the data collector to generate a scenario following the parameters extracted from real traces. This scenario allows us to understand the best solution in different circumstances

In a second phase we will use real data in order to evaluate the performance of social-enhanced content placement solutions and compare them to traditional content placement mechanisms. To this end we will select a subset of the connected users from the data collector and we will reproduce the behaviour of these users in the social network.

This test along with the simulation studies conducted within WP4 will allow deriving conclusions about the real gain that the proposed social-enhanced content placement solution brings compared to traditional mechanisms.

3.2 Personal Sharing Clouds

3.2.1 Test PSC01 (Social Data Collector/Data Analysis, Aggregation and Mining)

This functional test shall test the user linking and friend discovery phase which is strictly necessary to set up the network. The user should access the media centre web interface and link one or more social accounts to the media centre itself. The media centre should automatically discover which social network friends/followers have the same app. Their names and their statuses (online/offline) should appear as well as an access to an interface allowing browsing remote content.

3.2.2 Test PSC02 (Content look up/Content Copy Selection/Content Dissemination)

In this functional test we will test the functionality of browsing remote resources on top of the social-based peer-to-peer network. The application on the media centre should provide an interface to browse local content and remote content the same way. Access to resources should be managed in order to change if the social relationships change.

3.2.3 Test PSC03 (Hardware Performance)

In the assessment process a phase should be included to understand the precise hardware requirements of the media centre hosting the application. Besides the regular playback and browsing operations, local protocol dissemination of remote information tends to broadcast a lot of packets which can someway slow down connections or create problems if the hardware of the media-centre is not powerful enough. The goal of this test is to set precise hardware requirements for different target environments

3.2.4 Test PSC04 (Usability Testing)

A heuristic analysis of the prototype performed by user interaction expert should give precise indication of the operation needed to turn a prototype into a product.

3.3 Information-Centric Networking

3.3.1 Test IN1 (Content-Centric Routing Strategies for Twitter)

In this functional test, the demonstrator is in a first stage of development and will only provide an integration of a Twitter-like social network over the proposed ICN architecture. The demonstrator will be run either on a local network or using virtual machines, both targeting to emulate a realistic networking environment. We will mainly verify that our locality-aware naming scheme is valid and that our content-centric routing approach based on this naming scheme can work for publication and retrieval of tweets as expected and in the same way as the vanilla Twitter application.

3.3.2 Test IN2 (Integration of Video Live Streaming)

In this second test phase, we will integrate the video live streaming capability into our demonstrator. We will first ensure that the demonstrator can work correctly with respect to the use case defined in the deliverable D2.1. In particular, we will verify that our proposed content-centric and social-driven routing approach are correctly running, proving the feasibility to use NDN as a content transport for an OSN such as Twitter.

3.3.3 Test IN3 (Content Information Aggregation)

In this test we will address the correct functionality to collect and aggregate content availability information obtained from the content infrastructure via a daemon monitoring a caching node and sending update information messages to an ICN controller

The daemon on the caching node tracks a local metafile for changes in a cache index. Upon detecting changes in the metafile, the daemon sends an update message carrying information for newly added or removed content.

We will evaluate the capacity of the daemon to send update messages and of the logically centralized ICN controller module to capture the respective cache information updates received from caching nodes.

3.3.4 Test IN4 (Route Calculation)

In this test we will address the calculation of network routes for individual content objects using topological and content availability information. We will evaluate the benefits provided by evaluating quality parameters such as expected latency (i.e., number of hops traversed by the requests).

3.3.5 Test IN5 (Programming of Forwarding Elements)

In this test we will address the correct functionality of the ICN controller to instruct the forwarding elements according to the calculated content routes. We will check for the correct communication between a logically centralized ICN controller and the ICN controller clients deployed on forwarding nodes communicating out-of band with the controller.

3.3.6 Test IN6 (Final Evaluation of the Testbed)

In this last test phase, we will measure the performance of the demonstrator as a whole (i.e. Twitter-like application + video streaming + NDN delivery layer). We will assert the performance and efficiency of routing and caching, showing for example that processing and routing tweets/videos locally in an efficient way can lead to a large reduction of network traffic and processing load on the OSN servers. The expected results should quantify the benefits of using ICN architecture for an OSN such as Twitter.

We will focus on the performance evaluation of the integrated testbed. Accordingly, the evaluation tests will belong to one of three main categories:

- NDN node performance evaluation with respect to specific functionalities developed in the WP5 for the data and control planes (naming, routing, caching, cache control…),
- Functional and performance evaluation of the social interactions of users in the Twitter-like application (sending and receiving tweets, sharing videos in live streaming, quality of video streaming, etc.),
- Validation, performance assessment and demonstration of the integrated prototype, showing the benefits of delivering UGC over the ICN architecture.

