Quality of Experience-Enabled Social Networks

By

Ahmed Abouzeid

A Thesis

In

The Department of Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec Canada

May 2014

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ahmed Abouzeid

Entitled: "Quality of Experience-Enabled Social Networks"

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

Complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. K. Khorasani

_____ Examiner, External

Dr. L. Kosseim (CSE) To the Program

_____ Examiner

Dr. D. Qiu

_____ Supervisor

Dr. F. Khendek

_____ Supervisor

Dr. R. Glitho

Approved by: _____

Dr. W. E. Lynch, Chair

Department of Electrical and Computer Engineering

_____20_____ _____

Dr. C. W. Trueman

Interim Dean, Faculty of Engineering and Computer Science

ABSTRACT

Quality of Experience-Enabled Social Networks

Ahmed Abouzeid

Social Networks (SNs), such as Facebook, Twitter and LinkedIn, have become ubiquitous in our daily life. However, as the number of SN users grows, the SN usage grows and there is higher demand for users' Quality of Experience (QoE). For instance, some users would prefer to filter some posts, e.g. unwanted friendship requests and certain categories of posts, i.e. sports related posts. Users may also prefer to subscribe to a higher Quality of Service (QoS) level with their SN provider to have, for instance, higher priority on posting/retrieving.

3GPP 4G Evolved Packet Core (EPC)-Based systems are all IP network architectures that enable users to connect to mobile networks through their mobile devices and seamlessly change from one access technology to another. EPC systems enable service provisioning with guaranteed and differentiated end-to-end QoS.

This thesis proposes a novel architecture that enables differentiated QoS and information filtering in SNs to improve the users QoE. The SN is deployed on top of 3GPP 4G EPC-Based systems, and it uses EPC services to enable guaranteed and differentiated QoS. The components of the proposed architecture interact through RESTful web services. This architecture allows users to filter posts using their own criteria and have priority over other users in posting and/or retrieving; thereby, improving users' QoE.

A proof of concept prototype tool has been implemented to illustrate the viability of the proposed architecture and its performance has been partially evaluated.

ACKNOWLEDGEMENTS

I want to express my great recognition and gratitude to the members of the Telecommunications Service Engineering Research Lab, especially Dr. Ferhat Khendek, Dr. Roch Glitho, Fatna Belqasmi, Mohammad Majid Hormati and Ashis Kumar Bhowmik. They were very available and gave me precious advices during my master's thesis.

I acknowledge the financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada, Ericsson Canada and Concordia University.

All my gratitude goes to my family, Adham Abouzeid, Randa Ali, Rana Abouzeid and Omar Abouzeid for their support and encouragement. I also want to thank Cécile Langevin for her encouragements and her daily support.

Finally, thank you to my friends Ahmad Zakaria, Ahmed Shaltout and Khaled Zayed for their continuous encouragements during my master's studies.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SN	Social Network
3GPP	3rd Generation Partnership Project
4G	Fourth Generation
EPC	Evolved Packet Core
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
IP	Internet Protocol
UTRAN	UMTS Terrestrial Radio Access Network
WiMAX	Worldwide Interoperability for Microwave Access
SDP	Service Delivery Platform
REST	Representational State Transfer
ROA	Resource Oriented Architecture
WWW	World Wide Web
HTTP	Hypertext Transfer Protocol
URI	Uniform Resource Identifier
WADL	Web Application Description Language
XML	Extensible Mark-up Language
JSON	JavaScript Object Notation
XHTML	Extensible Hypertext Mark-up Language
HTML	Hypertext Mark-up Language

NGN	Next Generation Networks
ITU	International Telecommunication Union
ETSI	European Telecommunications Standard Institute
ICT	Information and Communication Technology
IP	Internet Protocol
VoIP	Voice over IP
FIFO	First In First Out
RED	Random Early Detection
WFQ	Weighted Fair Queue
IntServ	Integrated Services
DiffServ	Differentiated Services
RSVP	Resource Reservation Protocol
BA	Behavior Aggregate
PHB	Per Hob Behavior
DS Field	Differentiated Services Field
DSCP	Differentiated Services Code Point
IETF	Internet Engineering Task Force
IF	Information Filtering
IR	Information Retrieval
CF	Collaborative Filtering
GPRS	General Packet Radio Service
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Access Network

WiMAX	Worldwide Interoperability for Microwave Access
S-GW	Serving Gateway
PDN-GW	Packet Data Network Gateway
PCRF	Policy and Charging Rules Function
ANDGw	Generic Access Network Gateway
ePDG	Evolved Packet Data Gateway
MME	Mobility Management Entity
AF	Application Function
HSS	Home Subscriber Server
ANDSF	Access Network Discovery and Selection Function
AAA	Authentication, Authorization and Accounting
BBERF	Bearer Binding and Event Rules Function
PCEF	Policy and Charging Enforcement Function
PCC	Policy and Charging Control
EPS	Evolved Packet System
TCP	Transport Control Protocol
UDP	User Datagram Protocol
SCTP	Stream Control Transport Protocol
RADIUS	Remote Authentication Dial In User Service
IMS	IP Multimedia Subsystem
QCI	QoS Class Identifier
ARP	Allocation and Retention Priority
GBR	Guaranteed Bit Rate

MRP	Maximum Bit Rate
AMRP	Aggregate Maximum Bit Rate
OS	Operating System
API	Application Programming Interface
JS	JavaScript
FBJS	Facebook JavaScript
SQL	Structured Query Language
FQL	Facebook Query Language
FBML	Facebook Mark-up Language
IPX	Internetwork Packet Exchange
MPLS	Multiprotocol Label Switching
BSN	Body Sensor Network
E-Health	Electronic Health
M2M	Machine to Machine
RAM	Random Access Memory
GB	Gigabyte
KEA	Keyphrase Extraction Algorithm
Mbps	Megabit per second

Chapter 1: Introduction

This chapter first presents the research domain. It is followed by the thesis motivations and problem statement. After that, it introduces the thesis contributions. Finally, the last section presents the thesis organization.

1.1 Research Domain

A Social Network (SN) is a web-based service that allows users to create a profile (their personal details, interests, pastimes, etc.) and send, accept or reject friendship request(s) to/from other users. SNs allow users to view a list of their friends' profiles and to interact with them by posting and sharing information [1]. SNs are very important in our daily life and users use it in variety of spheres. According to a poll made in 2012, 58% of the people asked, use SNs. Among them, 56% are Facebook users, 14% are LinkedIn users while 11% are Twitter users [7]. SNs are used for different purposes including Leisure (e.g. Facebook), business-related work (e.g. LinkedIn), video sharing (e.g. YouTube), photo sharing (e.g. Flickr) and News (e.g. Twitter).

Quality of Experience (QoE) is the acceptability of a service by a user according to his/her expectations [2]. Quality of Service (QoS) is defined as a set of requirements to be achieved by a network for a certain flow [3]. The users usually have a service level agreement with the service provider and the network provider will make sure that the QoS agreement takes place within the network [24]. There are different parameters that a QoS should tackle, which are packet loss, latency, jitter, throughput and uptime [25]. Differentiated QoS is achieved by applying different QoS levels [27]. Each QoS level (Gold, Silver or Bronze) defines a set of

requirements to be met for the flows of the users who subscribed to it. Information Filtering allows users to only receive their desired data [4], thereby increasing user's satisfaction.

3GPP 4G Evolved Packet Core (EPC)-Based systems are all IP network architectures that separate the data and control paths. The EPC is the core network running on top of the long-term evolution (LTE) access network, any other 3GPP legacy access networks (e.g. UMTS Terrestrial Radio Access Network UTRAN) or even non-3GPP access networks (e.g. WiMAX and Wi-Fi). The main components of the EPC systems are the Serving Gateway (S-GW), Packet Data Network Gateway (PDN-GW), which are common in the data and control paths, and the Policy and Charging Rule Function (PCRF), which is responsible for the definition of the QoS policies [5]. 3GPP 4G EPC-Based systems support seamless service provisioning with guaranteed and differentiated QoS. Service Delivery Platform (SDP) supports the development and management of QoE-Enabled SN applications.

1.2 Motivations and Problem Statement

SNs are very popular and are widely used. Statistics shows that 58% of individuals already use SNs [7]. According to Quantcast, SNs (YouTube, Facebook and Twitter) are among the top 5 websites when ordered by the traffic of the users on the Internet [54]. Twitter, has 554 million monthly active users, yearly uptime of 90%, and its users send 9,100 tweets every second [8]. Another example of SNs, Facebook, has 1.11 billion active users. Every 20 minutes on Facebook, 1 million links are shared and 2 million friend requests and 3 million messages are sent [9]. The high demand on SNs can lead to outages due to congestion. During the FIFA World Cup 2010 for instance, Twitter has experienced 5 hours and 22 minutes of outage due to the excessive posting by the end-users [57]. During the Oscars

2014, it failed again due to the excessive number of retweets of Ellen DeGeneres's post [58]. Reddit, another SN, experienced a downtime in August 2012 during Barack Obama's Ask Me Anything session [59].

These examples of congestion triggered the issue of QoE in SNs. Certain users (e.g. corporations, stock market fund managers) may be very interested in having priority over other users for posting and/or retrieving information. Other users (e.g. professors) might want to define user criteria (e.g. to filter friendship requests from students) in order to have a better SN experience.

We propose in this thesis a novel architecture for SNs with differentiated QoS from the user's perspective, combined with information filtering at the user-level. We aim to improve users' QoE during their usage of SNs. SNs running on top of the EPC network can enable differentiated QoS. However, this must be done through a Service Delivery Platform (SDP). This thesis proposes an architecture that supports differentiated QoS and information filtering in SNs. This architecture consists of a SN-Server, SDP (QoS Enabler) and a Database and is deployed on top of the EPC network. The SN-Server interacts with the users to post, retrieve and filter information. The SDP's QoS Enabler enables the differentiated QoS of the users within the EPC network layer. Finally, the Database stores the user's information, QoS level subscriptions and filtering criteria.

1.3 Thesis Contributions

The contributions of this thesis are:

- A derived set of requirements for QoE-Enabled SNs deployed on top of the 3GPP 4G EPC-Based Systems.

- Review of the State of the Art relevant to the thesis work and its evaluation with respect to the aforementioned requirements.
- A novel architecture of the QoE-Enabled SNs on top of 3GPP 4G EPC-based systems. This architecture meets all the requirements mentioned earlier. It consists of functional entities (SN-Server components, QoS Enabler and Database), interfaces (RESTful and Diameter interfaces) between the components and procedures that the users can initiate.
- A proof of concept prototype of the proposed architecture and its partial performance evaluation.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2 introduces the background knowledge related to SNs, RESTful Web Services, QoE and EPC systems. The chapter explains the concepts related to this thesis.
- Chapter 3 introduces the motivating scenarios, the requirements and state of the art relevant to the QoE-Enabled SNs on top of 3GPP 4G EPC-Based systems. Furthermore, the chapter presents the evaluation of the state of the art with respect to the derived requirements.
- Chapter 4 presents the proposed architecture for QoE-Enabled SNs. The proposed solution is deployed on top of 3GPP 4G EPC systems. The chapter discusses the functional entities and interfaces in the architecture, procedures that the users can initiate and an illustrative scenario to show how the components of the architecture interact.

- Chapter 5 discusses the prototype through the software architectures of the different components and the operational procedures to show how these components interact. The proof of concept prototype is presented before conducting a partial performance evaluation.
- Chapter 6 concludes the thesis and presents potential future work on QoE-Enabled SNs.

Chapter 2: Background Information on SNs, RESTful Web Services, QoE and EPC

This chapter presents the background information that is relevant to this thesis. The background information covers four topics: Social Networks (SNs), RESTful Web Services, Quality of Experience (QoE) and Evolved Packet Core (EPC).

2.1 Social Networks (SNs)

This subsection first introduces the definition of SNs, then it presents the history of SNs. Furthermore, it illustrates the structure of SNs and defines the concept of a user profile. Finally, it describes the current applications on SNs.

2.1.1 Definition of SNs

Social Networks (SNs) are web-based services that allow users to:

- Construct a public or semi-public profile within the SN.
- Establish and manage a list of friends.
- Share Information with other users (friends) by posting and retrieving [1].

SNs accommodate "networking" within its framework. The term networking describes the phenomenon of initiating relationship with other users who can be strangers to the initiator. Furthermore, SNs allow its users to communicate with other users that they already know. Finally, the backbone of the SNs is allowing users to create profiles that are visible for other users and that lead to their future interactions within the framework of the SN [1].

2.1.2 History of SNs

Figure 2.1 shows the timeline of the launch dates of the most famous SNs from 1997 to 2006 and a brief discussion of their functionalities and aims of establishment is presented as follows [1].

- **SixDegrees.com:** SixDegrees.com is the first recognizable SN. It was established in 1997. SixDegrees.com allowed its users to create profiles, send/receive friendship requests, list their friends' profiles and interact with them through sending/receiving messages. At the time, SixDegrees.com was the first SN to combine the profile establishment, communication between the users and their friends and communication between users and strangers. SixDegrees.com closed in 2000 because the founders believed that it was ahead of time and there was not much to do after accepting friendship requests.
- **AsianAvenue, BlackPlanet & MiGente:** These three websites were established between 1997 and 2001. They allowed the users to create professional dating profiles. Their users were able to identify user friends within the SN without waiting for their approval.
- **Ryze.com, Tribe.net & LinkedIn:** Ryze.com is the first SN to help users to manage their business networks. It was established in 2001. Later in 2003, Tribe.net and LinkedIn took the same path as Ryze.com since the later did not have much of popularity. Tribe.net acquired some popularity while LinkedIn became the most popular business-related SN.
- **Friendster:** Friendster was established back in 2002 as a professional dating SN. The aim of Friendster was that users could make better romantic relationships

with friends of friends rather than strangers. According to that, Friendster denied the users access to strangers' profiles and restricted it to friends of friends. The phenomenon of fake profiles emerged because of this restriction. Alongside the fake profile phenomenon, the servers and databases of Friendster were not well equipped to face the exponential growth of users on the SN. Thereby the current status of Friendster is a Social Gaming site rather than a SN-site.

- **Flickr, Last.FM & YouTube:** These SNs emerged due to the rise of social media and user-generated content phenomenon. Last.FM was established in 2003 and targeted the music listening habit. Flickr was established in 2004 and targeted the photo sharing. Last but not least, YouTube was established in 2005 and targeted the video sharing.
- **MySpace:** MySpace was established in 2003. It attracted Friendster users after the rumors that Friendster will apply a fee-based system for its users. MySpace attracted Indie-Rock bands thus attracting their fans to become MySpace users. MySpace key feature was, at the time, to add features based on the users preferences. After that, a lot of teenagers signed up for MySpace. News Corporation purchased MySpace in 2005 attracting the media attention but the popularity of MySpace decreased dramatically after the site was implicated with a series of interactions between adults and minors.
- **Twitter:** Twitter began in 2006 as a SN and micro-blogging website. Its users express their daily life activities through short messages called tweets. Twitter gained a lot of popularity specially after attracting a lot of celebrities; thus

attracting their fans. Twitter is considered nowadays as one of the most popular SN sites alongside Facebook.

- **Facebook:** Facebook began in 2004 for Harvard-students only. The aim of Facebook was to support distinct college networks only. It gained a lot of popularity because it allows developers to develop applications that the users can use. Also, Facebook users were unable to make their full profiles public to other users and gaining access to corporate networks needs an appropriate '.com' address. After that, Facebook expanded and was opened to the public in 2006. Nowadays, Facebook is considered the most popular and successful SN.

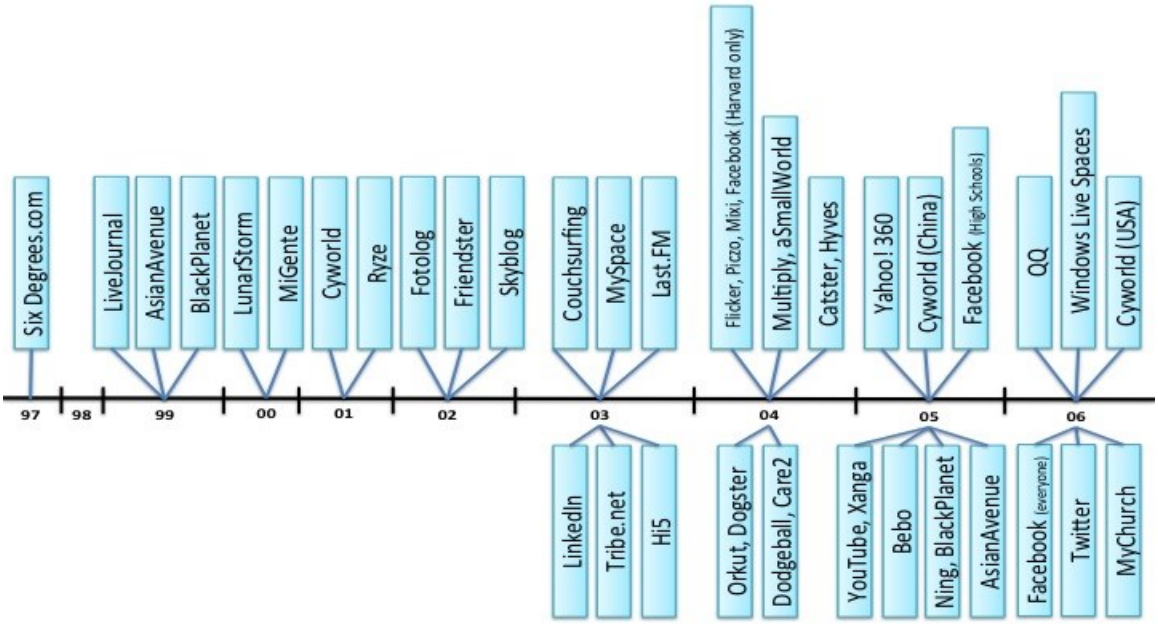

Figure 2.1 - SNs Timeline (1997-2006), taken from [1]

2.1.3 The SN Structure

A Network is a set of relationships consisting of nodes and connections between them. In a SN structure, the nodes are the SN users and the connection represents the friendship [10].

The SN structure can be seen either from the user perspective or from the SN perspective. As per the SN perspective, Figure 2.2 shows the SN of Karate club members. The nodes are users of the SN and the lines are the friendship established between them. On the other hand, as an example of the SN structure as seen from the user perspective, Valdis Krebs is a researcher and consultant in the field of social and organizational network analysis. Figure 2.3 shows the network of his followers on Twitter. Krebs is the central node, the other nodes are his followers and the lines are the connections [10].

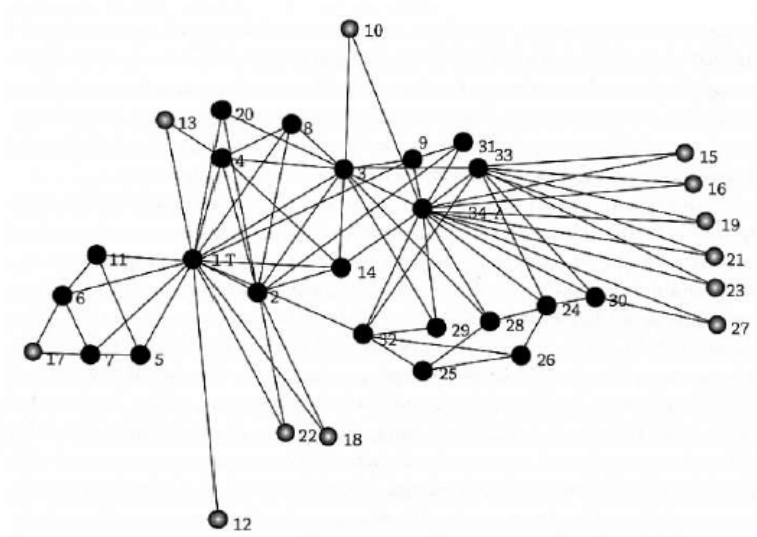

Figure 2.2 - The SN Structure from the network perspective, taken from [10]

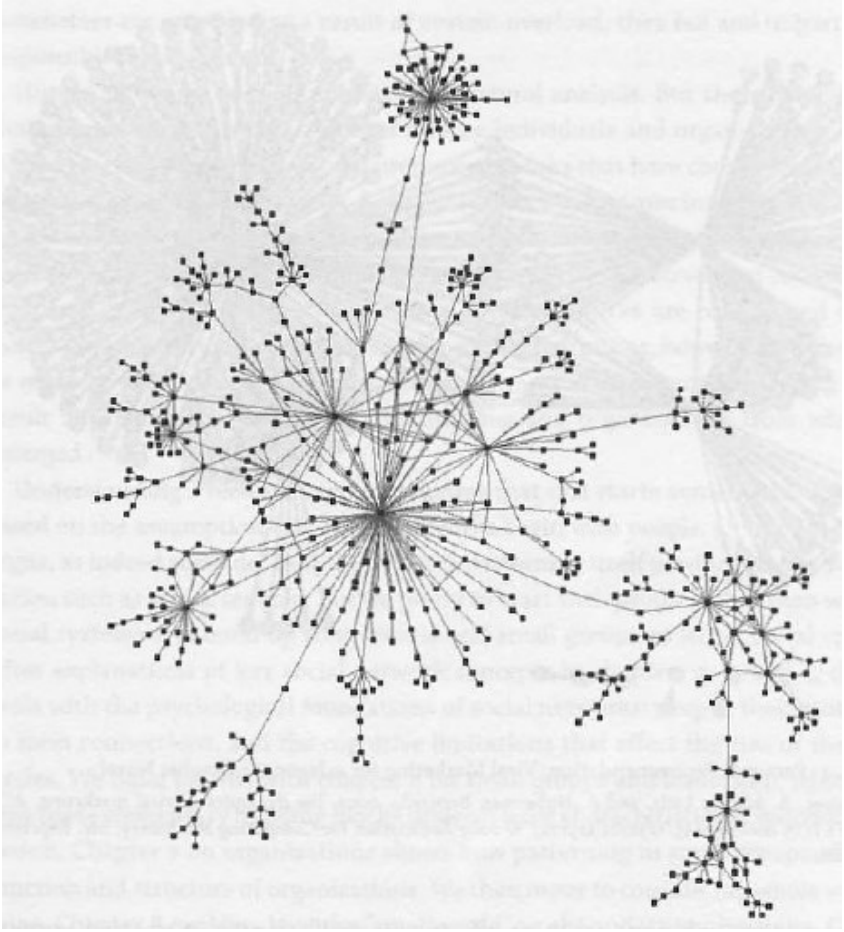

Figure 2.3 - The SN structure from the user perspective, taken from [10]

The SN structure from the user perspective is known as the social graph. The social graph is defined per user. It shows the connections of this specific user to other users of the SN. The social graph summarizes the connections that make up a SN [36].

