

VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING



**SOCIAL NETWORK SERVICES ON DECENTRALIZED SOCIAL
NETWORKS: TEXT-BASED SERVICES**

By
Le Quoc Thanh
ITIU09028

Supervisor
Vo Duy Khoi, MSc.

A thesis submitted to the School of Computer Science and Engineering in
partial fulfillment of the requirements for the degree of
Bachelor of Computer Science and Engineering

Ho Chi Minh City, Vietnam
2014

SOCIAL NETWORK SERVICES ON DECENTRALIZED SOCIAL NETWORKS: TEXT-BASED SERVICES

APPROVED BY SUPERVISOR

Vo Duy Khoi, MSc.

APPROVED BY COMMITTEE

Nguyen Duc Cuong, Dr.

Vo Duy Khoi, MSc.

Dao Tran Hoang Chau, MSc.

THESIS COMMITTEE
(Whichever applies)

ACKNOWLEDGEMENTS

Foremost, I would like to express my deepest gratitude to my supervisors MSc. Vo Duy Khoi and Dr. Tran Manh Ha who always steered me through every phase of my research work. Without their supervision and anticipation it would have been difficult for me to finalize my research work. Their supervision approach put me in a position where I could understand things much systematically and his critical yet fruitful remarks allowed me to dig deeper into the subject matter. Finally, I am so happy about their sharing experience not only about academic knowledge but also life knowledge.

Last but not least, I am greatly indebted to my dear family, who have encouraged and supported me through all my endeavors. Their love kept me steadily going in my pursuits.

February 10th, 2014

TABLE OF CONTENTS

LIST OF FIGURES	V
LIST OF TABLES	VI
ABSTRACT	VII
CHAPTER I – INTRODUCTION	1
1.1 - Motivation context.....	3
1.2 - Problems statement.....	4
1.3 – Scope.....	6
CHAPTER II – LITERATURE REVIEW	7
2.1 – P2P network	7
2.2 – Related work	12
2.2.1 – Likir.....	13
2.2.2 – Tribler	14
2.2.3 – PeerSoN	16
2.3 – Gnutella Protocol	17
2.3.1 - Protocol Definition	18
2.3.2 - Descriptor Routing	18
2.3.3 - Descriptor Header.....	19
2.3.4 – Detail of Descriptors	19
CHAPTER III – METHODOLOGY	23
3.1 – System architecture proposal	24
3.1.1 - Functionalities of peers.....	24
3.1.2 - Components inside peers	27
3.2 – Progresses in posting message service.....	32
3.2.1 – Update user’s statuses progress	32
3.2.2 – Update friends group’s statuses progress.....	34
3.2.3 – Update particular friend’s profile progress	35
3.2.4 – Acknowledgment messages progress	36
3.3 – Expanding Gnutella protocol	37
CHAPTER IV – RESULTS AND EVALUATIONS	47
4.1 – Results.....	47
4.1.1 – Home page	47
4.1.2 – Super peer and Friends connections	48
4.1.3 - Update user’s statuses.....	49
4.1.4- Update friends group’s statuses.....	53
4.1.5 - Update particular friend’s profile	53
4.1.6 – Optional functions.....	54
4.1.7 – Database structure	56
4.2 – Evaluations.....	58
CHAPTER V – CONCLUSION AND FUTURE WORKS	60
LIST OF REFERENCES	61