3.4 Content Offloading for Mobile Networks

3.4.1 Test C01 (Social Data Collection and Aggregation)

The purpose of C01 is to test the data collection and aggregation implementation functionality.

Target functionality: Accessing, processing, and aggregating data from Facebook using the Graph API is a rather complex process due to the large number of different content types. Especially nested items that refer to, e.g. items that were reshared and not directly posted to the feed, require a recursive processing. Test C01, therefore, aims at testing the correctness of this process.

Involved module(s): The involved modules are the Social Data Collector and the Data Aggregator as well as their components.

Methodology: In the course of this test, a representative set of content items are to be identified and generated using a real Facebook account that can be used to test the data collection and translation to Java objects (Step 1). Besides, it is to be tested if the objects are correctly mapped and inserted in the local SQLite database (Step 2). For both steps, it would be desirable to use an automated test framework, such as JUnit[9] and define a set of automated tests.

Expected results: The results of Test C01 are to show that the Social Data Collector and the Data Aggregation functionality are working. Thus, the expected result is a passing of all defined tests. Over time, tests might not pass anymore due to changes to the Facebook Graph API and data structures. The tests should help in such a case to identify and fix the encountered problems.

Impact on eCOUSIN: working Social Data Collection and Data Aggregation modules are essential for all further steps of the mobile prefetching demonstrator. The test, therefore, is also important for the eCOUSIN demonstrator.

3.4.2 Test C02 (Social Aggregator)

The correctness and completeness of the displayed information can be tested by manually comparing the Social Aggregator visualization with the data available on the Facebook website or the output of the retrieved data available in Test C01. Furthermore, aggregated statistics can be printed and compared to the data of the Facebook website.

Target functionality: The quality of the feed generator can only be assessed by conducting user studies. For this, the generated feed, or in the case of pre-fetching suggestions the generated list of items, can be compared with traces generated by the Social Tracer (C03).

[9] http://junit.org/

Involved module(s): The only involved module required assessing the completeness and correctness of the collected data is the Social Aggregator module as well as the Facebook web page. The quality of the generated feed, furthermore, might be verified using the traces generated by the Social Tracer (C03).

Methodology: The static visualization must be manually compared with the data on Facebook. As the Facebook API is strictly limited in frequency and scope of the possible requests, no 100% guarantee for the completeness of the data can be given. The quality of the generated feed can be assessed by comparing the traces generated by Test C03 with the generated feed. As this approach is only possible on historical data, the usability is limited. The best approach, also assessing the quality of the generated information, is to conduct user studies. For this, the information retrieved from the Social Aggregator is compared with the user interactions on the feed.

Expected results: The outputs of the Social Aggregator module are charts detailing the past user interactions and a feed generated based on the historical data. If the Social Aggregator module is deployed on a centralized instance, a list of candidates as also presented to the Social Predictor module is returned.

Impact on eCOUSIN: The Social Aggregator module provides a way to investigate and study the output of the Data Aggregator and analyse potential future interests in content items based on social information. This information might be used to define appropriate prediction mechanisms and, thus is important for the prefetching process. Besides, the module is essential for a demonstration of sub-steps of the prefetching process.

3.4.3 Test C03 (Social Tracer)

The purpose of C03 is to test the tracing functionality that is to collect real OSN user traces.

Target functionality: The Social Tracer is a module with an Android service, running in the background. Its functionality is to upload user traces to a static server. As introduced before, the purpose of the module is to collect user traces that include structural and behaviour information of an OSN user. Several sub-functionalities and components are to be tested individually, such as the activeness itself, its functionalities, as well as the upload component.

Involved module(s): The directly involved module is only the Social Tracer and its components. Also important are the inputs from the Social Data Collector and Data Aggregator module that the Social Tracer is based on.

Methodology: Sub-functionalities that are required for this module are going to be tested programmatically using a test framework, such as JUnit, similar to the methodology described in C01.

Expected results: the expected result of this test is a basic verification of the trace collection and upload functionality.

Impact on eCOUSIN: the collected user traces are important as basis for the design and implementation of the Social Predictor module and are expected to highly impact the efficiency of the overall mobile prefetching solution and how it matches the behaviour of real users.

3.4.4 Test C04 (Social Predictor)

The purpose of C04 is to test the prediction functionality used to select and rate content items to be prefetched on a mobile device.

Target functionality: The Social Predictor module uses as input the structural information on the one-hop friendship graph of a target user and a list of news feed posts of the user. Based on this data, it identifies items that a user will consume on his mobile device in near future with a high probability. Besides, the module assigns a relative importance score to the individual items.