2.1.3.1 The Concept of User Profile

The user profile is the user's information on the SN. The user starts using a SN by creating a profile. The profile includes the personal information of the user (e.g. name, date of birth, birthplace & current living city), user interests (e.g. favourite music, books & movies) and the user background education [11]. Depending on the type of the SN, the

user profile will fit the needs of the SN to advertise the user in terms of his/her profile. Figure 2.4 shows a sample user profile from Facebook.

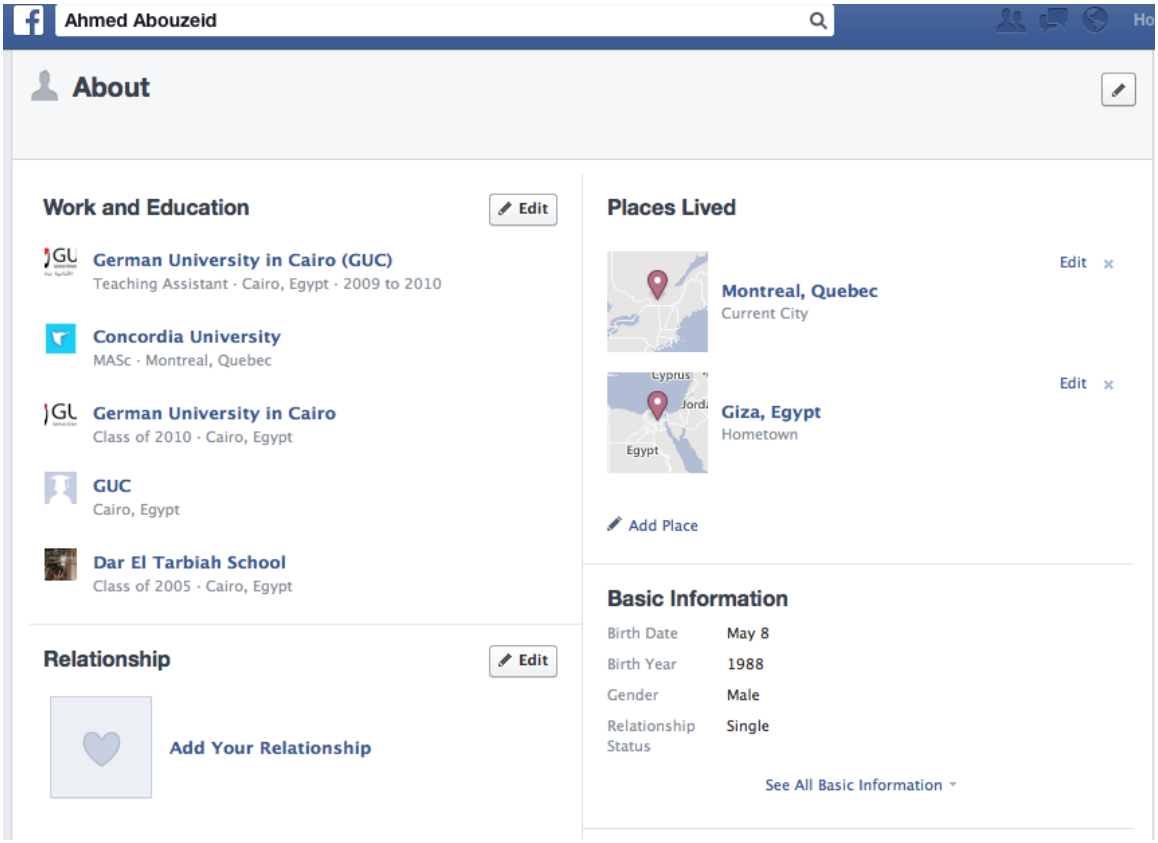

Figure 2.4 - A Sample User Profile on SN (Facebook)

2.1.4 Applications on SNs

SNs are not limited to the text-based communications. Users started sharing photos, videos, conferencing (e.g. Skype using Facebook) and using applications. Applications can be in form of games, quizzes or gift giving. Starting by Facebook, and later the other SNs, the SNs opened their interfaces for third party application servers. This makes the SN applications of third parties available on famous SNs. A distributed approach is usually used in which the SN acts as a proxy between the third party and the users. Figure 2.5 illustrates this approach. The data are stored on the SN database as well as the third

party provider database. The user issues HTTP requests in a designed REST API (Representational State Transfer Application Programming Interface) according to the SN platform used [56]. REST will be discussed in the next section and the most used SN platforms will be illustrated and compared in Chapter 3.

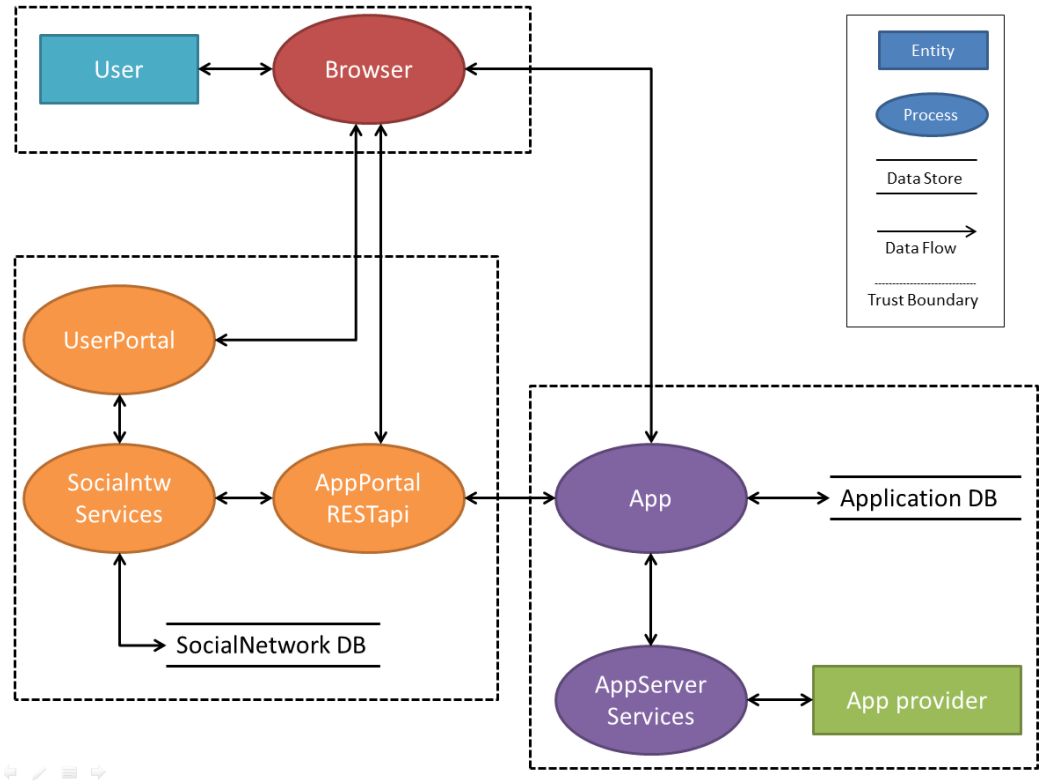

Figure 2.5 - Interaction between users, SNs and Third Party Servers, taken from [56]

2.2 RESTful Web Services

This section introduces the definition of Representational State Transfer (REST) and RESTful Web Services. After that, it discusses the Resource Oriented Architecture (ROA) and illustrates the REST constraints and operations. Finally, the development of RESTful web services is presented.

2.2.1 Definition of REST

R. Fielding first coined Representational State Transfer (REST) in his PhD thesis in 2000 [12]. REST is an architectural style that enables the building of distributed applications using the World Wide Web's (WWW) basic protocols and technology (e.g. Hypertext Transfer Protocol – HTTP [19]) [13].

REST is considered to be a player in Web 2.0. Web 2.0 is described as the WWW sites that use any technology (e.g. Information Sharing Websites as the Social Network sites) beyond the previous static web sites. REST uses the client-server architecture of the Web and uses the basic WWW protocol, which is HTTP, as means of communication [13].

RESTful Web Services were coined as a modification to other web services called 'BIG Web Services'. The RESTful Web Services main features that differentiate them from the others are [20]:

- It is lightweight compared to the BIG web services. It does not require Extensible Mark-up Language (XML [16]) parsing. Wider range of devices is supported by RESTful Web Services because of the lightweight property.
- No toolkit is required to build them, thus it is easy to build.
- RESTful Web Services are stateless, thus scalable.
- RESTful Web Services are human readable through the Hypertext Mark-up Language (HTML).

The Web Application Description Language (WADL [14]) is used to describe the RESTful Web Services. WADL describes the request to the service using the Uniform Resource Identifier (URI) for this service and the body of the request contains the data to

be requested or added. REST models the information as resources and each resource has a URI. The Client uses the RESTful interface to communicate with the server through HTTP messages of GET, POST, PUT and DELETE to get, create, edit and delete a resource, respectively [13]. Resource Oriented Architecture (ROA) is a RESTful architecture that defines the rules for the RESTful Web Services. REST has five main properties, which are: addressability, statelessness, connectedness, uniform interface and cache-ability. ROA and the properties of REST will be discussed later in this section.

2.2.2 Resource Oriented Architecture (ROA)

The ROA relies on five concepts, which are [15]:

- The Resources
- The Resources Names
- The Resources Representations
- The links between the Resources
- The Resources Interfaces

The resource is anything that can be named and which has an important state that the user would be interested in getting, creating or modifying (e.g. an item, a database entry … etc.). Each resource has at least one Uniform Resource Identifier (URI). This URI is used to identify the resource over the Web and to access it. However, each URI must be mapped to only one resource. The resources are available in the server and the client sends a request to get, create, modify or delete the resource [18]. An example of a URI: "http://www.socialnetwork.com/Profile/Alice". This resource contains the profile of user Alice within a SN. Each resource has a representation that shows the current state of the resource. REST allows the resources to have any representation format or media type.

The famous resource representations formats are XML [16], JavaScript Object Notation (JSON [17]), Extensible Hypertext Mark-up Language (XHTML) or plain text. The resources are linked using hyperlinks. The resources can be accessed and manipulated using a uniform interface [18]. The uniform interface relies on the standard HTTP protocol for the communication. The uniform interface will be discussed in the next section.

2.2.3 REST Constraints and Operations

2.2.3.1 REST Constraints

REST has five main constraints, which are [18]:

- **Addressability:** Each resource should be addressable by at least one Uniform Resource Identifier (URI). Any client to get, create, modify or delete the resource uses the URI. One resource can have more than one URI. However, each URI is mapped to only one resource.
- **Connectedness:** RESTful web services representations are hypermedia documents. These documents contain data and links to other resources. The server sends the client information about the states of the resource. Connectedness is the quality of having links. Resources should link to each other in their representations. Moreover, the human web is easy to use because it is well connected while the programmable web is not yet very easy to use.
- **Statelessness:** The statelessness concept implies that when the client sends a request to the server, it should contain all the details of the request and the server should not reply on any previous request. If the client needs to have information

from a previous state, it should send it along with the new request to the server. The possible states of the server are also resources and have their own URIs.

This principle gives the advantages of scalability, reliability and simple implementation to the RESTful Web Services. The application is considered scalable because the server should not store the previous states of the client or store the previous requests of the client. The server just answers the current request by the desired data or required action to be done. The application is considered reliable because, using statelessness, it is easy for the server to recover the partial failures. Finally, the application would be easily implemented because the server does not have to track resource usage across requests.

- **Cache-ability:** This principle implies that the client can cache resources. However, the resources should be defined as cacheable or not in the first place. This decreases the connections between the client and the server thus improving scalability and performance.
- **Uniform Interface:** The client communicates with the server through a uniform interface to manipulate resources. This uniform interface is based on standard HTTP messages. HTTP GET, POST, PUT and DELETE are used to read, create, modify and delete resources respectively. HTTP HEAD and OPTIONS are used to get the meta-data. The operations using these HTTP messages will be discussed in the next subsection.

2.2.3.2 REST Operations

REST uses the standard HTTP messages for communication between the client and the server in order to manipulate the resources [18]. These messages and the corresponding operations using them are discussed below.

- **HTTP GET:** The HTTP GET message is sent by the client to the server in order to read the current state of a specific resource. The GET request must contain the URI of the resource and by that the server can respond to the request. The server responds by a 200 OK message along with the state of the resource in case of a successful request. On the other hand, the server replies by 400 NOT FOUND message in case of a request failure.
- **HTTP POST:** The client, to create a new resource, uses the HTTP POST message. The message should be linked to a parent resource URI. Meanwhile, the server replies by a HTTP message of status code 201, which means that the resource is created. The reply will have the new resource URI in the header message.
- **HTTP PUT:** The client sends a HTTP PUT request to modify an existing resource. The body of the message contains the new representation of the resource in any format depending on the service (e.g. JSON). The server replies by a 200 OK message in case of a successful request and modification.
- **HTTP DELETE:** The client sends a HTTP DELETE message along with the resource URI in order to delete aforementioned resource. The server replies by a 200 OK message in case of success.

The client, to get the header meta-data of the request message, uses HTTP HEAD and the client, to know the operations supported by the resource, uses HTTP OPTIONS. Table 2.1 shows an example from each of the main HTTP messages exchanged between the client and the server.

Table 2.1 - List of Examples on HTTP Methods used by REST

HTTP Method	HTTP Example Message	Message Description
GET	GET http://www.SN.com/Profile/Alice	Reads the representation of Alice's profile in the SN
POST	POST http://www.SN.com/Sessions	Creates a session for a user with the SN (e.g. userId=Alice). The user receives the session ID as a reply.
PUT	PUT http://www.SN.com/Sessions/Session1	Modifies the ongoing session between the user and the server. The body of the message contains the request modification required.
DELETE	DELETE http://www.SN.com/Sessions/Session1	Deletes the session with ID=1 of the user from the SN (e.g. userId=Alice).

2.2.4 RESTful Web Services Development

The procedure to develop RESTful web services starts by identifying the dataset in the system. After that, a classification of the data exchange should be done and followed by splitting the data into resources. The resources reside in the server and the client should

use the HTTP messages to create, modify or remove the resources. The resources should be designed in a hierarchal manner, which means that there would be some child resources that are under their parent resources [18].

The following steps should be performed for each resource. The resource should be given a unique URI. Identification of a list of the HTTP messages used to interact with it should be made. Representations accepted by and served to the client to be designed. The resource should, then, be integrated into the hierarchy of the resources (Who is the parent resource, who is the child resource). Finally, the error conditions should be mentioned [18].

We can illustrate the previous procedure by giving an example on the SN sessions. We have two resources, which are identified by the URIs *Sessions* and *Session-ID*. The *Session-ID* resource is a child resource and its parent resource is *Sessions*. The SN server will keep the resources. The client will use HTTP GET, POST, PUT and DELETE messages to communicate with the server resources. Considering the *Sessions* resource, the client can use HTTP POST to create a new session and HTTP GET to retrieve a list of ongoing sessions with the Server. Considering the *Session-ID* resource, the client can use HTTP GET to retrieve the details of a specific ongoing session, HTTP PUT to modify this session and HTTP DELETE to remove this session from the server. The representation accepted by and served to the client would be, for example, JSON. Finally, the client will receive an error message if he/she uses any un-designed HTTP message mentioned earlier to its corresponding resource (e.g. HTTP DELETE with *Sessions* resource).

2.3 Quality of Experience (QoE)

This section discusses the definition of the term Quality of Experience (QoE) and then it discusses two features that contribute to the QoE in this thesis work, which are the Quality of Service (QoS) and Information Filtering (IF).

2.3.1 Definition of QoE

QoE is a term that emerged with the Next Generation Networks (NGNs) and the rise of service expectancy of the user. There is no general definition for the term QoE. According to the International Telecommunication Union (ITU [60]), QoE is the acceptability of a service by the user depending on his/her prior expectations. The acceptability of the service accounts for an end-to-end service not the intermediate network performance [2].

According to authors in [21], QoE is not the measure of the service delivered by the service provider but the overall experience of the user concerning the outcome of the service. Finally, according to European Telecommunications Standard Institute (ETSI [61]), QoE is the user satisfaction on the objective and subjective psychological measures of the product offered by the service provider [22].

From the definitions of ITU, ETSI and [21], the QoE of the users can be estimated by the acceptance of the service by the users and measured by the performance of the network and the equipment involved in the service delivery to the user. The next two sections will introduce the term QoS and Information Filtering, which are the features used in our thesis to enhance the users' QoE.

2.3.2 Quality of Service (QoS)

2.3.2.1 Definition of QoS

As well as QoE, there is no exact definition for the term QoS. According to ITU, the QoS is the network characteristics and performance used to deliver a telecommunication service that satisfies the user who requested it [23]. According to authors in [24], the QoS is generally stated as measurements that can control the network performance in order to meet the user's requirements. QoS is considered in the technicalities used in order to achieve the user's satisfaction. Figure 2.6 shows the difference between QoE and QoS in a sense that the QoE is always from the user behaviour and the QoS is the technical performance to achieve the required QoE.

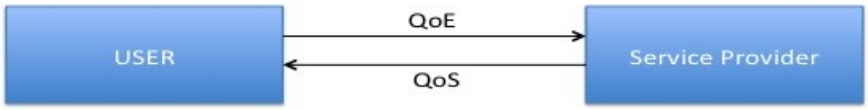

Figure 2.6 - Quality of Experience versus Quality of Service

2.3.2.2 QoS Entities

There are three main QoS Entities that should be available in order to make the QoS requirements and deliver those [24]. Their definitions are listed below and Figure 2.7 shows their interaction briefly.

- **User:** The entity that pays and makes use for/of the service and requests specific requirements for the service.

- **Network Provider:** The entity that provides the network for the telecommunication service. It can be the service provider as well if it offers the service.
- **Service Provider:** The entity that offers telecommunication services to the users.

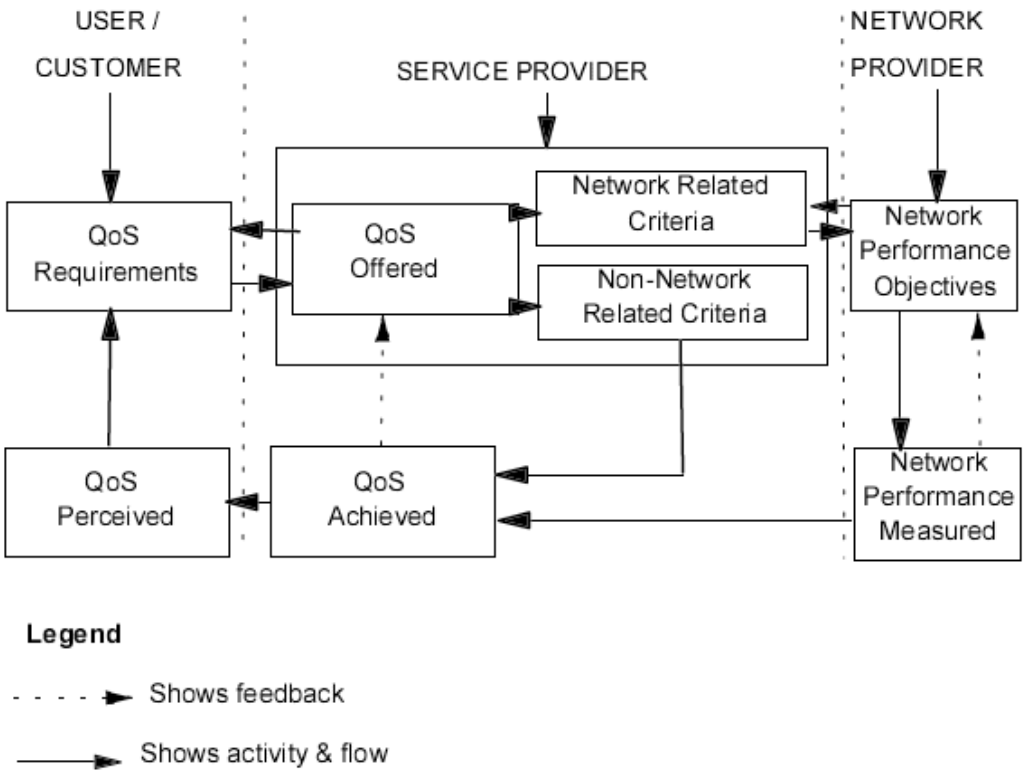

Figure 2.7 - Interactions between the entities, Figure taken from [24]

2.3.2.3 QoS Parameters

The following parameters are the main QoS parameters that affect the users' flow of packets in the network [25]. Applications may regard one of the parameters as more important than others (e.g. streaming applications care about packet loss more than other parameters).