LIST OF FIGURES

Figure 1- Popular social network activities on Facebook and Twitter in 2011 [2].....	2
Figure 2- Peer-to-peer hierarchy tree [28]	7
Figure 3- Structured (Distributed Hash Table) P2P [28]	8
Figure 4- Centralized peer-to-peer architecture [29]	9
Figure 5- Decentralized peer-to-peer architecture [29]	10
Figure 6- Hybrid architecture [28]	11
Figure 7- Likir architecture [23]	13
Figure 8- Tribler architecture [21]	14
Figure 9- Ping/Pong/ Query/QueryHit/Push Routing [15]	18
Figure 10 – Super peer P2P architecture [1]	24
Figure 11- Peer components and communication [1]	27
Figure 12 - Update user's statuses progress.....	32
Figure 13- Update friends group's statuses progress.....	34
Figure 14 - Update particular friend's profile progress	35
Figure 15 - Acknowledgment progress.....	36
Figure 16 - Posting message home page.....	47
Figure 17 - Super peer and Friends connections.....	48
Figure 18 - Writing a public status	49
Figure 19- Showing public status after posting	49
Figure 20- Friends receive and show status on Friends' status field	50
Figure 21- Detail of a status including like and comment	50
Figure 22- Writing a private status	51
Figure 23- Showing private status after posting	51
Figure 24- After click Profile button	52
Figure 25- After click News feed button	53
Figure 26 - After click Friend's name.....	53
Figure 27- Notification	54
Figure 28- After click Notification button	54

Figure 29- After click Friend Group button.....	55
Figure 30- List of liked users	55

LIST OF TABLES

Table 1- Architectural comparison	12
Table 2- Gnutella protocol messages	18
Table 3- Connectivity	58
Table 4 - Reliability	59

ABSTRACT

In the recent years, online social network has attracted a large number of people around the world and been considered as an effective way for people to communicate with friends, family and others. However, existing online social network has several drawbacks including scalability, privacy, dependence on other providers, needs of being online for every transaction, and the lack of locality. Hence, this thesis presents a proposal and implementation of posting message service on social network with a super peer peer-to-peer architecture in distributed environment. The purpose of this service provides users the capability of managing the dissemination of user social data, searching user data on the data silos of the network. Users use peers to participate the social network and services. Peers with sufficient storage, bandwidth and processing power become super peers that support peers for complex operations such as processing complex queries, transmitting large amount of data or group communication. In addition, I have extended the Gnutella protocol to implement the posting message service on the social network. Last but not least, this proposal is contributing apart from the content of the publishing research paper named “Peer-to-Peer Based Social Network [1]” on NICS conference.

CHAPTER I – INTRODUCTION

Online social network forms an important part of users' online activities. In fact, some popular sites as Facebook [8], MySpace [9], Twitter [10], etc. have attracted millions of users, many of whom have integrated these sites into their daily practices. In addition, online social network has provided a medium for people to communicate and share social data, which includes user profile, pictures and video files. In the online social network, members are interconnected with each other through some relationship like friendship or common preferences to share their thoughts, activities, events, photos, videos, links to web pages, and interests. When users' social data shared to the network, the shared social data propagate to each and every members of the network whether it is relevant to them or not. From the viewpoint of a sender, they are sharing only one social data at a moment. However, the receiver can be received more than one social data at a moment.

An online social network essentially consists of a variety of services, which focuses on building and reflecting of real social. There are many services in these popular sites such as media sharing, posting message, file sharing, instant message, etc. In fact, users participate in social network sites using these services to interact with their friends. These services' activities could be posting message about personal interests, composing contents, chatting with friends, like or commenting on friends' posting, sharing interested links, playing game, or tagging people, etc. In fact, the detail of time usage each activities of 623 users in the most two popular social networks [26] as Facebook and Twitter illustrated in Figure 1.