Involved module(s): The directly involved module is the Social Predictor and its components. It bases its prediction on the output of the Social Data Collector and the Data Aggregator.

Methodology: The predictions of the module are planned to be evaluated using the real user traces as collected by the Social Tracer module. For an observed list of content items of a real user, the predictions of the module are planned to be compared to the traced user behaviour.

Expected results: The results should allow estimating the potential to predict the behaviour of an OSN user for the access of content items as displayed on his wall.

Impact on eCOUSIN: A mechanism that is able to predict the content items that a user is about to access in the near future is essential to build mechanisms to improve the content delivery process, in particular to mobile devices using prefetching techniques.

3.4.5 Test C05 (Video Player)

The purpose of C05 is to test the functionalities of the video player under different circumstances.

Target functionality: the functionalities to be tested are playback of video content and pausing and skipping during the video payback. The tests will also check if the performed actions have been logged appropriately in a local database.

Involved module(s): Video Player, the Social Tracer is responsible to store the event logs of the video player in a local database.

Methodology: The tests are performed for videos which have been prefetched as well as for those which have not. For those videos which have not been prefetched we perform the testing with different network connections, e.g. Wi-Fi and 3G. Also the abortion of the mobile internet connection is part of the test. For a meaningful set of interactions the event logs will be compared and checked for correctness.

Expected results: The result should show that the video player is working under various circumstances in the expected manner for different basic functionalities like playback, pausing and skipping.

Impact on eCOUSIN: The result should demonstrate that the video player is a usable tool to gain meaningful information about the user behaviour.

3.4.6 Test C06 (Content Prefetcher)

The purpose of C06 is to test the Content Prefetcher functionality of the mobile prefetching application.

Target functionality: The functionality to be tested is the actual prefetching of content items. This includes the decision on which of the selected content items to be prefetched are to be downloaded in which order and when.

Involved module(s): Using the output of the Social Predictor, the only involved module is the Content Prefetcher.

Methodology: The module and its sub-components are planned to be tested programmatically, using a test framework like JUnit. The overall integrated functionality is planned to be evaluated by conducting a user study.

Expected results: The results should show that the prefetching functionality is able to download content items before the user is accessing these items on the mobile device.

Impact on eCOUSIN: A working and efficient content prefetching functionality is essential for a final integrated mobile prefetching demonstrator application and, thus, a proof of concept of the whole mobile prefetching concept developed.

3.4.7 Test C07 (Download Scheduler and Download Client)

Target functionality: The functionalities of the Download Scheduler module and the Download Client module are tested.

Involved module(s): Download Scheduler, Download Client

Methodology: A meaningful set of videos is passed to the Download Scheduler. Here we might insert erroneous items, e.g. videoIDs for videos which do not longer exist, duplicate entries, or a malformed URL. The Download Client is tested while the content is validated after the transfer. It will also be checked if the item is marked as downloaded in the database after the download finished.

Expected results: The correct functionality of the Download Scheduler and Download Client should be proven.

Impact on eCOUSIN: The output of the Download Scheduler and Download Client are the actual files which are prefetched. Therefore this is one of the major component for the Mobile Content Offloading scenarios, since it allows further to request the data downloaded by the modules from local storage and not from remote over the mobile network.

3.4.8 Test C08 (Evaluation of Bandwidth Availability Prediction)

Target functionality: The quality of the bandwidth and connectivity prediction can be evaluated by comparing the predicted values to real values in various scenarios. Since it is impossible to achieve perfect prediction, we will examine the magnitude of the deviation between the real and the predicted values and whether this deviation is small enough to enable us to use the output of the predictor to other modules. The evaluation will be materialized by means of simulation using the NS-3 simulator tool.

Involved module(s): This task will require the cooperation of the Network Monitoring module, as described in WP2, with external modules that provide a more detailed overview of the context of the UE. These include a mobility module and a background traffic module that monitors the behaviour of other UEs in the current and neighbouring eNodeBs.

Methodology: We will run a set of simulations representative of both simple toy scenarios and scenarios that represent real world situations (e.g., commuting by metro, car, and on foot). From these scenarios we will collect traces of key performance indicators (KPI). Our KPIs will include the size and time of arrival of transport layer packets to the UE, the size and time of arrival of LTE transport blocks to the UE, the RBGs allocated by the scheduler to the UE over time, signal quality indicators and finally, the position and velocity of the UE.

We will use these traces as input to the predictor and compare the output of the predictor with the output of the simulation.