- **Packet Loss:** Some routers would fail in the packet delivery due to the overload of their buffers. This will lead to higher packet drop rate. Some applications can cope with packet loss (e.g. Data Applications). However, other applications are very sensitive to packet loss (e.g. Streaming Applications), in this case the number of packets sent versus the number of packets received affects the QoS of the users.
- **Latency:** Latency is the end-to-end delay of the packets. It might take long time for the packet to reach its destination because of the delays in the buffers of the routers. Latency is very important in some applications like the Voice over IP applications (VoIP).
- **Jitter:** Jitter is the delay variation of the packets. Packets from the same source can reach the destination with different delays and thus will need more processing in the destination side. Jitter is very important in case of the streaming audio or video applications.
- **Throughput:** Throughput is the data transfer rate also regarded as the bit rate.

2.3.2.4 QoS Mechanisms

QoS mechanisms are used in order to insure that the network meets the specific requirements and the network performance is at its best. Below are the main categories of QoS mechanisms to insure that [26].

- **Admission Control:** It is the process of taking the decision whether the data flow will be admitted to the network or not. This depends on the available network resources and the admission policies.

- **Traffic Policing, Shaping or Dropping:** Traffic shaping is used to optimize the network performance. It includes packet marking and classification, enforcing policies and congestion management techniques. On the other hand, traffic policing is the process of marking or dropping packets that exceeds the traffic rate. There are some algorithms for traffic policing like the leaky bucket and the token bucket algorithms [26].
- **Queue Management and Scheduling:** Queue management and queue scheduling are used to improve the normal First In First Out (FIFO) queue. That is because the QoS requires the routers to have special queues for prioritizing packets and to schedule packets while avoiding network congestion. An example of queue management is the random early detection (RED [62]). An example for the queue scheduling is the weighted fair queuing (WFQ [26]).

This is a brief description of the QoS mechanisms, more can be found in [26].

2.3.2.5 QoS Levels

QoS levels are the service levels that the service provider can offer to the user through the network provider. The network provider defines the service levels to control the QoS mechanisms and parameters in the network. As per the Internet Engineering Task Force (IETF [63]), the IP networks can provide one of three QoS levels, which are best-effort service, integrated service and differentiated service [27].

- **Best-Effort Service:** Best effort service is the basic IP network service. It uses the FIFO queue in the queue management and scheduling. Some applications work properly using the best effort service. However, some applications sensitive to

some QoS parameters (e.g. delay and jitter) will be affected by the best effort service in case of network congestion.

- **Integrated Services (IntServ):** IntServ is a QoS related network service. It is a per-flow service that needs resource reservation before the admission of the flow. IntServ supports two types of traffic classes, which are guaranteed services and controlled-load services [26].

  In a controlled-load service, the flow receives a service close enough to the best-effort service in the case of unloaded network and then the service degrades with the increased network load. On the other hand, in the guaranteed service, the flow has strict requirements on the bandwidth, delay and other QoS parameters that the routers should meet before admitting the flow. The IntServ model is based on the Resource Reservation Protocol (RSVP) [26].

  RSVP is a signaling protocol used in reservation of resources within the network to guarantee the service for the flows. It uses two main messages to reserve the resources, which are PATH and RESV messages. Figure 2.8 shows how the resources are reserved using RSVP within an IP network. If any intermediate router fails to reserve the resources, the whole process will be aborted. IntServ is difficult to implement since the entire routers should support classification and scheduling mechanisms. Also, each router should store information about each reservation flow.

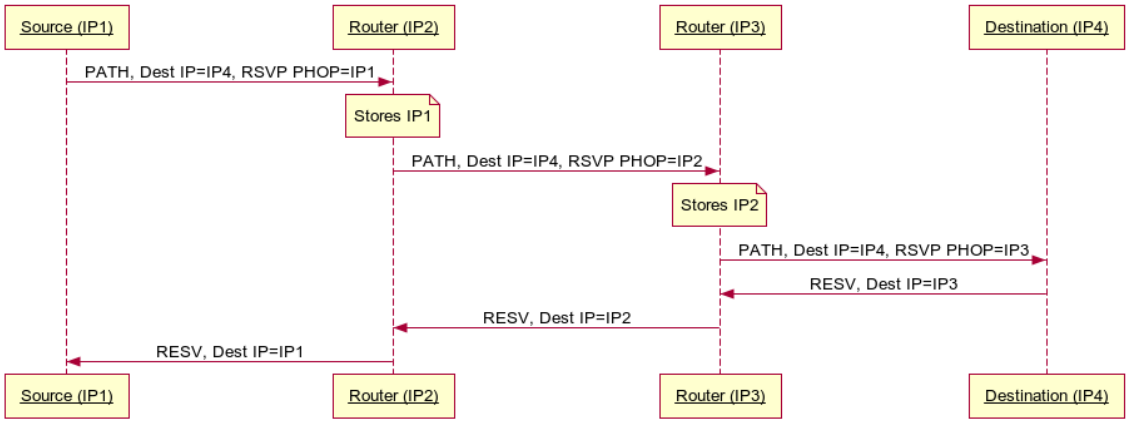

Figure 2.8 - Illustration of the RSVP Signalling

- **Differentiated Services (DiffServ):** DiffServ classifies the network into smaller classes of services compared to the IntServ. DiffServ performs per hob behavior (PHB). DiffServ, unlike IntServ, does not need resource reservation. DiffServ performs the marking; scheduling and policing of packets only in the border routers while the intermediate routers perform the behavior aggregate (BA). In the DiffServ, changing the Differentiated Services Field (DS Field) in the IP packets to specific Differentiated Services Code Point (DSCP) prioritizes the packets. Moreover, DiffServ is considered a more scalable solution when compared to IntServ since the network does not have to reserve resources for the flows. Also, it is easy to implement the QoS provisioning with the DiffServ. Figure 2.9 shows how the DiffServ works within the IP networks.

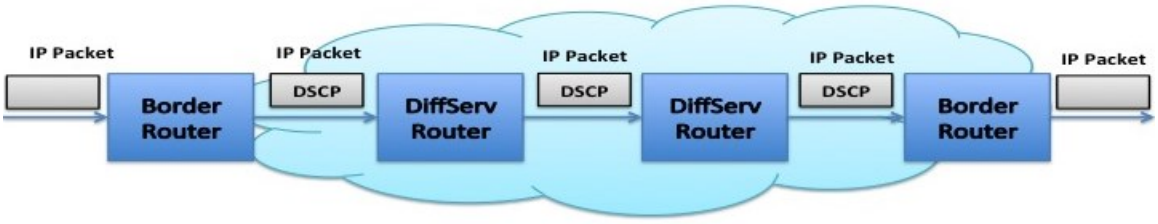

Figure 2.9 - DiffServ Routing

2.3.3 Information Filtering (IF)

This subsection will illustrate the definition and concepts of IF in the light of the Social Networks (SNs).

2.3.3.1 Definition and Concepts of IF

There are various definitions for the term Information Filtering (IF) but they refer to the same concept. According to authors in [4], IF is picking particular objects from incoming posts that would have high probability in satisfying the user. Another definition used by G. Pasi, IF is the process of choosing a subset of information stream and presenting it to the user. A process of identifying the user's needs and interests should be prior to the IF process [28]. IF needs an information filter, which stores users' needs and evaluates if the retrieved information is in the user's interest or not [28].

The IF systems have the following characteristics [29]:

- They are applicable for unstructured data as well as semi-structured data.
- They handle large amount of data.
- They deal with text-based messages.
- They take their decisions of filtering according to users criteria (user's preferences).
- Their objective is to remove irrelevant data from the retrieval stream (irrelevant due to the user's preferences).

There are different types of IF [28] [31], among them are:

- **User-Based Filtering:** User-based filtering is a type of filtering that allows users to hide entire activity from certain users in their SN context.
- **Content-Based Filtering:** Content-based filtering is concerned about text-based objects to be filtered. They use content analysis tools in the filtering engine to reach decisions about filtering.
- **Collaborative Filtering (CF):** Collaborative filtering is based on usage analysis and learning from the user's previous history with the entity (e.g. SN). There are three types of CF, which are: memory-based, model-based and hybrid. More information about CF is found in [31].
- **Hybrid Filtering:** Combines the previous filtering approaches.

2.4 Evolved Packet Core (EPC)

2.4.1 Definition of EPC

EPC-Based systems are all IP network architectures of the 4th generation mobile networks. EPC enables users to connect to mobile networks through their mobile devices and seamlessly change from one access technology to another. These systems separate the data and control paths. The EPC is the core network running on top of the long-term evolution (LTE) access network, any legacy 3GPP access network (e.g. GPRS and UTRAN) or any non-3GPP access network (e.g. Wi-Fi and WiMAX). The main components are the Serving Gateway (S-GW), Packet Data Network Gateway (PDN-GW), which are common in the data and control paths, and the Policy and Charging Rule

Function (PCRF), which is responsible for the definition of the QoS policies [5]. 3GPP 4G EPC-Based systems enable guaranteed and differentiated QoS.

2.4.2 EPC Architecture

Figure 2.10 shows the EPC architecture and its main components and they are discussed in details as follows.

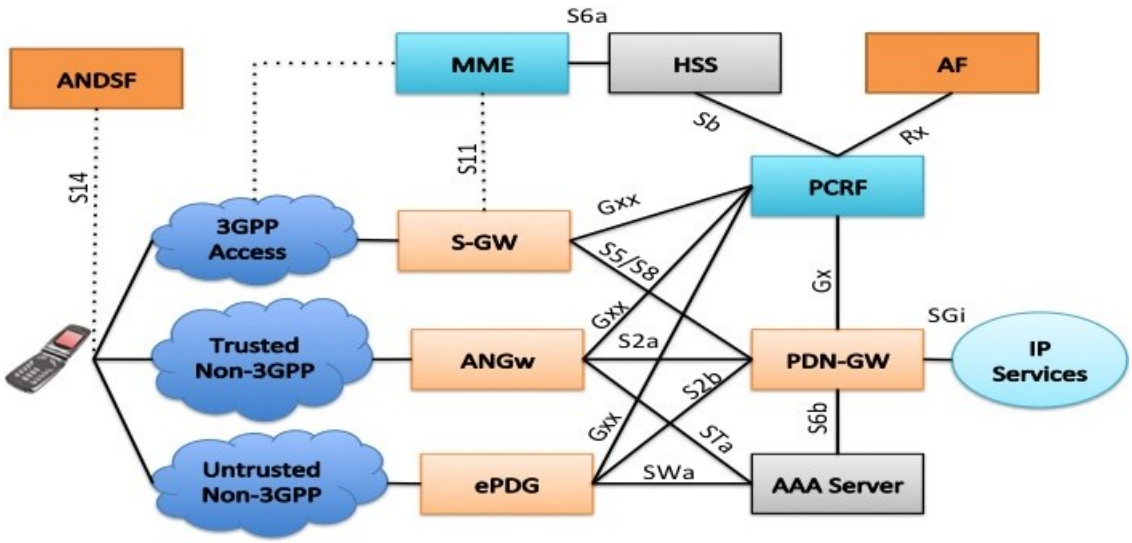

Figure 2.10 - EPC Architecture, taken from [5]

- **Serving Gateway (S-GW), generic Access Network Gateway (ANGw) & evolved Packet Data Gateway (ePDG):** S-GW, ANDGw & ePDG are access gateways responsible for connecting the 3GPP, trusted non-3GPP & untrusted non-3GPP access networks to the IP based core network, respectively. They are considered as the local mobility anchor points. S-GW contains the Bearer Binding and Event Rules Function (BBERF) component, which enforces the QoS rules and policies. Meanwhile, ePDG authenticates and authorizes its users. It also

marks the traffic packets with different DSCP to have different traffic classes within the core network [5].

- **Packet Data Network Gateway (PDN-GW):** PDN-GW is the gateway that connects the users with the external packet data networks. It contains the Policy and Charging Enforcement Function (PCEF) component, which supports packet marking, rate enforcement and packet filtering in order to provide QoS provisioning [5].
- **Mobility Management Entity (MME):** MME is considered one of the important components for 3GPP access networks because it authenticates the user with the Home Subscriber Server (HSS). MME is responsible for the selection of the appropriate S-GW and PDN-GW. Moreover, MME is also responsible for the bearer activation and deactivation.
- **Home Subscriber Server (HSS):** HSS is the database that saves the information of the users. It communicates with MME for authentication purposes. It communicates with the Policy and Charging Control Function (PCRF) component for service provisioning purposes [5].
- **Application Function (AF):** AFs refers to applications outside the domain of the EPC architecture and eventually communicates with the EPC architecture (e.g. IP Multimedia Subsystem (IMS) Architecture or third party service provider) [5].
- **Access Network Discovery and Selection Function (ANDSF):** ANDSF uses the users' current location, coverage maps and subscription information in order to discover access networks for the user. Moreover, ANDSF takes handover decisions for the non-3GPP access networks users [5].

- **Authentication, Authorization & Accounting (AAA) Server:** The AAA server is used for authentication, authorization and accounting of the users connected through non-3GPP access networks. The ePDG and ANGw communicates with it in order to authenticate the users. AAA server stores the user's information and communicates with the HSS for user's subscription profile information updates [5].
- **Policy and Charging Rules Function (PCRF):** PCRF is the main component for providing QoS in the EPC architecture. PCRF is the base of the Policy and Charging Control (PCC) system. It provides the PCC rules and policies to the BBERF component in the S-GW and the PCEF component in the PDN-GW to enforce the QoS policies in the network [5].

2.4.3 Diameter Protocol

The Diameter base protocol is an Authentication, Authorization, and Accounting (AAA [64]) protocol designed for network access applications. Diameter protocol is also used in user mobility in home and foreign networks [32]. Diameter can run on top of Transport Control Protocol (TCP [65]) or Stream Control Transport Protocol (SCTP [66]). It evolved and meant to replace the Remote Authentication Dial In User Service (RADIUS [68]), which runs on top of the User Datagram Protocol (UDP [67]). Diameter overcame RADIUS because it runs on top of reliable transport protocols, has network and transport layer security, supports stateless models and is more easily extendable. The diameter protocol was chosen by 3GPP to be the signalling protocol for the EPC architecture. More information about diameter can be found in [32].

2.4.4 QoS in EPC

3GPP EPC-based Systems use the class-based QoS. This is implemented by the Policy and Charging Control (PPC) system in the PCRF component in the EPC architecture. The bearer concept shapes the QoS levels in EPC. Each bearer is associated with certain QoS parameters that define the QoS level. The bearer concept will be discussed in the next subsection. Generally, the EPC architecture follows the DiffServ model to provide QoS to users.

2.4.4.1 The Bearer Concept

As per ETSI, An Evolved Packet System (EPS) bearer is the sum of characteristics applied to a flow in the network layer. The EPS bearer differentiates between the different treatments of packets within the network. Each bearer has the same forwarding treatment (e.g. scheduling policy, queue management policy, rate shaping policy, RLC configuration, etc.) [33]. EPC bearers are classified as default bearer or dedicated bearers in terms of time establishment. On the other hand, they are classified as Guaranteed Bit Rate (GBR) bearers or Non-Guaranteed Bit Rate (Non-GBR) bearers in terms of bandwidth guarantee [34]. The following QoS parameters are used to identify the QoS level within the EPC bearer [34]:

- **QoS Class Identifier (QCI):** QCI is a number that is used to indicate the transport characteristics of a flow (e.g. priority, packet loss rate) and the nodes that determine the packet forwarding treatment (e.g. queue management) uses the QCI as a QoS level indicator.

- **Allocation and Retention Priority (ARP):** This value determines the priority of the bearer either when established, modified or terminated. It is unlikely that a bearer with high ARP be terminated under low network resources conditions.
- **Maximum Bit Rate (MRP):** MRP indicates the maximum bit rate that the corresponding EPC bearer can use.
- **Guaranteed Bit Rate (GBR):** GBR indicates the guaranteed bit rate for the corresponding EPC bearer
- **Aggregate Maximum Bit Rate (AMBR):** AMBR is used to control the bandwidth usage of users across different bearers. The point behind it is that the user can have different bearers and the EPC may, at a point, need to control the overall bandwidth usage of a specific user across all his/her used bearers.

2.5 Chapter Summary

This chapter presented the background information that is necessary for the rest of the thesis. The next chapter will discuss the motivating scenarios of this thesis, the derived requirements and dig more into the related work and its evaluation.

Chapter 3: QoE-Enabled SNs: Motivating Scenarios, Requirements and State of the Art Evaluation

This chapter presents the scenarios that motivate this thesis followed by the derived requirements for QoE-Enabled SNs. Finally, the last section discusses the related work and we sum up the section with a state of art evaluation with respect to the derived requirements.

3.1 Motivating Scenarios

Two main motivating scenarios for QoE-enabled SNs are to be discussed. The first one presents Alice (professor), Charlie (student) and Bob (professor) as users of a SN. Alice provides a user filtering criteria in order to avoid receiving friendship requests from students and receiving sports-related posts from her SN friends. When Charlie and Bob send her a friendship request, Alice will only receive the friendship request from Bob. This scenario illustrates the use of Information Filtering in SNs. It can be used in terms of user-based filtering or content-based filtering.

In the second scenario the SN allows its users, like Sarah, to subscribe to a certain QoS level. There are three QoS levels, which are: Bronze (low), Silver (medium) and Gold (high), and they are user-based QoS levels. Moreover, the SN allows its users to upgrade or downgrade their QoS level during their ongoing sessions. Sarah has a low QoS level (Bronze) subscription in the SN and wants to share live updates of a conference with her colleagues. Due to the overloading of the network (e.g. a lot of ongoing video

conferences), she will not be able to post her updates, so she decides to upgrade her QoS level to Gold for posting the important updates. Furthermore, the SN will keep providing Sarah with the same QoS level subscription at any given time even if she switches from one radio access layer (e.g. Wi-Fi) to another (e.g. LTE).

3.2 Requirements for QoE-Enabled SNs

The requirements to be met in order to achieve a QoE-Enabled SN are derived from the motivating scenarios presented earlier.

In this thesis, we have derived four requirements that any architecture to enable QoE in SNs should meet. They are all functional requirements. The first requirement is supporting user-based filtering. The second requirement is supporting content-based filtering, which allows users to hide posts related to certain topics. The first and second requirements are derived from the first motivating scenario. The third requirement concerns the support for differentiated QoS. The solution should provide users with different QoS levels (Gold, Silver, and Bronze) in posting and retrieving. The QoS levels are user-based differentiated QoS. Moreover, the proposed architecture should enable the users to upgrade or downgrade their QoS level during an ongoing session without the need to restart the session. This requirement is derived from the second motivating scenario. The fourth and last requirement is about entirely/partially reusing existing SN infrastructure, whenever possible, and not build a new system from scratch. Table 3.1 summarizes these requirements.

Table 3.1 - The requirements for QoE-Enabled SNs

Index	Requirement
1	Support User-based Filtering
2	Support Content-based Filtering
3	Support User-based differentiated QoS
4	Reuse entirely/partially existing SN infrastructure

3.3 State of the Art

This section evaluates the related work with respect to the aforementioned requirements. The first section covers the SN platforms and evaluates which SN platform is the best to be the base of our work. The second and third sections cover the two main features for QoE-Enabled SNs, differentiated QoS and information filtering and the related work of each feature in the SNs domain. To the best of our knowledge, there is no related work that addresses both of these aspects together.

3.3.1 Social Network Platforms

A SN platform is an operating system (OS) for SNs. It provides the technologies that enable the social graph. The social graph is defined per user. It shows the connections of this specific user to other users of the SN. The social graph summarizes the connections that make up a SN [36]. Many SN platforms are available. Two main SN platforms are widely used, which are the Facebook Platform and Google's OpenSocial. They are to be discussed in the next subsections followed by an overall comparison and evaluation.

3.3.1.1 Facebook Platform

The Facebook Platform is a framework that provides set of Application Programming Interfaces (APIs), and tools for third party applications to develop and run their own applications on Facebook. According to [36], an API identifies how the software components should interact. It shows how an operating system, a library or a service should deal with requests initiated from computer programs.

The Facebook Platform consists of the following features [36]:

- **Graph API:** It is Facebook's web-service API. The basis of this API is the REST protocol [18]. Facebook uses REST because it is light-weight protocol, easy to implement and uses standard protocols such as HTTP. The REST resources are summarized to: user, friend, group, group_member, event, event_member, photo, album, and photo_tag.
- **Authentication:** This process follows the Facebook Login process, in which the user provides his/her username and password and the Facebook server validates these data with the corresponding database records [44].
- **Facebook Mark-up Language (FBML):** It is a subset of the Hypertext Mark-up Language (HTML). The third party server should use FBML for its content in order for Facebook server to publish and read them.
- **Facebook Query Language (FQL):** It is similar to the Structured Query Language (SQL). It does not support a join query, it selects from one table at a time and it must be index-able.
- **Facebook JavaScript (FBJS):** It is similar to the normal JavaScript but with some differences in the function and variable names.

- **Social Plugins:** These plugins are add-on features to the basic SN functionalities. They provide more interaction means to the SN. Examples of the social plugins are: the like button, the share button, the follow button and the recommendation feed.

Figure 3.1 shows the Facebook Platform architecture. The client sends a request (e.g. Registration request for a gaming application on Facebook) to the Facebook server-using HTTP, the Facebook server forwards the request to the third party server using also HTTP, and the third party server gets the data using SQL. After that, the third party server sends a query using FQL to the Facebook server and the Facebook server replies by the Facebook formatted content. The third party server forms the response and sends it in terms of FBML to the Facebook server, which forwards the reply to the client in terms of normal HTML [45].