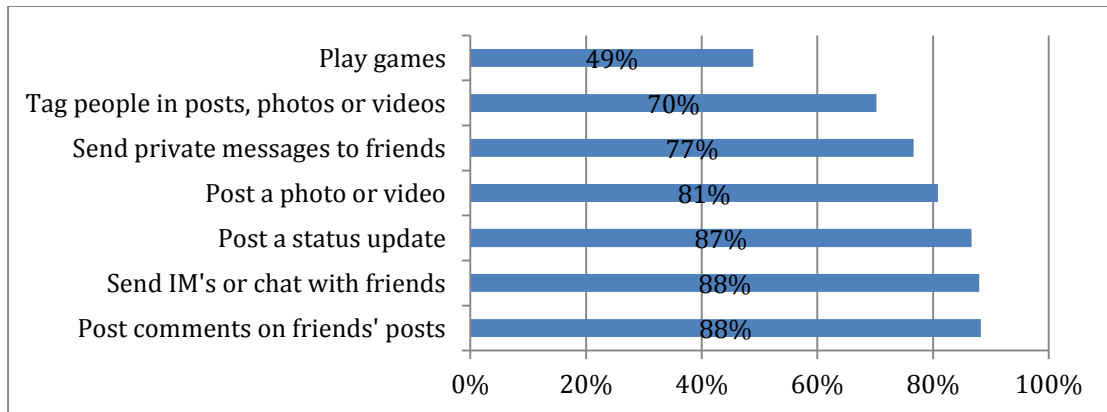


Figure 1- Popular social network activities on Facebook and Twitter in 2011 [2]

Although these services on social network are varied, but it can be categorized into three folds:

- Text exchanging: focuses on short updates by text that are pushed out to anyone subscribed to receive the updates. Users interact with each other by text as discussion posting message, chat, blog, quizzes, article, etc. with some most popular are Facebook and LinkedIn, Twitter.
- Resource sharing: allows user to upload and share various media such as pictures and video, lecture notes, slides. Most services have additional social features such as profiles, commenting, etc. These materials must have clearly metadata for managing and controlling resources that exchanging by the system. There are two most popular sites about resource sharing are YouTube and Flickr.
- Browsing event: contains all actions as searching, and making a friend, etc. This is a valuable data input for some user tracking algorithms.

1.1 - Motivation context

Most of the available online social network are based on client-server architecture in which users' social data are kept centralized. This architecture supports high accessibility since users can access the service from any web-browser wherever and whenever they desire. However, the fast development of these social network sites not only brings benefits to users but also poses several problems are shown as follows:

- Firstly, privacy issues that users cannot manage the dissemination of their personal data and search data on the data repository of social networks caused by dependence on central administration of provider.
- Secondly, storage issues that single point of failure caused by central data storage result in the requirement of larger servers and bandwidth to accommodate the growing number of users.
- Finally, users cannot consolidate their personal data from various social networks. In fact, information on one site is not usable in the others.

These limitations of existing social network services, motivated to consider another trend of extending current these services toward decentralized online social network while retaining the functionalities offered by centralized infrastructure. In fact, a decentralized online social network [24, 25] is a distributed system for social networking with independence on any dedicated central infrastructure, where users have more control over their own social data.

1.2 - Problems statement

A decentralized online social network [24, 25] is an online social network implemented on a distributed information management platform, such as a network of trusted servers or a peer-to-peer systems. By decentralized online social network, the concept of a service provider is changed, as there is no single provider but a set of peers that take on a share of the tasks needed to run the system. Decentralized online social networks can solve problems of centralized infrastructure by three respects:

- *Privacy*: users in decentralized social networks decide who to show their social data to and what restriction there is on the data.
- *Ownership*: as the social data is stored on a trusted servers or on the local device that users have complete ownership of the data. They would not lose their data suddenly as the proprietary service hosting their data decides to shut down without much notice.
- *Dissemination*: social data is disseminated according to users' preferences and friendship relations.

Moreover, decentralized online social network can be implemented with the using of peer-to-peer (P2P) network [3]. A P2P network is a distributed network in which nodes are connected with each other to participate in processing, memory, and bandwidth intensive tasks. These networks scale better than centralized server architectures without the need of costly centralized resources. In addition, P2P networks have been popular mostly as file sharing networks such as KaZaa [19],

BitTorrent [17], etc. and sometimes as collaborative sharing networks such as Skype [20], but have not been used as a medium for online social networking.