Expected results: This test should allow us to evaluate the magnitude of the deviation between the expected and the real bandwidth values in different scenarios.

Impact on eCOUSIN: The output of the predictor is one of the main inputs to the bandwidth optimization algorithm. A proper understanding of its potential can allow us to maximize its contribution to an optimized utilization of network resources.

3.4.9 Test C09 (Evaluation of Bandwidth Allocation Optimization)

Target functionality: This test will allow us to evaluate the real life gains in regards to network resources, of using the bandwidth allocation optimization algorithm. We expect to test its behaviour in different scenarios and monitor the consumed network resources and QoE indicators related to the multimedia content being consumed by the users in each scenario. The evaluation will be materialized by means of simulation using the NS-3 simulator tool.

Involved module(s): This test will require the cooperation of the Network Monitoring module, the Network resource allocator module and external modules that keep track of the behaviour of the client and server applications that participate in the content transmission.

Methodology: We will run a set of simulations representative of both simple toy scenarios and scenarios that represent real world situations (e.g., commuting by public transport, car, and as pedestrian). Initially, we will observe the behaviour of a benchmark application that generates constant bit rate traffic and generate traces of network KPIs and QoE indicators specific to the multimedia content. Next, we will run the same scenarios and we will collect the same traces, but instead of CBR we will use a varying bit rate which will be dictated by the bandwidth allocation optimization algorithm.

Finally, we will compare the performance of the benchmark and the optimization algorithm in regard to capacity savings, robustness and overhead.

We intend to repeat the same process for various implementations of the optimization algorithm in order to both examine the possible tradeoffs of each implementation, as well as find the most appropriate values of the various parameters for each scenario.

Expected results: The output of the optimization algorithm is the varying bit rate which the server application should use at different time intervals. We will compare the aggregated performance, derived by the combination of KPIs and QOE, of different implementations of varying bit rate servers both between them and against a CBR server, in different scenarios.

Impact on eCOUSIN: One of the main objectives of eCOUSIN is the reduction of utilization of network resources, while guaranteeing QoE. The optimization algorithm and in turn its output (the varying bit rates) is the last step towards achieving this goal. Thus, its fine-tuning that this test enables, can significantly contribute to this end.

3.4.10 Test C10 (Evaluation of the Network Visualization)

Target functionality: The network visualization functionality will be tested here. However, in order to be sure that the impact of the other functionalities can be properly visualized, this test will be repeated on the output of the other tests.

Involved module(s): The main module under test here will be the network visualization tool, but the actual output of the test will also depend on the positive outcome of the tests of the other modules.

Methodology: First, synthetic inputs will be fed to the visualization in order to verify the intended behaviour under controlled and repeatable conditions. The synthetic inputs will be diverse so as to be able to visualize all the needed performance metrics.

Subsequently, the output of the previous tests will be fed to the demonstrator in order to verify whether it is possible to effectively show the outcome of the tests and to be able to highlight the benefit of the eCOUSIN solutions.

Finally, the visualization tool will be tested against real-time data, in order to verify whether it can be used as real-time network visualization tool.

Expected results: The test is meant to verify the capabilities of the network visualization tool and the tool is expected to be able to visualize networks and their performance in a practical and well-understandable way.

Impact on eCOUSIN: The project will highly benefit from this test as it will be a crash test for actual demonstration and dissemination activities.

3.4.11 Test C11 (Evaluation of the Passive Measurement Module)

Target functionality: The network monitoring functionality will be tested here.

Involved module(s): Passive monitoring module.

Methodology: Passive measurement techniques will be compared to active and lightweight active technique in order to validate the accuracy of passive solutions.

Expected results: This test should be able to prove that accurate passive measurement can be achieved without the need for expensive and time-consuming bandwidth probing.

Impact on eCOUSIN: This test will prove that advanced techniques such as bandwidth prediction and optimization, which require a quite dense bandwidth sampling, are feasible on mobile devices.

3.5 Timelines

The timeline for each demonstrator will be given in terms of functionality and test definitions and test execution.