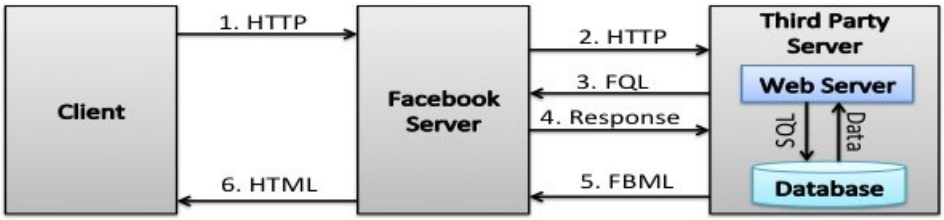

Figure 3.1 - Facebook Platform Architecture, taken from [45]

3.3.1.2 Google's OpenSocial

Google launched a counter part for Facebook platform and called it OpenSocial. OpenSocial is a set of APIs for SNs development. The power of OpenSocial that it is not only exclusive to Google+, third party servers can run associated with different SNs. The supported SNs are Friendster, Hi5, Hyves, imeem, LinkedIn, MySpace, Ning, Oracle, Orkut, Plaxo, saleforce.com, Six Apart, Tianji, Viadeo and Xing [36].

OpenSocial is based on HTML and JavaScript API. The OpenSocial specification uses also RESTful APIs with the following resources: People, Groups, Messages, Application Data, Activities, Media Items and Albums. It is considered simpler using HTML, JavaScript API and SQL as base technologies [35].

Figure 3.2 shows the OpenSocial architecture. The container implements OpenSocial API and Gadget API specifications. Apache Shindig is used as open source software that allows any third party server to serve OpenSocial applications. The existing networking software (third party) communicates with the shindig environment using the OpenSocial Service Provider Interface (SPI). The shindig environment contains a gadget server that stores JavaScript and HTML specifications of social applications in the iframe component and make them available by HTTP methods. More can be found in [35].

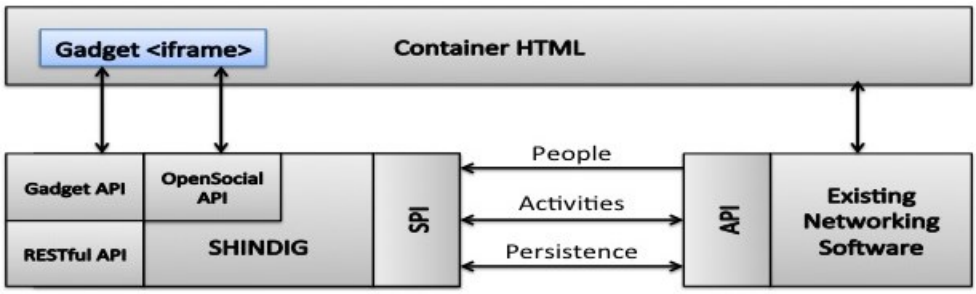

Figure 3.2 - OpenSocial Architecture, taken from [35]

3.3.1.3 SN Platforms Evaluation

Table 3.2 summarizes the key similarities and differences between the Facebook platform and Google's OpenSocial platform.

Table 3.2 - Comparison of Facebook Platform and Google OpenSocial

Facebook Platform	Google OpenSocial
Getting users data, sending notifications and publishing users activities	Getting users data, sending notifications and publishing users activities
Based on RESTful API	Based on RESTful API
URL addressable, REST-like server API	Client-side JavaScript oriented
Uses Facebook JavaScript (JBJS), Facebook Mark-up Language (FBML) and Facebook Query Language (FQL)	Uses JavaScript (JS), Hypertext Mark-up Language (HTML) and Structured Query Language (SQL)

This thesis will be based on Google's OpenSocial and it will reuse and build up on its resources for the following reasons:

- It allows the integration of the third party applications with most of the nowadays SNs.
- It uses widely used technologies (e.g. JavaScript, SQL & HTML).
- Most of the open-source projects run on top of the OpenSocial APIs.
- Taking into account our fourth requirement, which is to support reusing entirely/partially existing SN infrastructure. OpenSocial is a more suitable solution.

3.3.2 Differentiated QoS and SNs

The authors in [37] suggest a differentiated QoS scheme for social media delivery over mobile networks. They categorize the social media content into different classes

according to the QoS Service Level Agreement (SLA) Traffic Classes. These classes are: Conversational (e.g. Social Health CureTogether), Streaming (e.g. YouTube), Interactive (e.g. Social Game SecondLife) and Background (e.g. Twitter). The social media content is transmitted through the IP Multimedia Subsystem (IMS) then to Internetwork Packet Exchange (IPX) network until it reaches the application servers. The session established between the mobile device and the application server will have a specific QoS level (Platinum, Gold, Silver or Bronze). This can be done using Resource Reservation Protocol, Multiprotocol Label Switching (MPLS) and/or Differentiated Services (DiffServ). Figure 3.3 shows the architecture into which end-to-end QoS sessions are established while table 3.3 shows QoE service translations for different traffic classes of social applications.

Table 3.3 - QoE service translations for different traffic classes in [37]

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="4">QoE</td></tr>
<tr><td>Platinum</td><td>Gold</td><td>Silver</td><td>Bronze</td></tr>
<tr><td rowspan="4">QoS SLA Traffic Classes</td><td>Conversational</td><td colspan="2">Tolerable</td><td colspan="2">Non-Tolerable</td></tr>
<tr><td>Streaming</td><td colspan="3">Tolerable</td><td>Non-Tolerable</td></tr>
<tr><td>Interactive</td><td colspan="2">Tolerable</td><td colspan="2">Non-Tolerable</td></tr>
<tr><td>Background</td><td colspan="4">Tolerable</td></tr>
</table>

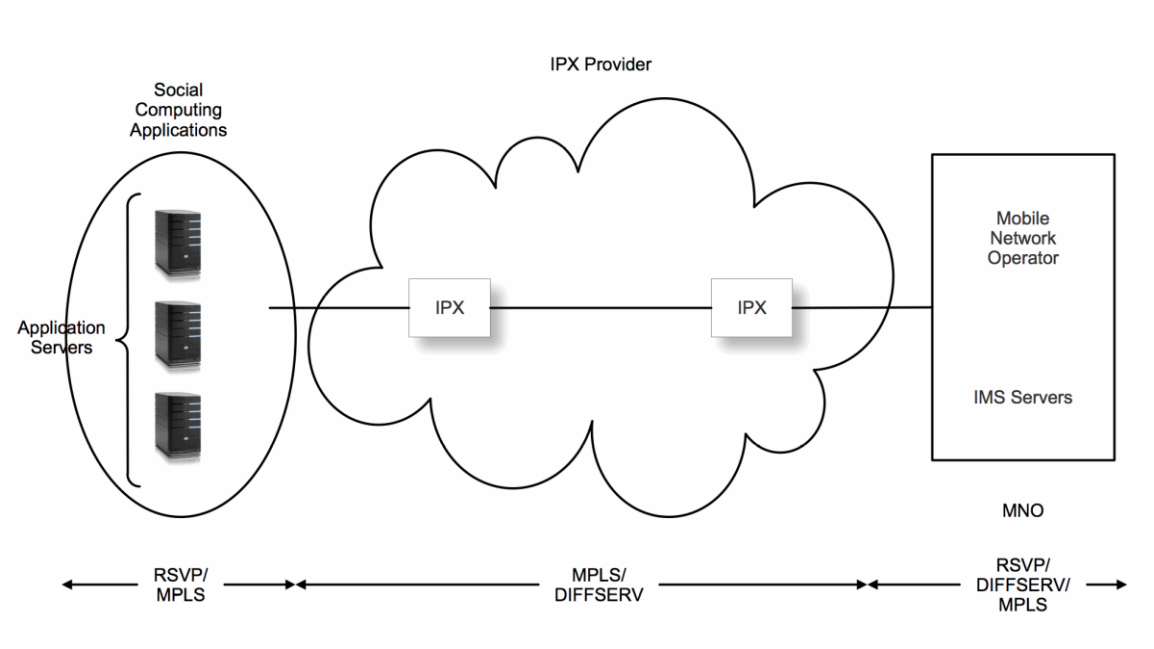

Figure 3.3 - Establishing end-to-end QoS in [37]

This work does not meet our third requirement of user-based differentiated QoS because it does not guarantee a user's requested QoS level (e.g. Silver) if network resources are not available, or if the user will use a social media content (e.g. Conversational), which is not supported by this QoS level. Moreover, it supports session-based differentiated QoS not user-based differentiated QoS. The first and second requirements are not discussed because this work is not related to filtering. However, this work meets our fourth requirement since it uses standard technologies to offer the differentiated QoS so it can reuse any existing social media application and build a differentiated QoS framework.

Some related work has been done in the E-health area regarding both the SN and QoS, e.g. in emergency response. A framework, called SenseFace that dynamically posts information from a user's Body Sensor Networks (BSNs) to his/her SN has been proposed in [38]. The BSN sends information to a smartphone device via Bluetooth, the

smartphone forwards information via a Wi-Fi/cellular network to the E-health service provider that interacts with the user's SN and makes critical medical/health decisions for the user. Figure 3.4 shows the SenseFace architecture. This work allows the prioritization of certain messages in SNs (e.g. emergency cases have the highest priority). However, it does not support differentiated QoS from the user perspective and it does not support different QoS levels for different users. The first and second requirements are not discussed because this work is not related to information filtering. Finally, this work does meet our fourth requirement since the E-health service provider can be added as an option to any existing SN solution.

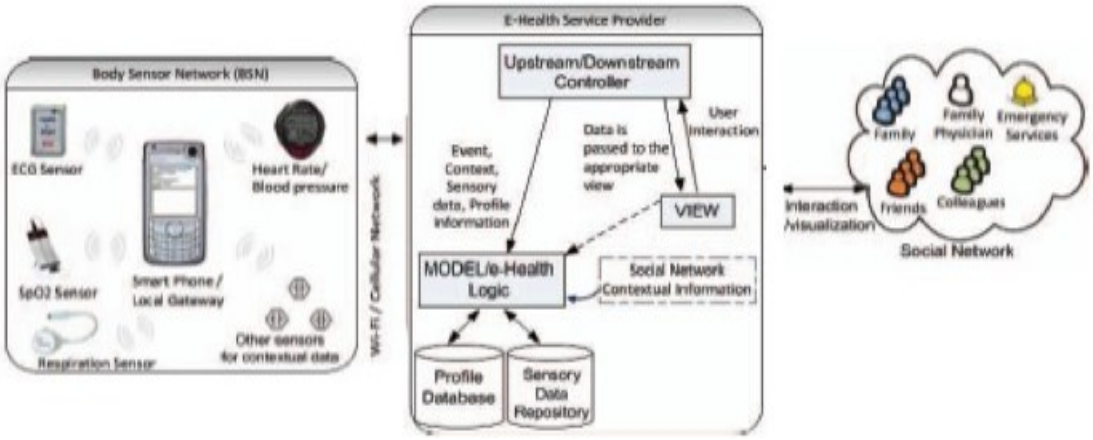

Figure 3.4 - SenseFace Architecture in [38]

Differentiated QoS is achieved by applying different QoS levels. Each QoS level (Gold, Silver or Bronze) defines a set of requirements to be met for the flows of the users that are subscribed to this QoS level. The Evolved Packet Core (EPC)-based Systems' new network aware services allow provisioning with differentiated and guaranteed QoS according to the users' preferences [5]. It meets our third requirement of supporting user-

based differentiated QoS. So, this can be used in our work. However, this requires a Service Delivery Platform (SDP), which will be discussed in the next subsection.

3.3.2.1 Differentiated QoS Service Delivery Platform (SDP)

The authors in [39] propose a new system architecture for video surveillance applications that runs on top of 3GPP 4G EPC-based systems. They used 3GPP 4G Evolved Packet Core (EPC) to enable guaranteed and differentiated QoS in mobile video surveillance applications. The main components of their proposed architecture are the Surveillance Server, the SDP (QoS and Streaming Enablers) and the Machine-to-Machine (M2M) network. The M2M gateway enables interactions with the M2M devices. While on the other hand, the SDP enables the development and management of QoS mobile video surveillance applications.

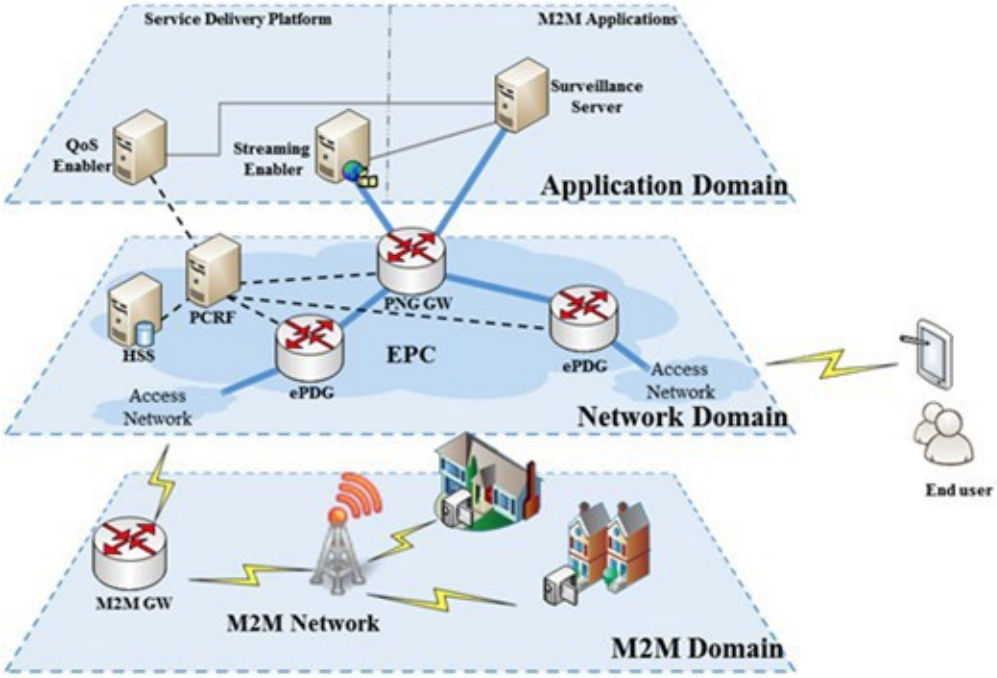

Figure 3.5 - The overall architecture in [39]

The QoS Enabler component in [39] provides the users with end-to-end differentiated QoS through the EPC layer. It meets our third requirement of providing user-based differentiated QoS. Moreover, it also meets our fourth requirement because the QoS Enabler can be reused with different systems to communicate with the EPC domain and provide differentiated QoS. However, the scope of their work is not in the SNs domain.

3.3.3 Information Filtering and Social Networks

Reference [40] discusses content-based filtering. It analyzes the features that are important for a user, to develop a means to filter the posts that appear in their SN news feed. It proposes an information filtering system based on the user's preferences. Figure 3.6 shows their information filtering system architecture. The training data are kind of raw data in order to prepare it for the test data. The output of the training data should not contain redundant or irrelevant data of the user. Their architecture focuses on the data pre-processing. It contains the set of inputs according to the user preferences. It contains set of features that transforms the data inputs to known features (e.g. topic of post). Also, it includes the database that collects the user data with the feature along with the decision of the user to allow or deny the data. Finally, it has the feature selection that extracts the set of features relevant to the user in order to take the appropriate decision. The test data uses the actual situation of the users and also uses edge rank algorithm of Facebook in the set of inputs and data mining algorithm in the feature selection component. This work does not meet our first filtering requirement of user-based filtering because it does not allow entire posts from certain users to be filtered. However, it does meet our second and fourth requirements. It provides a solution to existing SNs thus reusing them. It provides the SNs with content-filtering approach that matches the second requirement since it can

filter posts according to a certain topic. Finally, it did not discuss our third requirement of supporting differentiated QoS because it is out of the scope of their work.

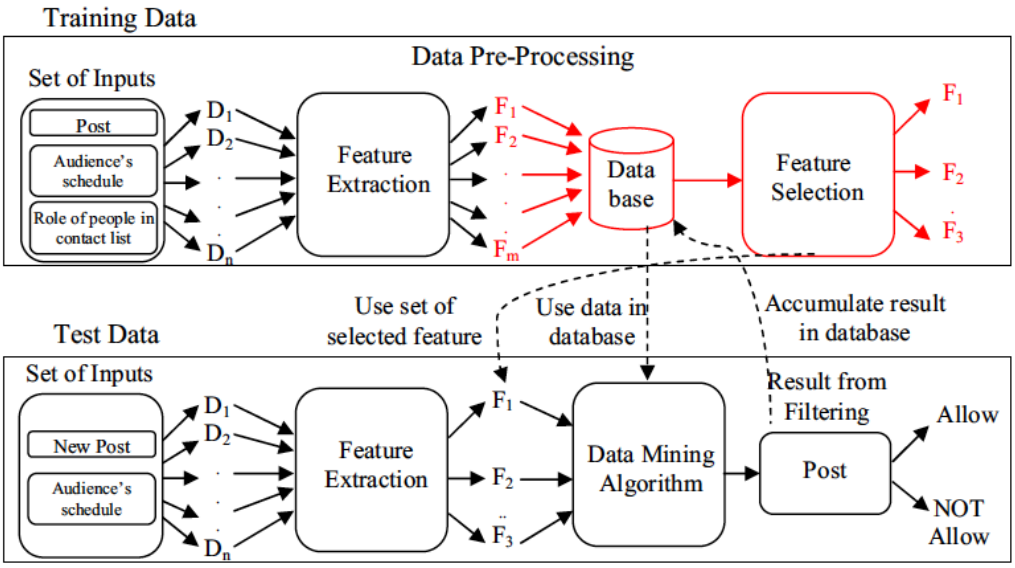

Figure 3.6 - Information Filtering System Architecture in [40]

An anti-spoiler system that changes the filtering system dynamically according to a user's preference is proposed in [41]. The system implements content-based filtering according to the user's current situation. The filtering system knows the intended user's action (e.g. watch a certain movie) from his/her previous actions (e.g. reservation of a movie ticket). After that, the system filters the content related to the action until the user does it (e.g. filter all posts related to this movie until the user sees it). Their filtering system has the following requirements:

- **User detection:** It detects the target users of the system.
- **Schedule detection:** The schedule of the users in order to determine their upcoming actions. For simplicity, the users add their own schedules in this work.
- **Activity detection:** The system has to detect the user current activity.

- **Filter Creation:** The system has to be able to make the filter automatically and set the content to be filtered at any given point of time.
- **Content division:** This implies the division of the text into different parts (e.g. Topic name).
- **Filtering:** This implies the actual detection of unwanted posts by the aforementioned filter.
- **Visualization:** The system, finally, has to prevent the filtered items from being displayed to the user.

The authors implemented their "anti-spoiler" system and met their requirements. This work does not meet our first requirement of supporting user-based filtering because the user cannot hide entire posts/requests from certain users in his/her SN. Furthermore, it does not support the fourth requirement of supporting the reuse of existing solutions. The proposed solution is specific to their system and cannot be seamlessly integrated with nowadays SNs. However, this work partially meets our second requirement of supporting content-based filtering because it supports temporarily content filtering of unwanted posts of the users. However, it does not support entirely the filtering of a specific content from the user's news feed. Finally, it does not discuss the third requirement of supporting user-based differentiated QoS because it is out of their work's scope.

Reference [42] presents a unified information-filtering model that prevents the user from being overloaded by information and offers him/her relevant information. It also provides the user with information that he/she did not yet know. This model deduces what the user is interested in. This is made possible by searching through users' friends of friend's profiles. These profiles will reveal overlapping interests, may indicate the same

educational institutions, and/or indicate income levels similar to those of the user. This filtering system may also discover users that are performing the same task at the same time or sharing similar posts at the same time as the target user. These authors discussed a case study demonstrating that the information shared by a user's friends is not always relevant, but that information augmentation (showing the user additional information) is useful for most users. More about information augmentation and their approach of reaching it can be found in [42]. This paper does not meet our first requirement of supporting user-based filtering because it focuses only on the content-based and collaborative filtering approaches. However, it partially meets our second requirement of supporting content-based filtering. It filters specific content from the user and provides him/her with additional information that is believed to be more relevant. It does not support entirely the filtering of posts related to a specific topic. Furthermore, this work does not discuss our third requirement because it is out of its scope. Finally, it meets our fourth requirement because it can be deployed on any existing solution.

There are more works discussing IF in SNs but this section presented the most relevant to our scope.

3.3.4 Overall State of the Art Evaluation

Table 3.4 shows the summary of all the related works (columns) in comparison to our four requirements (rows). Each related work is evaluated with respect to each requirement and this evaluation is assigned one of these values:

- **Not Applicable (N/A):** That means that the related work is out of the scope of the requirement.

- **Met:** That means that the related work meets the corresponding requirement.
- **Partially met:** That means that the related work partially meets the corresponding requirement.
- **Not met:** That means that the related work does not meet the corresponding requirement.

Table 3.4 - State of the art overall evaluation

Related Work / Require-ments	[37] A Convergent Framework for QoS Driven Social Media Content Delivery over Mobile Networks	[38] A Framework to bridge SN & body sensor network: An e-Health perspective	[39] EPC-Based System Architecture for QoS-Enabled Video Surveillance Applications	[40] Feature Selection Based on Audience's Behavior for Information Filtering in Online SNs	[41] Temporal filtering system to reduce the risk of spoiling a user's enjoyment	[42] IF and personalization: Context and group profile effects	Google's OpenSocial	Facebook Platform
Req. 1: User-based filtering	N/A	N/A	N/A	Not Met	Not Met	Not Met	N/A	N/A
Req. 2: Content-based filtering	N/A	N/A	N/A	Met	Partially Met	Partially Met	N/A	N/A
Req.3: User-based Diff QoS	Not Met	Not Met	Met	N/A	N/A	N/A	N/A	N/A
Req. 4: reusing existing solutions	Met	Met	Met	Met	Not Met	Met	Met	Not Met
	Differentiated QoS & SNs			**Information Filtering & SNs**				

Table 3.4 shows that Google's OpenSocial is more suitable than the Facebook Framework as a basis of our thesis. None of the differentiated QoS and SNs related works (Ref [37] and [38]) met all of our requirements. However, the use of Evolved Packet Core (EPC) systems would provide differentiated QoS to our thesis proposed architecture and partially reusing the SDP proposed in [39] would allow that. However, the work discussed in [39] does not focus on the SN perspective. Finally, none of the IF and SNs related works (Ref [40], [41] and [42]) met all of our requirements as well.