3.5.1 Content Placement

Functionality name	Module code	Test code	SW 1st rel	SW final rel	Test defined	Test done	Final assessment
Social and Content Information Collection	Social Data Collector	Test CP1	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
		Test CP4	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
	Content Information Collector	Test CP1	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
		Test CP2	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
Data Analysis, Mining and Aggregation	Data Analysis, Mining and Aggregation	Test CP2	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
		Test CP4	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
Social Predictor	Social Predictor	Test CP2	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
		Test CP4	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30

Strategies for Content Placement	Content Placement Strategies	Test CP3	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
		Test CP4	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
Integrated Functionality	All Modules	Test CP4	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30

3.5.2 Personal Sharing Clouds

Functionality name	**Module code**	**Test code**	**SW 1st rel**	**SW final rel**	**Test defined**	**Test done**	**Final assessment**
Social Based Peer Interconnection	Social Data Collector	SC01	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
		SC02	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
	Network monitoring	SC03	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
Content Sharing	Social Data Collector	SC01	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
	Social aware content naming scheme	SC04	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30
Content Availability and Binding	Content Look Up	SC04	Month 24-26	Month 28-29	Month 18-20	Month 27-28	Month 29-30

3.5.3 Information-Centric Networking

Functionality name	**Module code**	**Test code**	**SW 1st rel**	**SW final rel**	**Test defined**	**Test done**	**Final assessment**
Twitter-like Application over the NDN Architecture	1/ Twitter-like OSN 2/ NDN delivery with social-driven content naming	Test IN1	Month 17-18	Month 21	Month 18-19	Month 20-21	Month 21
Integration of Video Streaming in the	media streaming transport protocol over	Test IN2	Month 23-24	Month 24-25	Month 23-24	Month 24-25	Month 25

Demonstrator	NDN						
SDN Controller to Instruct Forwarding Nodes	ICN/SDN Controller	Test IN3	Month 23-24	Month 24-25	Month 23-24	Month 24-25	Month 25
SDN Controller Application to Calculate Routing	SDN/ICN Controller App	Test IN4	Month 23-24	Month 28-29	Month 24-25	Month 28-29	Month 29-30
Client on Forwarding Node Receiving Instructions	SDN/ICN ICN Client	Test IN5	Month 17-18	Month 21	Month 18-19	Month 21	Month 21
Evaluation of the Demonstrator in the Testbed	Final Demonstrator & Evaluation	Test IN6	Month 23-24	Month 28-29	Month 24-25	Month 28-29	Month 29-30

3.5.4 Content Offloading for Mobile Networks

Functionality name	**Module code**	**Test code**	**SW 1st rel**	**SW final rel**	**Test defined**	**Test done**	**Final assessment**
Social Data Collection and Analysis	Social Data Collection and Aggregation	C01	Month 14-16	Month 18-20	Month 16-18	Month 18-20	Month 28-30
	Social Aggregator	C02	Month 14-16	Month 18-20	Month 16-18	Month 18-20	Month 28-30
	Social Tracer	C03	Month 14-16	Month 18-20	Month 16-18	Month 18-20	Month 28-30
	Video Player	C05	Month 22-24	Month 26-29	Month 24-26	Month 28-29	Month 28-30
Content Access Prediction	Social Predictor	C04	Month 18-20	Month 26-29	Month 24-26	Month 28-29	Month 28-30
Prefetching of Content Items	Content Prefetcher	C06	Month 20-22	Month 26-29	Month 24-26	Month 28-29	Month 28-30
	Download Scheduler	C07	Month 20-22	Month 26-29	Month 24-26	Month 28-29	Month 28-30
	Download Client	C07	Month 20-22	Month 26-29	Month 24-26	Month 28-29	Month 28-30
Network	Bandwidth	C08	M18	M20-	M19	M21-	M28-30

Monitoring	prediction			24		26	
	Passive measurement	C11	M18-20	M20-24	M19	M21-26	M28-30
Network Configurator	Bandwidth optimization	C09	M15-16	M20-24	M16-18	M21-26	M28-30
Network Visualization	All	C10	M20-22	M26-29	M22-23	M23-25	M28-30

4. CONCLUSIONS

This deliverable completed the preliminary plan for the realization of the eCOUSIN demonstrator and assessment campaign. In particular, four demonstrators have been foreseen in order to speed up the realization process while the technical work packages are still working on the technical solutions. This will allow a cross-checking mechanism between the assessment part of WP6 and the technical solutions.

WP6 next activities focus on the release of the software components that are forming the demonstrators and that will be used to validate the project (D6.3 and D6.4). Finally, D6.5 will report the outcome of the final assessment campaign.

REFERENCES

[BUIE14] Nicola Bui, Foivos Michelinakis, Joerg Widmer. 2014. A Model for Throughput Prediction for Mobile Users. In Proceedings of European Wireless (EW'14). Barcelona, Spain, 14, May 2014, pages.1-6.

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

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

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

[D4.2] EU eCOUSIN: Public Deliverable D4.2 - Final Report and Initial Software Release of the Design Extensions and Preliminary Implementation, April 2014.

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

[D6.1] EU eCOUSIN: Public Deliverable D6.1 - Preliminary Plan for System Integration and Assessment, January 2014.