3.4 Chapter Summary

This chapter discussed the thesis motivating scenarios, then the derived requirements that a QoE-enabled SN should meet. In this chapter, we discussed four functional requirements for our solution. They are supporting user-based and content-based filtering, differentiated QoS and reusing existing solutions, if possible. After that, it presented the state of the art in the light of the requirements and provided an evaluation summary of the state of the art. None of the related work supported all our requirements. However, we concluded that Google's Open-Social platform is better than Facebook's platform to form the basis of our work and using EPC systems can provide the SN users with differentiated QoS. The next chapter will introduce and discuss our proposed architecture, its functional entities and interfaces, procedures and will provide an illustrative scenario to demonstrate the working of the architecture.

Chapter 4: QoE-Enabled SNs: The Proposed Architecture

This chapter presents the proposed architecture. The chapter is organized as follows: the first section presents the overall architecture. The second presents the different interfaces in the architecture and the RESTful resources used. The third presents the procedures that can be initiated by the users to use the functions of the proposed architecture. The fourth discusses an illustrative scenario that shows how the entities of the architecture interact using combined procedures. We conclude this chapter with an evaluation of our architecture with respect to the previously mentioned requirements.

4.1 The Overall Architecture of QoE-Enabled SNs

The overall architecture, shown in Figure 4.1, consists of two layers, the application and network layers. The application layer consists of the SN-Server, the QoS Enabler and the Database. The SN-Server, which is the core of this thesis, provides the information filtering and differentiated QoS functionalities. It also provides all the common functionalities of the SN, such as adding/removing friends and posting and retrieving updates. The Database contains all the users' profiles, the filtering policies as well as the supported QoS levels. The last component is the QoS Enabler, which is the service delivery platform (SDP) component that interacts with the EPC network in order to provide differentiated QoS to the users. The authors in [39] propose a new system architecture for video surveillance applications that runs on top of 3GPP 4G EPC-based systems. Our QoS Enabler partially reuses their work to communicate with the PCRF of the EPC layer and initiates/modifies users' sessions with end-to-end differentiated QoS.

The 3GPP 4G EPC is the network layer and the basis of the proposed architecture because it enables guaranteed and differentiated QoS.

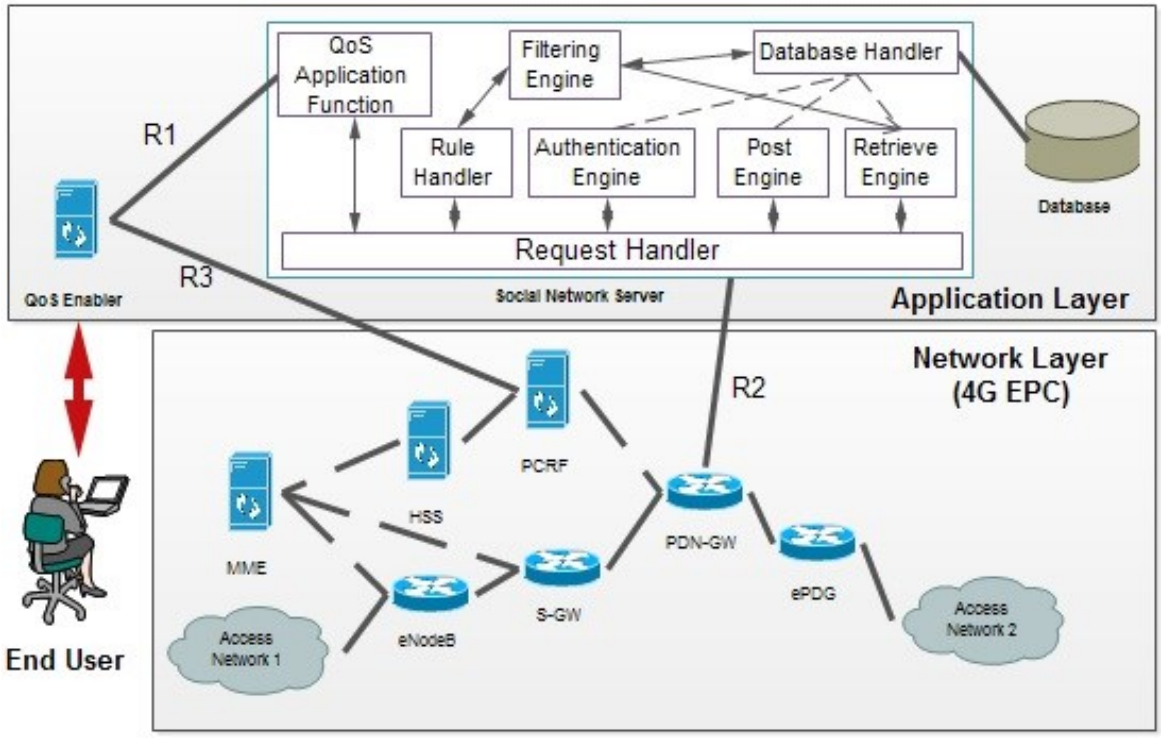

Figure 4.1 - The Overall Architecture of QoE-Enabled SNs

There are different functional entities that compose the overall architecture.

- **The Request Handler**: It is the contact point between the EPC network and the Application Layer components (SN-Server). It classifies the types of requests sent to the system and forwards them accordingly. In the case of a connection request (e.g. HTTP GET Request with username and password), it is forwarded to the Authentication Engine to authenticate the user and log him/her in. If it is not a connection request, it is forwarded to the Authentication Engine (along with the connection ID) for validation and when approved, it is forwarded to the appropriate engine.

- **The Authentication Engine**: It is responsible for the authentication, connection and validation of a user and his/her requests.
- **The Post Engine**: It handles the post requests sent by the user and save them into the database.
- **The Retrieve Engine**: It manages the user's retrieve requests. It gets the updates from the database and forwards them to the user after applying his/her filtering criteria. The user criteria are defined through rules, which are added, modified or removed by the Rule Handler.
- **The Rule Handler:** It adds, modifies or removes the users' rules and defines the user's criteria.
- **The Filtering Engine**: It has different functions, all related to the filtering options. It receives the updates and the user criteria from the database. It then implements the rule selection and rule enforcement filtering policies and finally forwards the output to the Retrieve Engine, which forwards the results to the user. The Filtering Engine also receives the rule updates from the Rule Handler and enforces the changes to the rules in the Database.
- **The QoS Application Function:** It is responsible for the initiation, modification (upgrade/downgrade) and teardown of users' sessions. It communicates with the QoS Enabler in order to incorporate the QoS policies in the EPC system and create the network layer sessions.
- **The QoS Enabler:** It makes decisions about whether to create/modify the sessions according to the available bandwidth and the corresponding priority of the users.

- **The Database Handler:** It is the interface between the SN-Server and the Database. It maps the requests to the database entities and communicates with the Database.
- **The Database:** It stores all of the users' profiles, the users filtering criteria as well as the supported QoS levels of the system and the user sessions.

4.2 Interfaces and REST Resources of the Architecture

This section illustrates the main interfaces in the proposed architecture and discusses the REST resources of the SN-Server and QoS Enabler. According to Figure 4.1, R3 is Rx Diameter interface [32], while R1 and R2 are RESTful interfaces. REST is favoured in this architecture because it is lightweight and easy to implement. It supports multiple data representations for resources (JSON, XML, etc.) and its mapping rules allow the server to implement a single interface. Furthermore, the SN architecture uses the OpenSocial framework [35], which uses a RESTful interface. OpenSocial framework is opted, as previously discussed, because many open-source projects run on top of OpenSocial APIs, e.g. LinkedIn, Google+. The next sections will provide the resource modeling for the SN-Server and the QoS Enabler.

4.2.1 The SN-Server Resources

Table 4.1 shows the resource modeling for the SN-Server. The resources of the SN-Server contain the OpenSocial API resources (Profiles, Messages, Groups, Application Data, Albums, Media Items and Activities) and the resources that are added to perform our SN-Server functionalities (added as child resources and their parent resource is the Profiles resource). Each resource has one or more operations and one or more

corresponding HTTP actions. For example, as shown in the following table, user Alice will send "POST:/SNSRoot/Profiles/ContentFiltering/Category/" with payload "sports" to the SN-Server in order to filter sports-related updates. Also, user Alice to hide user Bob's entire updates, for instance, will send "POST:/SNSRoot/Profiles/UserFiltering/ Hide/Updates/" with a payload value of "userId=Bob".

Table 4.1 - The SN Server REST Resources

Resource	Operation(s)	HTTP Action(s)
/Profiles/Content-Filtering	Get a list of the content filtering options in the SN-Server	GET:/Profiles/Content-Filtering
/Profiles/Content-Filtering/Category	Get a list of all the categories available that the user can filter	GET:/Profiles/Content-Filtering/Category
	Create a content filter for a specific category with a specific user	POST:/Profiles/Content-Filtering/Category
/Profiles/Content-Filtering/Category/{CategoryId}	Get the details of the specific Category action done by the user (e.g. Sports)	GET:/Profiles/Content-Filtering/Category/{CategoryId}
	Delete a category from a specific user's filtering criteria	DELETE:/Profiles/Content-Filtering/Category/{CategoryId}
/Profiles/User-Filtering	Get a list of the user filtering options in the SN-Server	GET:/Profiles/UserFiltering
/Profiles/User-Filtering/Hide	Get a list of the hiding options in the SN-Server	GET:/Profiles/User-Filtering/Hide
/Profiles/User-Filtering/Hide/AddRequest	Get a list of all the users that add request hiding can be applied to in the SN-Server	GET:/Profiles/User-Filtering/Hide/AddRequest
	Hide the add request from a specific user	POST:/Profiles/User-Filtering/Hide/AddRequest
/Profiles/User-Filtering/Hide/AddRequest/{userId}	Get the Boolean reply whether this specific user's add request is hidden or not	GET:/Profiles/UserFiltering/Hide/AddRequest/{userId}
	Delete the hide add request option towards a specific user	DELETE:/Profiles/UserFiltering/Hide/AddRequest /{userId}
/Profiles/User-Filtering/Hide/Updates	Get a list of all the users that updates hiding can be applied to in the SN-Server	GET:/Profiles/User-Filtering/Hide/Updates
	Hide the updates from a specific user	POST:/Profiles/User-Filtering/Hide/Updates

/Profiles/User-Filtering/Hide/Updates/{userId}	Get the Boolean reply whether this specific user's updates are hidden or not	GET:/Profiles/UserFiltering/Hide/Updates/{userId}
	Delete the hide updates option towards a specific user	DELETE:/Profiles/UserFiltering/Hide/Updates /{userId}
/Profiles/User-Filtering/Block	Get a list of all the users that blocking can be applied to in the SN-Server	GET:/Profiles/User-Filtering/Block
	Block a specific user	POST:/Profiles/User-Filtering/Block
/Profiles/User-Filtering/Block/{userId}	Get the Boolean reply whether this specific user's is blocked or not	GET:/Profiles/UserFiltering/Block/{userId}
	Delete the block option towards a specific user who is currently blocked	DELETE:/Profiles/UserFiltering/Block/{userId}

4.2.2 The QoS Enabler Resources

Table 4.2 shows the resource modeling of the QoS Enabler. We partially reuse some of the QoS-Enabler resources initially presented in [39]. Each resource also has operation(s) and HTTP action(s). The QoS Enabler resources are used to initiate, modify or delete user sessions with the EPC layer. For instance, the SN-Server will send "POST:/QoSEnablerRoot/SNS/Sessions/" to the QoS Enabler with payload "userID=Alice & ClassofServiceID=Gold" to initiate a session for user Alice with Gold QoS level, and the QoS Enabler will reply with the session number in the 201 OK message, in case of success. The SN-ID as a resource is provided because the QoS Enabler can run when it is associated with multiple SNs, which makes it an add-on to existing SN solutions.

Table 4.2 - The QoS Enabler REST Resources

Resource	Operation(s)	HTTP Action(s)
/ClassOfService	Get a list of all the classes of service supported by the QoS Enabler	GET:/ClassOfService
/SN-ID	Get a list of all the SNs supported within the QoS Enabler domain	GET:/SN-ID
	Create a domain of support by the QoS Enabler for a specific SN	POST:/SN-ID
/SN-ID/Sessions	Get the active sessions of a specific SN with the QoS Enabler	GET:/SN-ID/Sessions
	Create a new session with the QoS Enabler for a specific user of a specific SN	POST:/SN-ID/Sessions
/SN-ID/Sessions/ {sessionId}	Get the details of a specific session (e.g. userId, ClassOfService supported)	GET:/SN-ID/Sessions/{sessionId}
	Modify a specific session (e.g. upgrade the ClassOfService)	PUT:/SN-ID/Sessions/{sessionId}
	Delete a specific session	DELETE:/SN-ID/Sessions/ {sessionId}

4.3 User Initiated Procedures

Six procedures are defined for users: Login/Logout, Create/Delete Session, Post, Retrieve, Upgrade/Downgrade, and the Rule Add/Remove.

- **The Login/Logout procedure**: It starts with the user sending a connection/disconnection request to the Request Handler. The Request Handler forwards the user's request to the Authentication Engine to authenticate the user. The Authentication Engine checks with the Database whether the user is

registered or not. If so, it will reply to the Request Handler with the approval and the Connection ID to be forwarded to the user.

- **The Create/Delete Session**: The user sends a POST/DELETE message for the session creation/deletion with a specific QoS level. The Request Handler, after the user validation, forwards it to the QoS Application Function. The QoS Application Function communicates with the QoS Enabler to create/delete the session. The QoS Enabler sends a Diameter protocol message, which is an Authentication-Authorization Request (AAR) to the PCRF. The PCRF reserves the network resources and replies to the QoS Enabler with an Authentication-Authorization-Answer (AAA) message.
- **The Post procedure**: The user sends a POST request to the SN-server, which contains the update in the payload. After the user's validation, the Request Handler forwards the request to the Post Engine, which saves it in the Database and sends back an acknowledgment to the user.
- **The Retrieve procedure**: The user sends a GET message to the SN-server with a payload of the updates needed. The Request Handler, after the user's validation, forwards the request to the Retrieve Engine. The Retrieve Engine forwards the request to the Database Handler to retrieve the information from the Database. The Database Handler forwards the reply to the Filtering Engine along with the user's criteria. The Filtering Engine performs a rule selection process followed by a rule enforcement process and finally forwards the final reply message to the Retrieve Engine to reply to the user.

- **The Upgrade/Downgrade procedure**: The user sends a PUT message to the SN-Server along with the new QoS level subscription in the payload. The Request Handler sends the request to the QoS Application Function. The QoS Application Function communicates with the QoS Enabler to enforce the update. At the end, the QoS Enabler communicates with the PCRF to carry out the QoS level update.
- **The Rule Add/Remove procedure**: The user sends a POST request, which contains the rule addition, modification or removal in the payload, to the SN-Server. The Request Handler forwards it to the Rule Handler, which puts the payload in the appropriate filtering rule format and forwards it to the Filtering Engine, which in turn sends the rule to the Database Handler to be saved. The rule format is derived from that in the firewall domain [43] and used "<Order> <SourceID> <DestinationID> <Period> <Topic> <Action>". The user for simplicity provides the order. The SourceID usually is the userID that the rule will be applied to. The DestinationID is always the userID of the user initiating the filtering process. The period is the specified time that the rule will be applied within. The Topic is filled when it is the case of content filtering. The Action is either "allow" or "deny".

4.4 An Illustrative Scenario

In this section, the operations of the proposed architecture are illustrated with the following scenario: Alice (professor), Charlie (student) and Bob (professor) are members of a SN. Alice provides a user criterion to avoid receiving friendship requests from students and hide sports-related posts. When Charlie and Bob send her a friendship request, Alice will only receive the friendship request from Bob. The sequence diagram in Figure 4.2 shows the operations and the different messages exchanged. Users Bob & Charlie send friendship requests to user Alice using the Post procedure (Steps 1 to 4). Bob and Charlie's POST requests are realized by the Request Handler and saved to the Database through the Post Engine. Next, user Alice connects to the SN using the Login (Steps 5 to 8) and Create Session procedures (Steps 9 to 14). User Alice sends a GET message, which is realized by the Request Handler. The Request Handler forwards it to the Authentication Engine for authentication and then user Alice receives a Connection ID. A POST message is sent by Alice to create a session with her desired QoS level (Gold). This request is realized by the Request Handler, and then forwarded to the QoS Application Function. The QoS Application Function contacts the QoS Enabler to communicate with the PCRF and reserve the user session. Finally, Alice retrieves only the friendship request from user Bob using the Retrieve procedure (steps 15 to 19). This occurs after Alice sends a GET request to the SN-Server. The Request Handler sends it to the Retrieve Engine. The Retrieve Engine forwards it to the Database Handler to get the user updates and criteria from the Database. The reply passes through the Filtering Engine to apply Alice's criterion (filter friendship requests from students) and is finally forwarded to her.

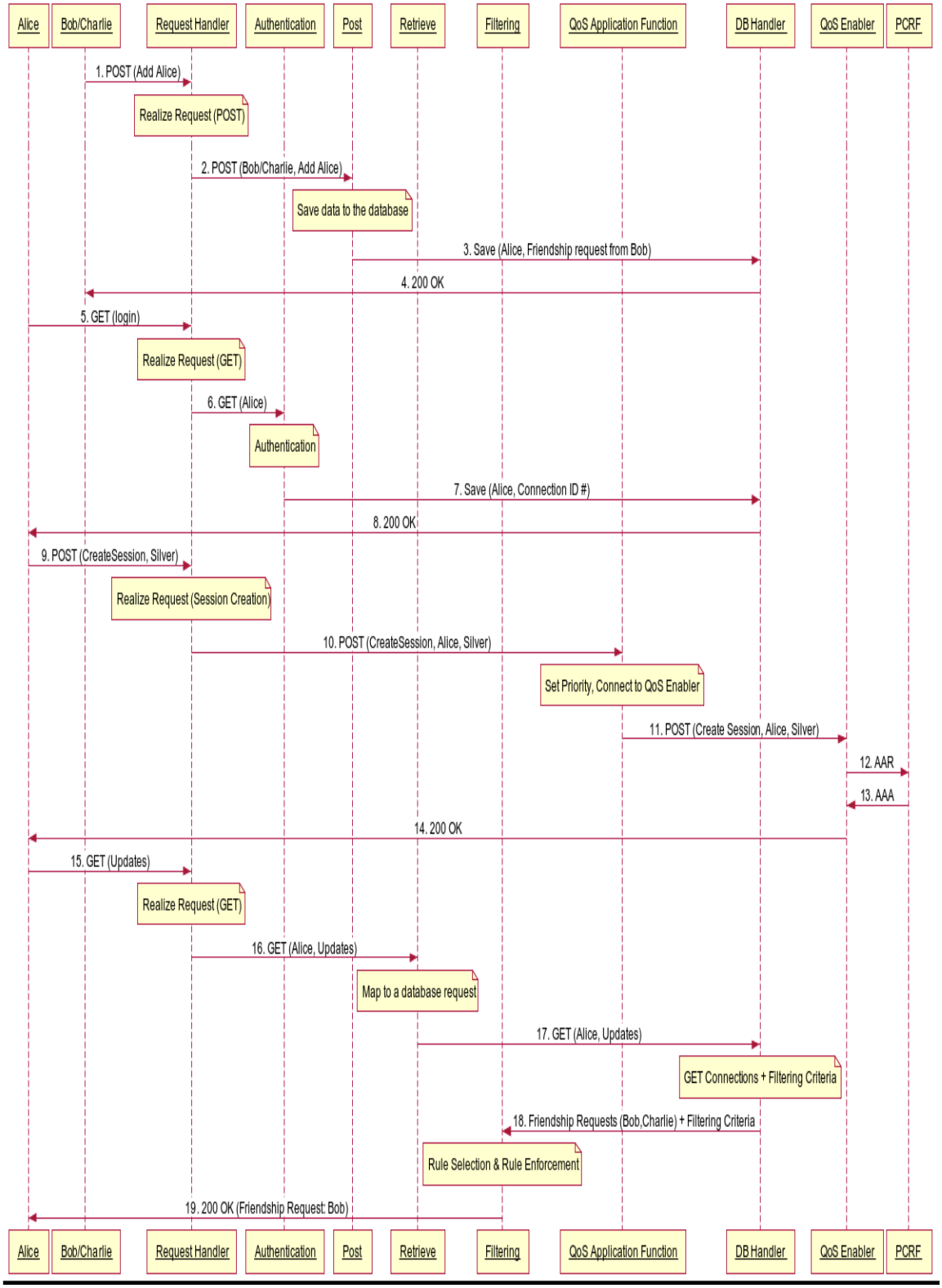

Figure 4.2 - An Illustrative Scenario for the operations of the proposed architecture

4.5 Chapter Summary

This chapter presented our novel proposed architecture. The architecture consists of application and network layers. The application layer contains the SN-Server, QoS Enabler (a SDP) and Database. The architecture runs on top of the 3GPP 4G EPC systems, which is the network layer, to provide end-to-end user-based differentiated QoS. The interfaces that link the architectural components are RESTful and Diameter interfaces. After that, the chapter identifies the REST resource modeling of the SN-Server and the QoS Enabler, their operations and HTTP Actions.

In the previous chapter we designed certain requirements and our proposed architecture satisfies all of them. Our architecture meets the first (user-based filtering) and second (content-based filtering) requirements through the Filtering Engine in the SN-Server. The Filtering Engine allows the rule definition, selection and enforcement during the retrieval of information by the user. The rule, provided by the user, enables him/her to hide posts/requests from other users and/or posts related to certain topics. Furthermore, the architecture meets the third requirement (Support user-based differentiated QoS) through the EPC layer, which acts as the differentiated QoS provider. The SN-Server's QoS Application Function alongside with the QoS Enabler controls the user sessions and QoS levels. Finally, the architecture meets the fourth requirement (re-using existing solutions) because the proposed architecture can be used, as an extension, to any existing SN. The components of the architecture act as add-on solutions to existing SN-related works.

The next chapter will discuss the implementation and evaluation of the QoE-Enabled SNs, the software architectures of the SN-Server and the QoS Enabler, and the proof of concept prototype implementation and partial evaluation.

Chapter 5: QoE-Enabled SNs: Implementation and Evaluation

This chapter discusses the implementation and evaluation of the proposed architecture. It starts by presenting the software architectures of the SN-Server and the QoS Enabler. After that, it presents a simple content filtering algorithm used in the SN-Server. Finally, it discusses the proof of concept prototype implementation and evaluation.

5.1 Software Architectures for the Proposed Solution Components

5.1.1 SN-Server Software Architecture

Figure 5.1 shows the Software Architecture of the SN-Server and its components.

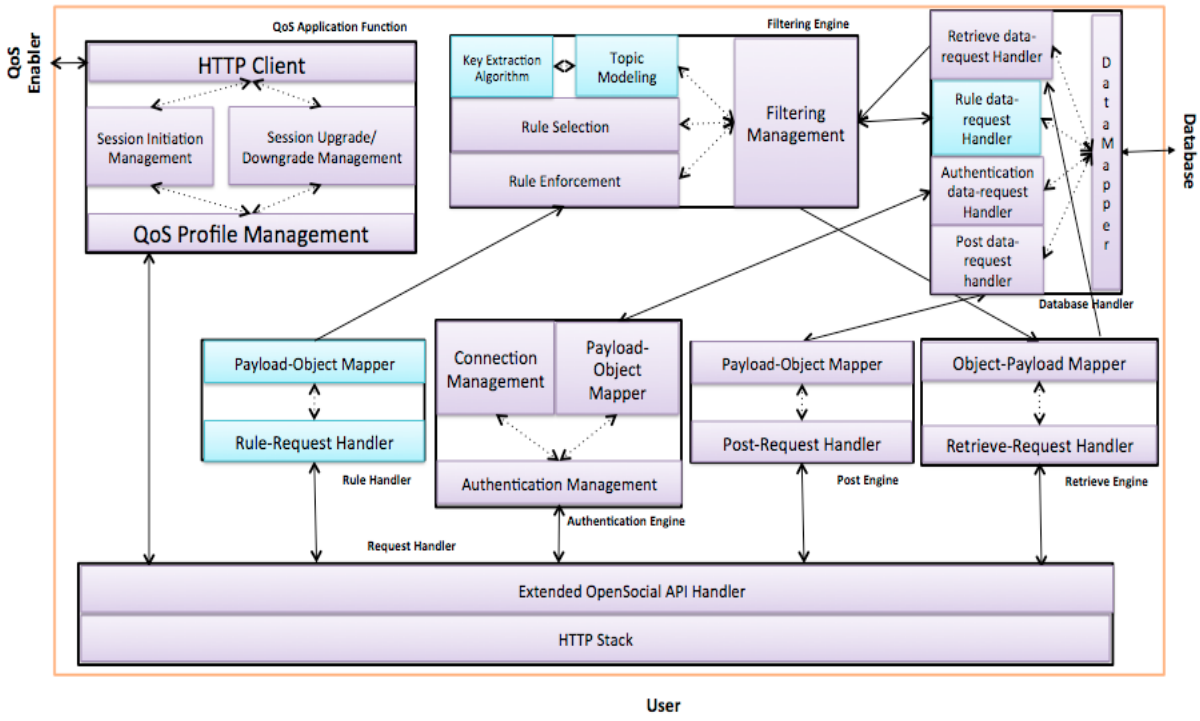

Figure 5.1 - The SN-Server Software Architecture

- **The Request Handler** consists of two components, the *HTTP Stack* and the *Extended OpenSocial API Handler*. The HTTP Stack receives the HTTP messages from the users and analyzes them. The Extended OpenSocial API Handler allows the Request Handler to know if a message has been validated and to which destination the Request Handler should forward the message.
- **The Authentication Engine** contains three components: the *Authentication Management, Connection Management* and *Payload-Object Mapper*. The Authentication Management component recognizes if a message is a login message or a validation message. If it is a message to be validated, it will be forwarded to the Connection Management block, which has a list of the current connection IDs and replies back to the Authentication Management component with an acknowledgment. The Payload-Object Mapper translates the payload of the login message into the system object language and forwards it to the Database Handler.
- **The Post Engine** contains the *Post-Request-Handler*, which analyzes the post message received from the user. The Post Engine also has a *Payload-Object Mapper* to change the payload of the post message to the system object language and then forwards it to the Database Handler.
- **The Retrieve Engine** and the **Rule Handler** have the exact same functionality as the Post Engine, but on the retrieving and managing rules sides, respectively. The Rule Handler communicates with the Database through the Filtering Engine. An example of the user filtering criteria request to the Rule Engine follows: <1> <Charlie> <Alice> <8am-5pm> <Sports> <Deny>. This example says that user

Alice will filter all sports-related updates from user Charlie between 8 am and 5 pm. The user, for simplicity, will provide the order of the rules.

- **The Database Handler** contains a *Data-Request Handler* for each engine. The *Retrieve-Data-Request Handler* forwards the reply message to the Filtering Engine with the user's data and criteria. The *Data-Mapper* is the interface between the system objects and the database objects. Utilizing this interface makes the system adjustable to different databases by merely editing the Data-Mapper.
- **The QoS Application Function** contains the *QoS Profile Management*, which analyzes and forwards the request. The analyzed request is forwarded to the *Session Initiation Management* or *Session Upgrade/Downgrade Management*. The *HTTP Client* communicates with the QoS Enabler by sending and receiving HTTP messages through the RESTful interface.
- **The Filtering Engine** contains the *Filtering Management* that communicates with the Database to retrieve the user updates and criteria. The Filtering Management communicates with the *Rule Selection* to analyze the rule, and with the *Rule Enforcement* to apply the rule. The Rule Enforcement module also receives the rule addition, adjustment or deletion request from the Rule Handler, applies it and forwards it to the Filtering Management to save in the Database. To determine if a content-based filtering category applies to a post, the *Topic Modeling* module receives the post and the topic. It then provides the post to the *Key Extraction component*, which replies by extracting keywords from the post. In this thesis, "Keyphrase Extraction Algorithm KEA" [46] is used to identify the keywords

from every post. According to these keywords, the Topic Modeling module will decide whether to filter the post or not. This decision is made through a simple Content Filtering Algorithm that will be presented in the next subsection.

5.1.2 A Simple Content Filtering Algorithm

A simple Content Filtering Algorithm is provided for the Topic Modeling component of the SN-Server presented in Figure 5.2. It is used to decide if a post should be filtered from the user's news feed or not. This decision is based on the keywords that are extracted by the KEA tool [46]. However, the proposed architecture allows for the usage of any content filtering algorithm or tool.

The algorithm is summarized as follows. The algorithm is given an activity sentence 'K' (e.g. FIFA is considering adding a new referee to be present in football matches), a topic of filtering 'T' (e.g. sports) and a set of critical keywords 'C' (e.g. FIFA, sports, football, basketball, volleyball, referee). The K and T are sent to the KEA tool and in return the algorithm receives a set of keywords (e.g. FIFA, referee and football). The algorithm will allow (i.e. the post will not filtered) the post, if no keywords were returned. Otherwise, the algorithm will test if the keywords construct at least 5% of the original post. If so, the post will be denied (i.e. filtered). Otherwise, each keyword will be compared to all the C keywords. If it matches any of them, the post will be denied (i.e. filtered).

```
Get Activity_Sentence 'K'
Get Topic_of_Filtering 'T'
Get Critical_Keywords 'C'
Send K & T to "Key phrase Extraction Algorithm - KEA"
Receive Keywords 'N'
IF N = 0
        RETURN NO
END IF
IF N/K >= 5%
        RETURN YES
ELSE
        IF N >= 1
        RESULT = NO
        For each Keyword {
                For each Critical_Keyword 'CK' {
                        IF Keyword is equal to CK
                        RESULT = YES
                        End IF
                } End FOR
        } End FOR
        RETURN RESULT
        END IF
END IF
```

Figure 5.2 - A simple Content Filtering Algorithm

Keyphrase Extraction Algorithm (KEA) is an open-source software implemented in Java and platform independent. Digital Libraries and Machine Learning Labs, Computer Science Department of the University of Waikato in New Zealand, established KEA. It is used for Keyphrase indexing. The software is responsible for receiving text phrases and extracting keywords from this text according to a certain topic [46]. The topics are provided earlier in form of vocabularies. The software tool learns by time and gives more accurate results with usage. KEA is used in our thesis because it provides the software architecture with keywords according to certain topics and helps the content filtering algorithm to take the right decision whether to allow the post or deny it.

5.1.3 QoS Enabler Software Architecture

Considering the QoS Enabler, we re-use the software architecture in [39], and add a Session Manager component to the software architecture's Service Layer in [39]. This Software Architecture, shown in Figure 5.3, consists of three layers.

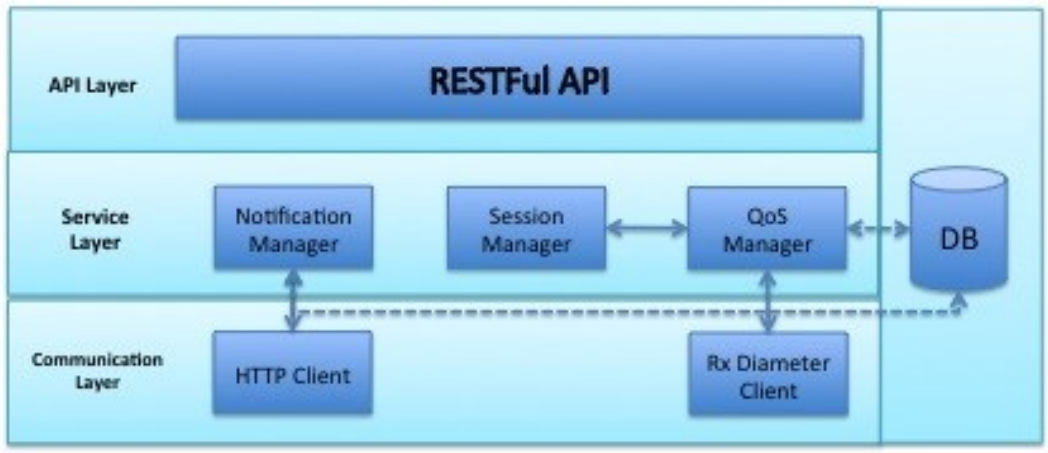

Figure 5.3 - The QoS Enabler Software Architecture

- The first layer is the **API Layer**, containing the *RESTful API* for user-developers.
- The second layer is the **Service Layer**, containing both the *QoS Manager* and the *Session Manager*. The QoS Manager receives messages from the SN-Server for the creation, modification or teardown of sessions in the EPC network. It communicates with the Session Manager that determines the confirmation. The Session Manager acts upon a maximum bandwidth for the SN-Application in the EPC layer in order to make the confirmation decision. It divides the bandwidth among the available QoS levels, in which each QoS level has a percentage of the available bandwidth. The Gold QoS level subscription has priority and majority of the bandwidth compared to the Silver QoS level subscription and the least priority

would be the Bronze QoS level subscription. The *Notification Manager* is not used in our work because its functionalities are out of the scope of the thesis.

- The Last Layer is the **Communication Layer**, which has two components. The *Rx Diameter Client*, which communicates with the PCRF, and the *HTTP Client*, usually used to send HTTP messages as instant response messages to the main Server. The HTTP Client is also not used in the thesis scope.

5.1.4 An Operational Procedure

This operational procedure illustrates how the software architectures of SN-Server and the QoS Enabler will work together to achieve the Session Creation procedure. How it would operate using the previously mentioned software architectures/components is shown in the sequence diagrams of Figure 5.4 and Figure 5.5.

User Alice sends a POST request to the SN-Server in order to create a session with QoS level: Gold. In steps 1-9, The HTTP Stack in the Request Handler first realizes the request. Then the Extended OpenSocial API Handler takes the decision whether this user should be validated or not. The username is then sent to the Authentication Management of the Authentication Engine for validation. The Connection Management component returns the decision to the Authentication Management component whether the user has a Connection ID or not. The result is then returned to the Extended OpenSocial API Handler, which forwards the POST request to the QoS Profile Management of the QoS Application Function. The QoS Profile Management realizes the request and knows that it is a Session Creation request. After that, it forwards the request to the Session Initiation Management component that constructs the proper RESTful request to be sent to the QoS Enabler through the HTTP Client. In steps 10-17, The RESTful API of the QoS Enabler

then receives the request, and forwards it to the QoS Manager component. The QoS Manager checks with the Session Manager if a Gold session can be admitted or not. After that, the QoS Manager gets the QoS profile of Alice from the database and updates it. Finally, it triggers the Rx Diameter Client to send an Authentication-Authorization Request (AAR) to the PCRF. In steps 18-26, Authentication-Authorization Answer (AAA) is sent from the PCRF and the 201 OK responses is sent back step by step respectively until it reaches Alice and the session starts.

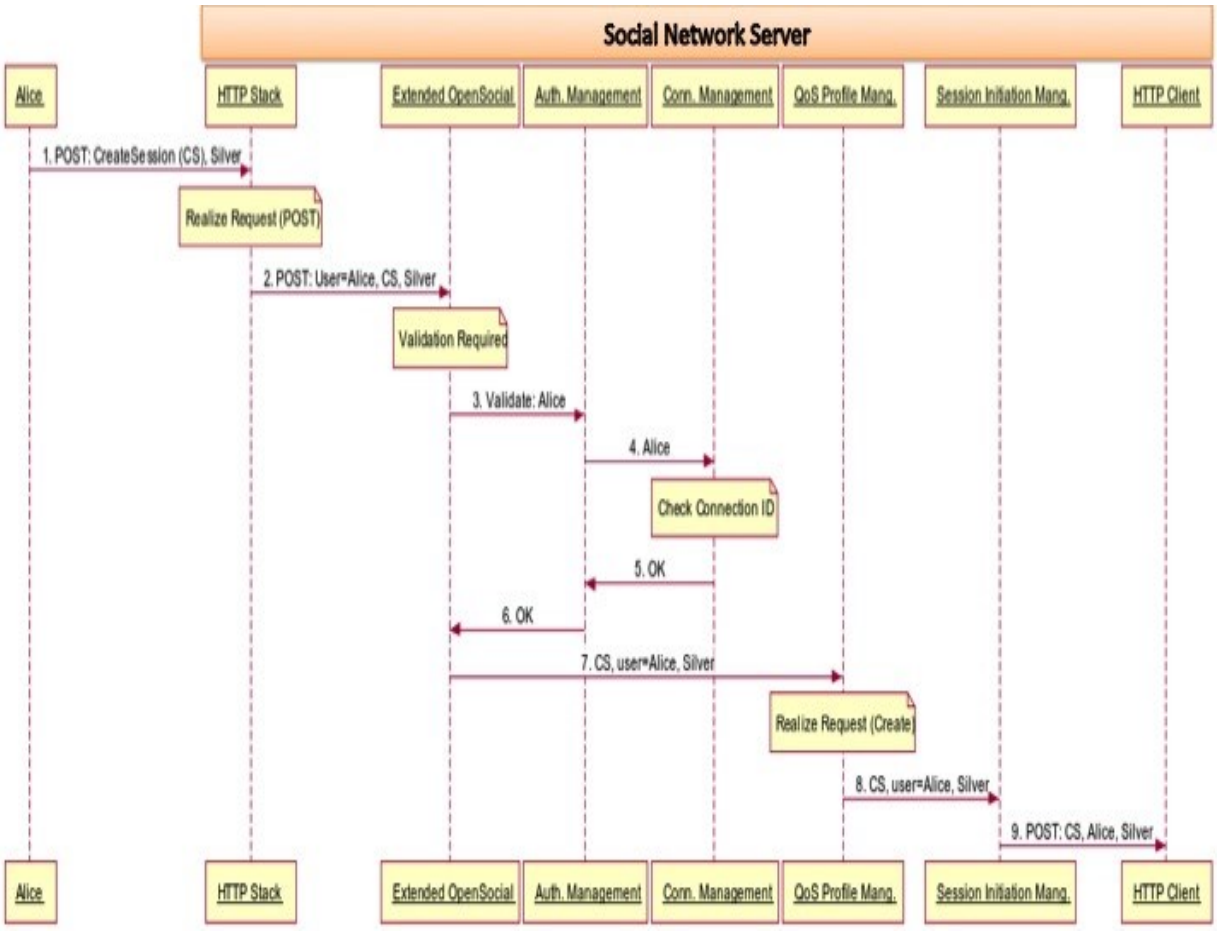

Figure 5.4 - Part one of the operational example sequence diagram for using the software architecture components of the SN-Server and the QoS Enabler

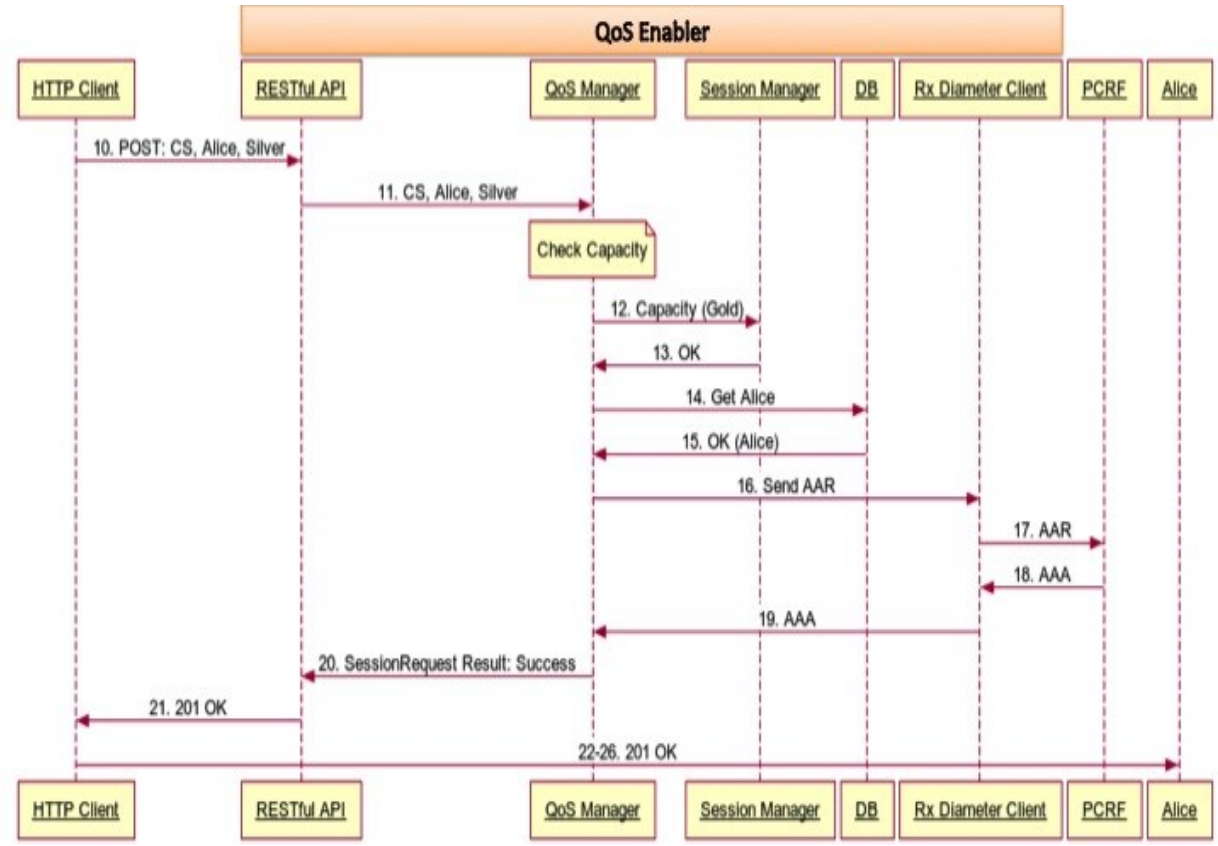

Figure 5.5 - Part two of the operational example sequence diagram for using the software architecture components of the SN-Server and the QoS Enabler

5.2 The Proof of Concept Prototype

A proof of concept prototype has been implemented for user-based and content-based information filtering and to offer user-based differentiated QoS to the SN users. The prototype runs on top of the 3GPP 4G EPC network. This section presents the prototype functionalities, prototype architecture, the experimental setup and environment that were used and finally it discusses briefly the tools and libraries used in the prototype implementation.

5.2.1 Prototype Functionalities

The implemented prototype provides its users with the following functionalities:

- A SN where users can:
    - Add/remove friends
    - Post statuses
    - Retrieve status updates of their friends
- The SN runs on top of the 3GPP 4G EPC Network.
- The SN allows each user to have a certain QoS Profile with EPC (e.g. Gold, Silver or Bronze).
- The SN allows its users to filter friendship requests from certain category of users (e.g. according to profession: Students) as an implementation of user-based filtering.
- The SN allows its users to filter their friends' updates according to certain topics (e.g. filter Food and Agriculture related posts).

5.2.2 Prototype Architecture

Figure 5.6 shows the prototype architecture. The client(s) are connected to the EPC network, for example via Wi-Fi through the ePDG component. The PDN-GW is the gateway to the application layer components and forwards the client(s) request(s) to the SN-Server. The SN-Server communicates with the QoS Enabler and the database. The QoS Enabler initiates, modifies and terminates the sessions by communicating with the PCRF. Figure 5.7 shows the prototype architecture of the SN-Server. ZING is an open-source implementation of a SN site based on an Apache Shindig framework and OpenSocial APIs [47]. Some features are added to ZING to be able to post (i.e. Post/Friendship-Request Handler) and retrieve (i.e. Retrieve-Request Handler). While retrieving, the Retrieve-Request Handler communicates with the Filtering Engine in order

to implement the filtering policies. The QoS Application Function is added to ZING for providing the user-based differentiated QoS and communicating with the QoS Enabler. The Content Filtering Algorithm discussed in section 5.1.2 is implemented in the Topic Modeling component. KEA tool is used to extract keywords [46]. The filtering tables are added to the same MySQL database used by ZING.

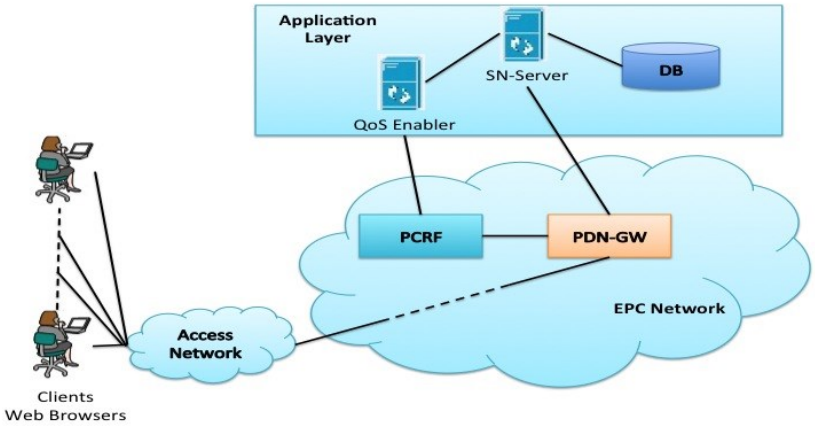

Figure 5.6 - The Prototype Architecture of QoE-Enabled SNs

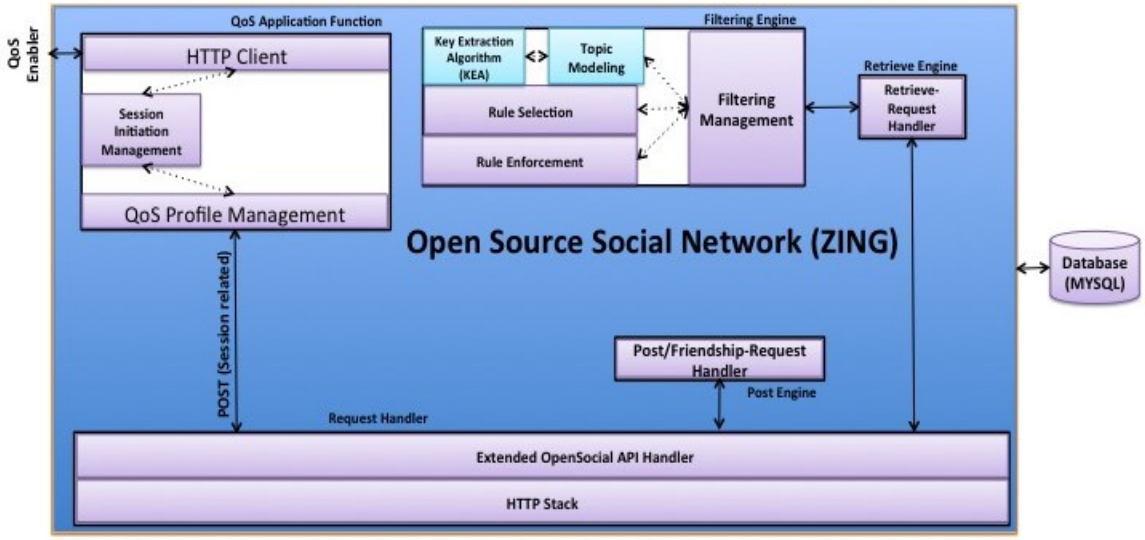

Figure 5.7 - The Prototype Architecture of the SN-Server

5.2.3 Tools and Libraries Used

The SN-Server runs on top of Fraunhofer Fokus OpenEPC release 2 as an implementation of the 3GPP 4G EPC [49]. OpenEPC consists of six virtual machines, which are: Client, S-GW, eNodeB, ePDG, PDN-GW and EPC Enablers. Each virtual machine is Ubuntu based virtual machine and they communicate with LAN segments. Our thesis prototype uses the ePDG, PDN-GW, EPC Enablers and duplicates the Client virtual machine to three Client virtual machines in order to support multiple clients (each Client virtual machine contains users from the same QoS level). OpenEPC partially implements the PCC architecture. It supports QoS Control. However, it does not implement the Charging control. The REST interfaces are implemented using the RESTLET framework [50]. The QoS Enabler is implemented using the JavaDiameterPeer library [51]. The client is simply a web-browser and this prototype uses Mozilla Firefox [52]. The SN-Server and the QoS Enabler are implemented using JAVA programming language using the Eclipse IDE [53].

5.2.4 Experimental Setup

The experiment runs on a local desktop environment using six virtual machines. The desktop has 4 gigabytes (GB) random access memory (RAM). The six virtual machines are: three Client virtual machines, ePDG, PDG-GW, EPC Enablers. The SN-Server, QoS Enabler, PCRF and HSS are inside the EPC Enablers virtual machine. All the virtual machines run on VMware Workstation 8 [48]. Multiple users from the three client virtual machines connect to the ePDG virtual machine via Wi-Fi. Figure 5.8 shows the login page displayed by the SN-Server when a user requests it.

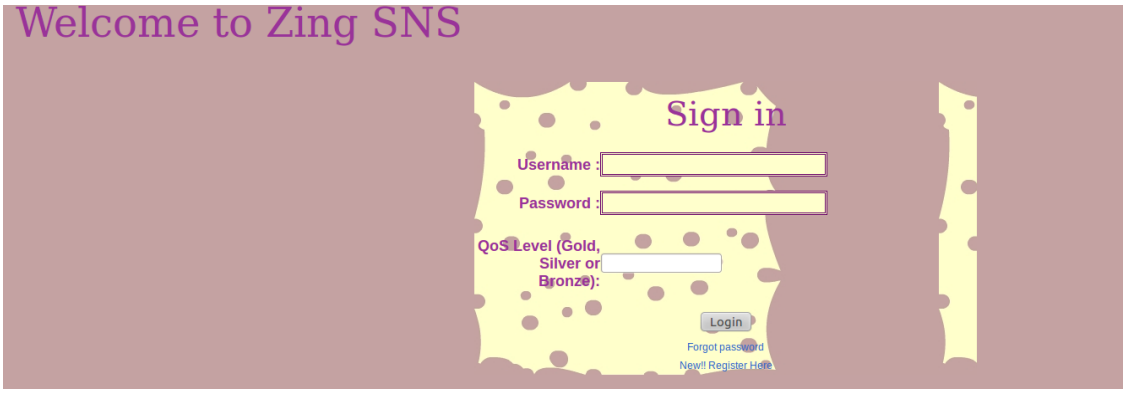

Figure 5.8 - SN-Server Login Page

The SN-Server has 100 Mbps as the maximum bandwidth for all the users' sessions. This bandwidth is divided among the Gold, Silver and Bronze QoS levels. In case of congestion, the Gold, Silver and Bronze has the highest, medium and lowest session admission priorities, respectively. Gold, Silver and Bronze sessions have a maximum bandwidth of 160, 80 and 40 Kbps respectively. The SN-Server can terminate a session from a certain QoS level (e.g. Bronze) for the admission of another session with higher QoS level (e.g. Gold or Silver). Table 5.1 shows the QoS profiles for the Gold, Silver and Bronze in Fraunhofer Fokus OpenEPC [49] and their attributes.

Table 5.1 - QoS levels in OpenEPC and their Attributes

Attribute	Gold QoS level	Silver QoS level	Bronze QoS level
Service Identifier	Gold	Silver	Bronze
QCI	5	6	7
Reservation Priority	1	2	3
Media Type	Text	Text	Text
Max Bandwidth	160 Kbps	80 Kbps	40 Kbps

5.3 Performance Evaluation

5.3.1 Evaluation Scenario

Figure 5.9 shows a sample SN-Profile for user Alice. The users can specify their filtering criteria by editing their profiles (e.g. Filter students and Food & Agriculture related topics as seen in the Figure). The illustrative scenario presented in Chapter 4 is implemented. The only difference is that the filtering topic is changed to Food and Agriculture related posts. The KEA tool [46] needs an implemented vocabulary in order to build its extraction model and then it extracts the keywords from any text. The Food and Agriculture Organization of the United Nations provides a controlled vocabulary on this topic and calls it "Agricultural Thesaurus Agrovoc" [55]. We use and provide it as an input for the KEA tool (Instead of building an amateur implementation of sports vocabulary). Moreover, multiple users from each QoS level subscription (Gold, Silver and Bronze) are to be active in the evaluation experiment. The Bronze users will send continuous session creation requests to the SN-Server followed by the Silver users and finally by the Gold users. This is done using a Sample Java File with the appropriate parameters. An evaluation of the bandwidth allocation of the system according to the QoS levels will be done.

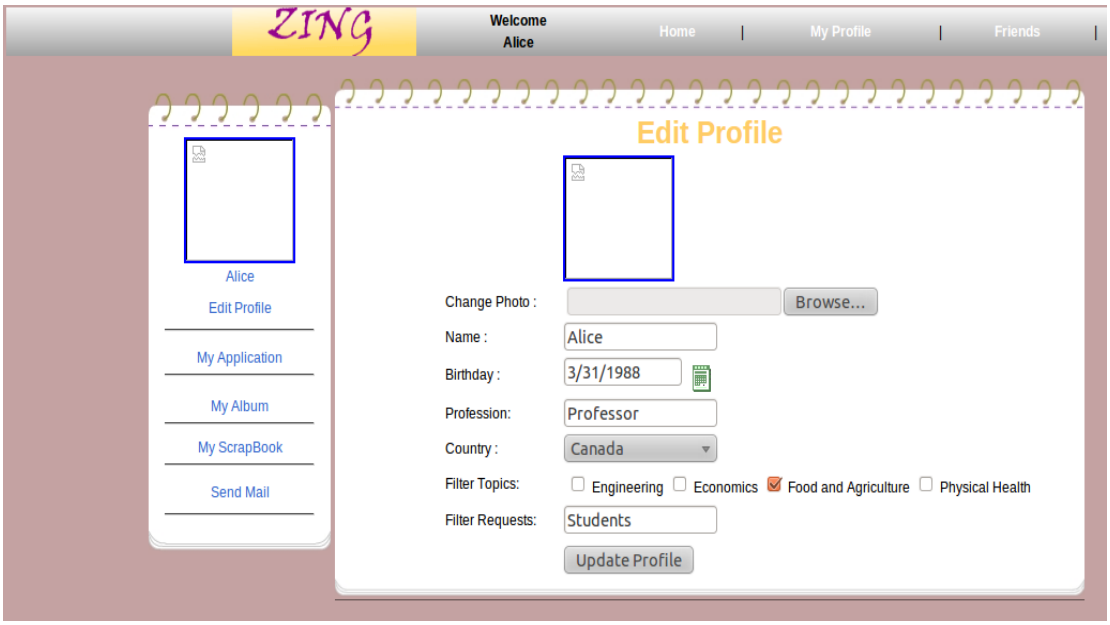

Figure 5.9 - Alice's SN profile

5.3.2 Performance Metrics

The following metrics are the performance metrics used to evaluate the prototype. They are intended to show the feasibility of our QoE-Enabled SN:

- **Bandwidth Allocation per each QoS level.** The bandwidth allocation per QoS level is the percentage of the bandwidth occupied by it. It will be interesting to see how the system deals with different QoS levels sessions and the occupied percentage of the total bandwidth per each QoS level. An experiment to be done showing the allocation of bandwidth for each QoS level over time. The total time is 12 minutes and the data are read every 5 seconds. This metric is used to show if the system will favour higher priority users over lower priority ones or not.
- **End-to-end session creation delay**. The end-to-end session creation delay is the total time spent between, first, sending the connection request for login to the system and the establishment of the session with the EPC layer according to the

user's desired QoS level. An experiment to be done showing the variation of the end-to-end session creation delay per user, on average, over time. The total time is 12 minutes and the data are read every 5 seconds. This metric is used to show if the average session creation delay throughout the experiment is acceptable or not, given that the user will benefit from his/her desired QoS level during the session.

- **Latency** of the Content-Filtering System. It is the overhead delay experienced by the system due to the content-filtering component. A result graph to be shown indicating the filtering delays against the number of posts to be filtered. The variation of the number of posts will be between 1 and 100 post(s). This metric is used in order to show the variation of the filtering delays affected by the number of messages to be filtered and if it is acceptable or not, given that the users will benefit from hiding their undesired data.
- **Accuracy** of the Content-Filtering System. It is the percentage of successful filtered messages that were forbidden to pass through the filtering component and are not displayed to the user. A result graph to be shown indicating the percentage of successful filtering against the attempt number. There are ten attempts. This metric is used to show if the filtering system enhances/learns with usage or not.

5.3.3 Performance Evaluation Results

Each experiment was repeated 10 times and the average results are presented in the same order as the performance metrics section.

- Figure 5.10 shows the **bandwidth allocation** over the time of the experiment. The graph shows that the Bronze sessions were admitted successfully until they occupied all the bandwidth available for the SN-Server with EPC (100 Mbps).

After that there was a period of Bronze session rejections by the QoS Enabler. Then the Silver sessions started to be admitted to the system. On the other hand, forced session terminations were ongoing for the previously established Bronze sessions. From 650 seconds until 720 seconds, the Silver sessions were rejected to favour the Gold session admissions and the Bronze sessions were continuously terminated. At 720 seconds, when the Bronze bandwidth reached the minimum possible (15%), the Silver sessions started to face forced session terminations in favour of the Gold session admissions. Finally, all the QoS levels reached a maximum bandwidth and the SN-Server started to reject all the session creation requests sent from all the QoS levels. The graph shows that the system is reliable. The system gives the highest priority to the Gold sessions followed by the medium priority for the Silver sessions and the lowest priority for the Bronze sessions. The Gold sessions, since they have the highest priority, did not undergo forced session termination. But they faced session rejections in order to keep part of the available bandwidth for the Silver and Bronze sessions.

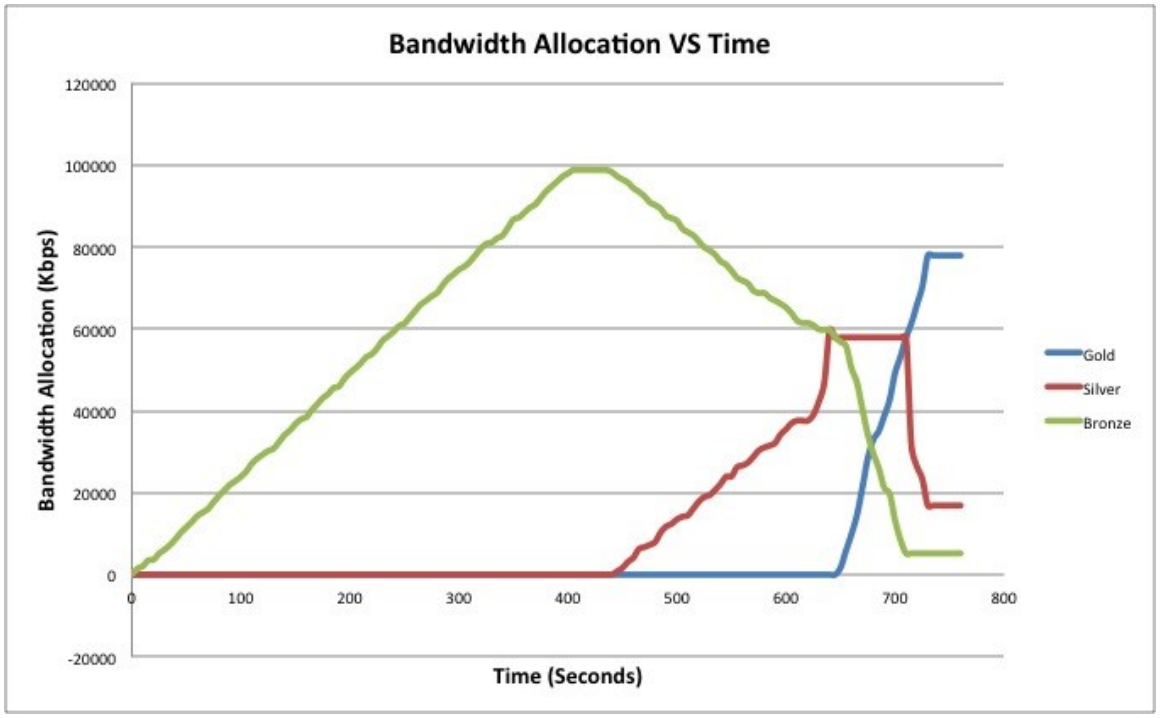

Figure 5.10 - Bandwidth Allocation for QoS levels over time

- Figure 5.11 shows the **end-to-end session creation delay per user** over time. This is the exact same experiment as the last one. During the experiment, every 5 seconds, session creation delays per users are calculated and average values are taken. The graph shows that, until 400 seconds, the average session creation delay was around 200 milliseconds because there were only session creations for the Bronze users. Between 400 and 440 seconds, the delay became zero milliseconds because there were only session rejections for Bronze users and no session initiations for Gold or Silver users. Between 440 and 650 seconds, the average delay became around 1800 milliseconds because there were forced session terminations for the Bronze sessions before the session admissions of the Silver users. Finally, after 650 seconds, the average delay rose to an average of 4000 milliseconds because more Bronze or Silver sessions need to be terminated (4

Bronze sessions and 2 Silver sessions) to admit one Gold session. Generally, these session delays are acceptable given that the users will benefit from privileged QoS during their SN usage. Furthermore, the users will not realize these delays while dealing with the SN.

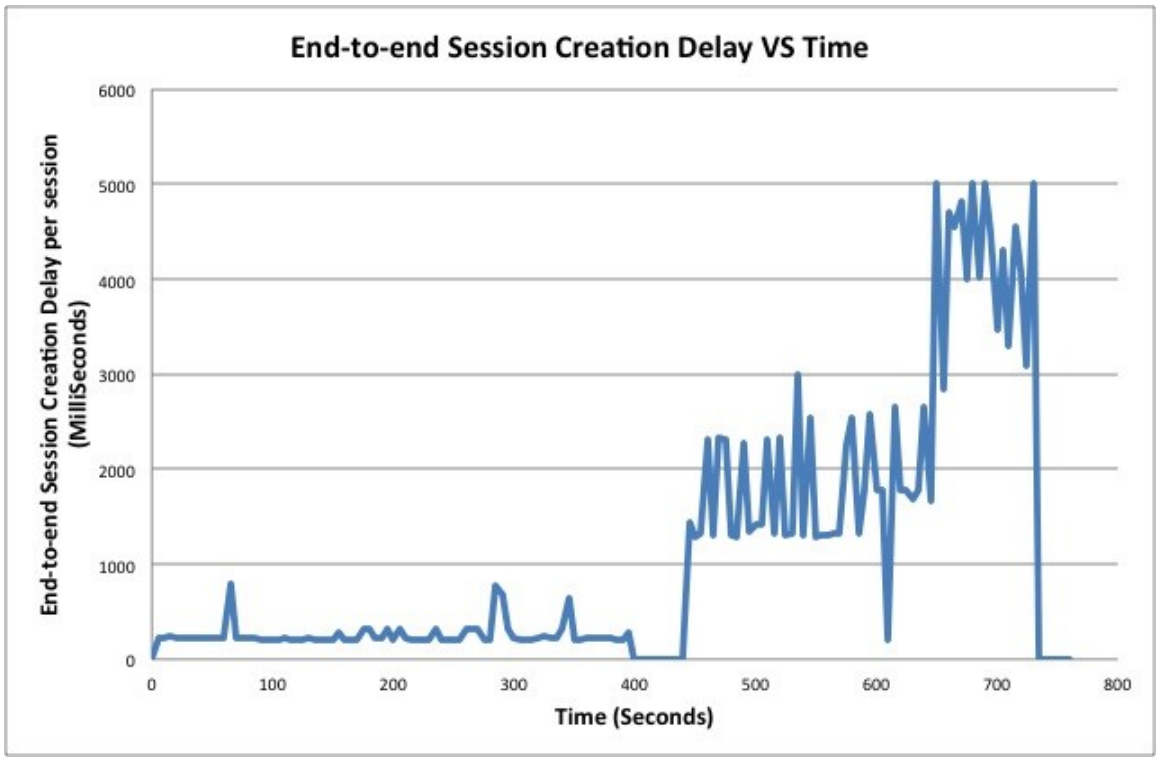

Figure 5.11 - End-to-end Session Creation delay per user over time

- Figure 5.12 shows the **filtering delay** of the content-filtering system against the number of posts. The graph shows that, generally, as the number of posts increases, the filtering delay increases. The average filtering delay for 100 posts is 14.443 seconds. This delay is acceptable considering that the users will not be bothered with undesired data during their SN usage.

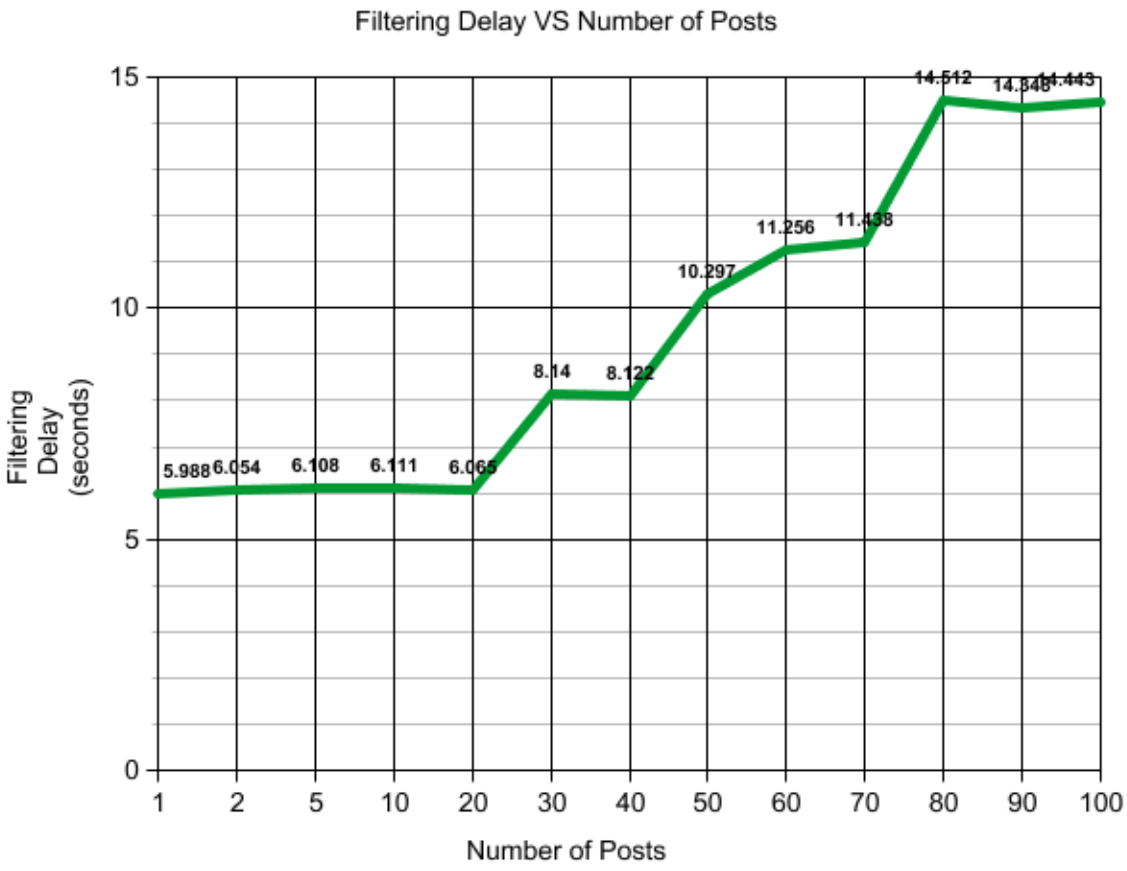

Figure 5.12 - Filtering Delay vs. Number of Posts

- Figure 5.13 shows the **filtering accuracy** against the number of attempts. The graph shows that the filtering accuracy increased from 80% on the first attempt to 87.5% in the 10th attempt. That means that the system's performance enhances with usage. In order to start using the KEA tool in the first place, a vocabulary for a certain topic and some training documents (some documents with their corresponding keywords) must be provided. This helps the tool to extract keywords in the future from the posts. However, the tool uses each attempt for keyword extraction as an ad-on training document. Thus, helping the accuracy of the tool to get better with usage [69].

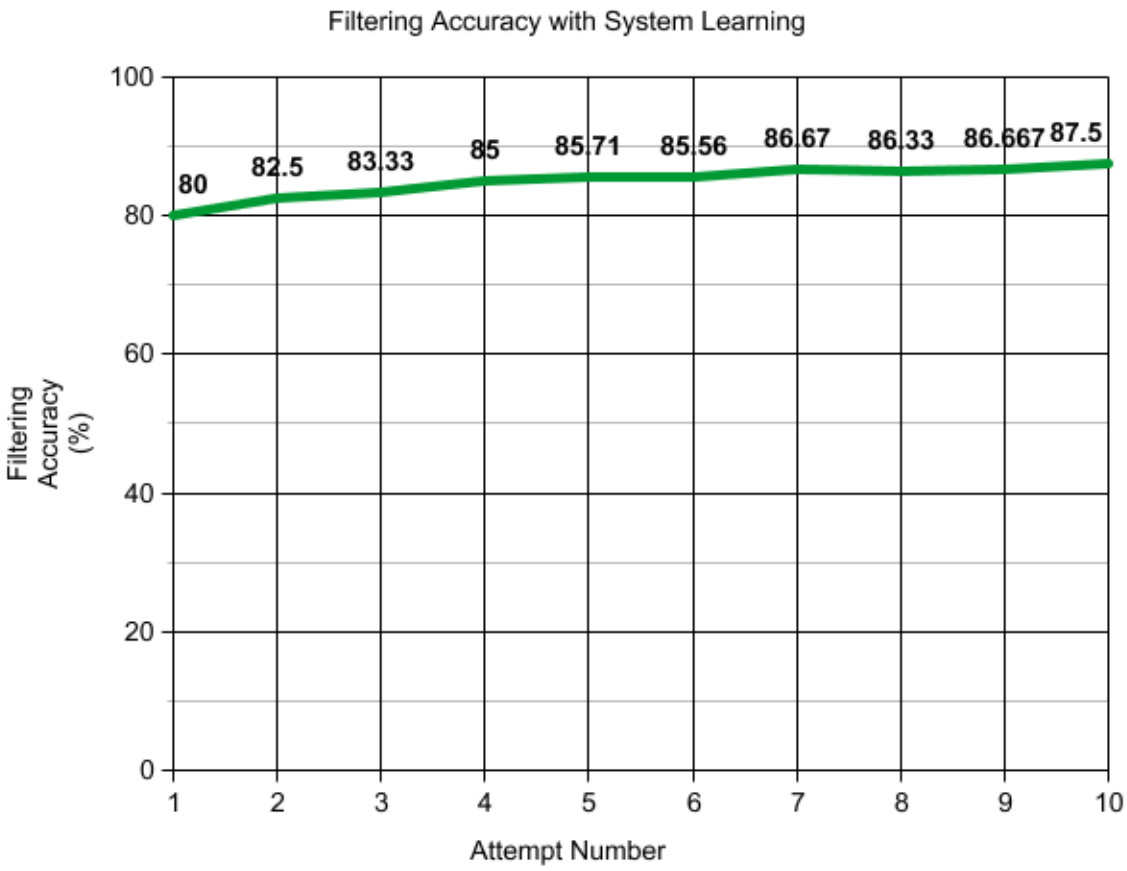

Figure 5.13 - Filtering Accuracy vs. Number of Attempts

5.4 Chapter Summary

This Chapter presented the implementation and evaluation of the aforementioned proposed architecture. The evaluation showed that the proposed architecture is feasible. The system favours high priority users over low priority users. Gold QoS level users allocated most of the available bandwidth followed by Silver and Bronze ones, respectively. Furthermore, the evaluation showed that the session creation delays vary between 200 milliseconds and 4000 milliseconds. These added delays are acceptable considering that the users will benefit from privileged QoS during their sessions with the SN. On average, there is 14.443 seconds of added delay to filter 100 posts. This added

delay, as well, is acceptable considering that the users will not be bothered with unwanted requests and undesired posts. Furthermore, these added delays are tolerable given the behaviour of the users with respect to the SN. The users will not realize these delays while dealing with the SN. Finally, the filtering accuracy increased from 80% (on the first attempt) to 87.5% (in the 10^{th} attempt), which means that the filtering system learns over time and produce better filtering results. The next Chapter will summarize our thesis contributions and discuss our potential future work.

Chapter 6: Conclusion and Future Work

In this chapter we summarize the contributions discussed in this thesis and overview the potential future work.

6.1 Summary of Contributions

As the number of SN users grows, there is higher demand for users' QoE. Some users would prefer to filter some posts, e.g. unwanted friendship requests and certain categories of posts. Other users may prefer to subscribe to a higher QoS level with their SN provider, e.g. to have higher priority on posting/retrieving.

We started, in this thesis, presenting the background information on the fields of Social Networks, RESTful web services, Quality of Experience, Quality of Service, Information Filtering, and 3GPP 4G Evolved Packet Core systems. The main focus of this chapter was to introduce the concepts of SNs, the QoE as the user satisfaction over the services offered, QoS as the means to deliver the services to the users, IF for hiding undesired data from the users, and the 3GPP 4G EPC systems as the emerging architecture that provides its users with end-to-end differentiated QoS.

We identified the requirements that a QoE-Enabled SN should meet. These requirements are supporting user and content-based filtering, user-based differentiated QoS, and reusing existing solutions, if possible. None of the related work met all our requirements for QoE-Enabled SNs. However, we concluded that Google's OpenSocial Framework can be used as a reference SN in our architecture and we can reuse its resources, and we can also use the 3GPP 4G EPC systems as a differentiated QoS provider.

We proposed a novel architecture that can enhance the users' QoE in SNs. the architecture provides the users with two features, the user-based differentiated QoS in the network layer and the filtering criteria that the users can provide. The filtering approach is user and content-based. The architecture relies on the 3GPP 4G EPC networks, which provides guaranteed and differentiated QoS. The components of the SN-Server, especially the Rule Handler, Filtering Engine and the QoS Application Function are all novel. The same applies to the interfaces and their modeling using RESTful Web services technology. A full description of the functional entities of the architecture, interfaces, and REST resources used, procedures and an illustrative scenario were provided.

Finally, a proof of concept prototype has been implemented to demonstrate the work and conduct a partial evaluation. After the performance results, the work was proven to be feasible. The system favours higher QoS level users over lower QoS level ones, thus allocating more bandwidth for their sessions. The session creation delays and filtering delays introduced due to session initiation in the EPC layer (sometimes also due to session terminations of lower QoS level sessions) and the filtering system, respectively, are considered acceptable. The filtering component of our system learns with time and produces more accurate filtering results as the number of attempts to use the system increases.

6.2 Future Work

There are different directions that can be considered. The first is the SN direction. In this thesis, our SN supports only text-based posts. This can be extended to support voice and video posts and even conferencing within the SN domain. Also, the SN implemented in

this thesis supported only user profiles. This can be extended to the participation in SN groups and allowing notification systems to be implemented for the users.

On the other hand, since we are using the 3GPP 4G EPC networks as the differentiated QoS provider, the 3GPP 4G EPC capabilities can be further used. The database of the system, that contains all the information of the users, can be implemented within the HSS of the EPC network instead of being locally used by the SN-Server and the QoS Enabler. An implementation of a Charging system can be done, as well, for the users. This can benefit the service providers in evaluating the charging control of the system and its reliability. Moreover, reviewing the related works on fairness between users will enhance the QoS system. This will make the system more reliable and encourage more users in every QoS level. Finally, session downgrades instead of session terminations can be considered for even better QoE of the users from different QoS levels.

The third and last research direction is about Information Filtering. A collaborative filtering technique can be used as an extension to our work. This will provide the users with possible posts within their interests. This can be an extension to simply filtering the unwanted category of posts from the user's news feed. Last but not least, a dynamic filtering criteria according to the users preferences can be suggested to the users and upon that, the filtering system should be extended and operate automatically.

Bibliography

[1] D. Boyd and N. Ellison, “Social Network Sites: Definition, History and Scholarship”. Journal of Computer-Mediated Communication 13 (2008) pp.: 210-230

[2] “Definition of Quality of Experience (QoE),” Reference: TD 109rev2 (PLEN/12), ITU-International Telecommunication Union

[3] J. Gozdecki, *et al*, “Quality of Service Terminology in IP Networks”. IEEE Communications Magazine, March 2003, pp.153-159

[4] N. Belkin and W. Croft “Information filtering and information retrieval: two sides of the same coin?” Communications of the ACM, NY, USA December 1992, pp. 29-38.

[5] M. Corici, *et al* “3GPP Evolved Packet Core-the Mass Wireless Broadband all-IP architecture”, in Telecommunications: The Infrastructure for the 21st Century (WTC), 2010, pp. 1–6.

[6] C. Fu et al, “RESTful web services for bridging presence service across technologies and domains: an early feasibility prototype,” Communications Magazine, IEEE, December 2010, pp. 92–100.

[7] Social Networking Statistics, Retrieved on November 2013 from “http://www.statisticbrain.com/social-networking-statistics/”.

[8] Facebook Statistics, Retrieved on November 2013 from “http://www.statisticbrain.com/facebook-statistics/”.

[9] Twitter Statistics, Retrieved on November 2013 from “http://www.statisticbrain.com/twitter-statistics/”.

[10] C. Kadushin, “Understanding Social Networks: Theories, Concepts and Findings”. Published in December 2011 by OXFORD University Press.

[11] C. Lampe *et al,* “A familiar face (book): profile elements as signals in an online social network”. ACM SIGCHI Conference on Human Factors in Computing Systems, NY, USA, 2007, pp.435-444.

[12] R. Fielding, “Architectural styles and the design of network-based software architectures”. PhD thesis. 2000, University of California, Irvine, department of Information and Computer Science.

[13] F. Belqasmi *et al*, "RESTful web services for service provisioning in next-generation networks: a survey", IEEE Communications Magazine, December 2011, pp.66-73.

[14] Web Application Description Language, Retrieved on February 2014 from: “http://wadl.java.net/”.

[15] Resource Oriented Architecture, Retrieved on February 2014 from: “http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#resource_oriented_model”.

[16] Extensible Markup Language (XML), Retrieved on February 2014 from: “http://www.xml.com/”.

[17] JavaScript Object Notation (JSON), Retrieved on February 2014 from: “http://www.json.org/”.

[18] L. Richardson *et al*, “Restful Web Services”, 1st edition, O’Reilly Media, May 2007.

[19] R. Fielding *et al.*, “Hypertext Transfer Protocol – HTTP/1.1”. IETF RFC 2616, June 1999.

[20] C. Pautasso *et al.*, “RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision”, In Proceedings of the 17th International World Wide Web Conference, ACM, Beijing, China, April 2008, pp. 805–814.

[21] K. Kilkki, “Quality of Experience in Communications Ecosystem”. Journal of Universal Computer Science, 2008, pp. 615–624.

[22] ETSI, “Human Factors (HF). Quality of Experience (QoE) requirements for real-time communication services”. 2009-2012.

[23] ITU-T Recommendation E.800, “Terms and Definitions Related to Quality of Service and Network Performance Including Dependability.” 1994.

[24] ETSI “Network Aspects (NA), General aspects of Quality of Service (QoS) and Network Performance (NP)”, Reference: RTR/NA-042102, October 1994.

[25] A. Meddeb, “Internet QoS: pieces of the puzzle,” IEEE Communications Magazine, 2010, pp. 86–94.

[26] M. Kaufmann, “Network Quality of Service, know it all”. 1st Edition, November 6th, 2008.

[27] B. Carpenter and K. Nichols, “Differentiated services in the Internet,” Proceedings of the IEEE, vol. 90, no. 9, 2002, pp. 1479–1494.

[28] G. Pasi, “Information Filtering”, technical report in Università degli Studi di Milano-Bicocca, Retrieved on February 2014 from: “http://www.ir.disco.unimib.it/wp-content/uploads/2010/09/CorsoSAI1213_Information_Filtering.pdf”

[29] U. Hanani *et al*, “Information filtering: Overview of issues, research and systems”, User Modeling and User-Adapted Interaction 11, 2001, pp. 203–259.

[30] C. Manning, “Introduction to Information Retrieval”, 1st Edition, Published by Cambridge University Press, July 2008.

[31] X. Su, and T. Khoshgoftaar, “A survey of collaborative filtering techniques,” Advances in Artificial Intelligence, no. 421425, 19 pages, January 2009.

[32] RFC6733, “Diameter Base Protocol”, October 2012.

[33] ETSI TS 123 401 V8.14.0, “LTE, General Packet Radio Service (GPRS), enhancements for Evolved Universal Terrestrial, Radio Access Network (E-UTRAN) access (3GPP TS 23.401 version 8.14.0 Release 8)”, 2011.

[34] H. Ekstrom, “QoS control in the 3GPP evolved packet system,” IEEE Communications Magazine, vol. 47, no. 2, 2009, pp. 76–83.

[35] M. Häsel and Otto Group, Hamburg, Germany. “Opensocial: an enabler for social applications on the web”. ACM New York, NY, USA. January 2011, pp.139-144

[36] J. Goldman, “Facebook Cookbook”, Published by O’Reilly Media, October 2008.

[37] S. Mohan and N. Agarwal, “A Convergent Framework for QoS Driven Social Media Content Delivery over Mobile Networks”, IEEE Wireless VITAE, 2011, pp.1-7.

[38] M.A. Rahman, et al, W. Gueaieb, “A Framework to bridge Social Network and body sensor network: An e-Health perspective”, IEEE International Conference on Multimedia and Expo. 2009, pp. 1724-1727.

[39] M. Abu-Lebdeh, “A 3GPP 4G Evolved Packet Core-Based System Architecture for QoS-Enabled Mobile Video Surveillance Applications”, Master thesis, May 2012, Concordia University, department of Electrical and Computer Engineering.

[40] A. Ratikan and M. Shikida, “Feature Selection Based on Audience's Behavior for Information Filtering in Online Social Networks”, IEEE Knowledge, Information and Creativity Support Systems (KICSS), 2012, pp.81-88.

[41] S. Nakamura and K. Tanaka, “Temporal filtering system to reduce the risk of spoiling a user's enjoyment”. ACM New York, NY, USA 2007, pp. 345-348.

[42] S. Loeb and E. Panagos, “Information filtering and personalization: Context, serendipity and group profile effects”, IEEE Consumer Communications and Networking Conference (CCNC), 2011, pp.392-398.

[43] E. Al-Shaer and H. Hamed, “Firewall Policy Advisor for anomaly discovery and rule editing”. IEEE Integrated Network Management, 2003, pp. 17-30.

[44] Facebook Login. Retrieved on February 2014 from:
“https://developers.facebook.com/docs/facebook-login”.

[45] W. SHU, “Facebook Platform”. McGill University retrieved on March 2014 from: “http://www.cs.mcgill.ca/~wshu/taing.html”.

[46] Keyphrase Extraction Algorithm (KEA), retrieved January 2014 from "http://www.nzdl.org/Kea/index.html".

[47] Zing, retrieved January 2014 from "https://code.google.com/p/zing/".

[48] VMware Workstation, Retrieved on January 2014 from: "http://www.vmware.com/products/workstation/overview.html".

[49] Fraunhofer Fokus OpenEPC retrieved January 2014 from: "http://www.openepc.net/index.html"

[50] The RESTLET Framework retrieved January 2014 from: "http://restlet.org"

[51] JavaDiameterPeer Library retrieved January 2014 from: "http://www.openimscore.org/project/jdp".

[52] Mozilla Firefox retrieved on January 2014 from: "http://www.mozilla.org/firefox/".

[53] Eclipse IDE, Retrieved on January 2014 from: "http://www.eclipse.org/".

[54] Top Websites by Traffic, Retrieved on February 2014 from: "http://www.statisticbrain.com/top-us-websites-by-traffic/".

[55] FAO Agricultural thesaurus Vocabulary Agrovoc, retrieved on February 2014 from: "http://aims.fao.org/standards/agrovoc/about".

[56] T. Reynaert, "PESAP: a Privacy Enhanced Social Application Platform", International Workshop on Security and Privacy in Social Networks (SPSN), Amsterdam, September 2012.

[57] World Cup 2010 Twitter outage retrieved November 2013 from: "http://www.nydailynews.com/news/money/world-cup-twitter-outages-fail-whales-article-1.180774"

[58] The Oscars 2014 Twitter's outage retrieved March 2014 from: "http://www.news.com.au/technology/online/ellen-degeneres-selfie-at-oscars-2014-breaks-twitter/story-fnjwnhzf-1226843798572"

[59] Reddit's Obama AMA outage retrieved March 2014 from: "http://thenextweb.com/socialmedia/2012/08/31/reddit-obama-ama-record-traffic-stats/"

[60] International Telecommunication Union retrieved February 2014 from: "http://www.itu.int/en/Pages/default.aspx"

[61] European Telecommunications Standard Institute retrieved February 2014 from: "http://www.etsi.org"

[62] S. Floyd, and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", August 1993, p. 397-413.

[63] Internet Engineering Task Force retrieved March 2014 from: "http://www.ietf.org/"

[64] C. Perkins and P. Calhoun, "Authentication, Authorization and Accounting", IETF RFC3957, March 2005.

[65] Information Sciences Institute, University of Southern California, "Transmission Control Protocol", IETF RFC793, September 1981.

[66] R. Stewart *et al*, "Stream Control Transmission Protocol", IETF RFC2960, October 2000.

[67] J. Postel, "User Datagram Protocol", IETF RFC768, August 1980.

[68] C. Rigney *et al*, "Remote Authentication Dial In User Service", IETF RFC2865, June 2000.

[69] O. Medelyan, "Semantically Enhanced Automatic Keyphrase Indexing." (Poster) In: Proc. of the Women in Machine Learning (WiML) Workshop co-located with the Grace Hopper Celebration of Women in Computing. San Diego, USA, 2006.