In slew of initiatives, originating both from academic research such as Diki [24], FOAF [24], PeerSoN [22], SafeBook [18] as well as open source and free privacy speech centric communities have in the last couple of years started looking at decentralized architectures which can provide online social networking services without the trappings of such services being run by centralized providers. The briefly contents of these academic research are shown as follows:

- Safebook: extends peer-to-peer approach to gain objective of protecting users' privacy, integrity, and availability.
- FOAF: the framework enables users to export their FOAF profiles, store them on dedicated trusted servers. Users query and manage these profiles through open Web-based protocols such as WebDAV or SPARQL/Update
- PeerSoN: the main properties of PeerSoN are encryption, decentralization, and directly data exchange, which aims at keeping the features of centralized online social network but overcoming two limitations: privacy issues and the requirement of Internet connectivity for all transactions. To address the privacy problem, it uses encryption and access control coupled with a peer-to-peer approach to replace the centralized authority of classical online social network.
- Diki: a social bookmarking service that allows users to encrypt and share their bookmarks with trusted friends via the Extensible Messaging and Presence Protocol (XMPP). Three design principles for user's privacy:

enable data exchange only between trusted friends, not storing any data centrally, and any stored data is encrypted, which uses the PGP private key encryption scheme.

1.3 – Scope

This paper presents a proposal architecture and implementation of posting message service on decentralized online social network based on a peer-to-peer architecture. The social network employs a super peer peer-to-peer architecture that contains peers and super peers [7]. In fact, this proposal expresses that how to these problems of centralized infrastructure can be solved by adopting a decentralized approach to online social networking. Moreover, this approach can provide the same or even higher level of user interaction as with the functionalities offered by centralized online social network in many of the popular social network sites today. In addition, it also allows users to have more control over their own data. Lastly, the implementation of posting message service on desktop application environment will approve feasibility of the approach, in which this service supports users posting, like or comment on a text message. In addition, this service is extending Gnutella protocol, which most of the user are free riders in file sharing P2P systems, to inherits several good characteristics of peer-to-peer architecture including scalability in architecture, reliability in content distribution and autonomy in administration to implement on decentralized online social network.

CHAPTER II – LITERATURE REVIEW

In this chapter, I will mention about P2P network because posting message service are built based on super peer peer-to-peer network, which is one kind of P2P network. In addition, I will review some similar systems that have applied P2P technology to building decentralized social networks. Lastly, I will describe the pure Gnutella protocol in file sharing system, which is extended to implement the posting message service.

2.1 – P2P network

A P2P network contains a large number of workstations that share computing resources including storage, bandwidth and processor power. There are two types of P2P networks: structured and unstructured networks that are shown as in Figure 2.

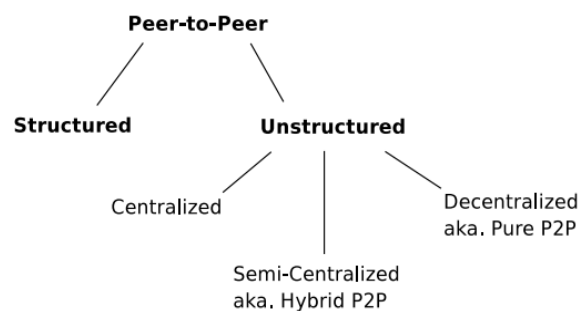


Figure 2- Peer-to-peer hierarchy tree [28]

Each workstation or peer in the network acts as a client and server to consume and provide services respectively. Peers can dynamically join and leave the network without causing the instability of other peers. The network also possesses advantages in reducing collaboration cost through ad-hoc communication process and providing high fault-tolerance and scalability.

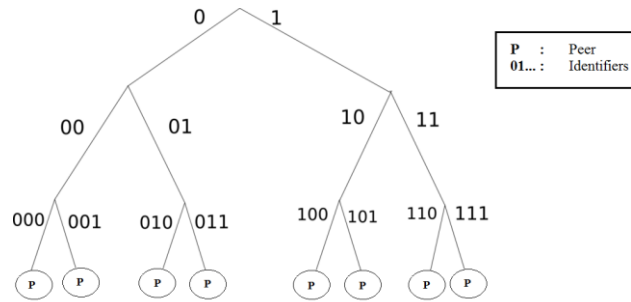


Figure 3- Structured (Distributed Hash Table) P2P [28]

Structured P2P network uses Distributed Hash Table (DHT) to generate uniquely consistent identifiers for peers and resources such that the peers hold the resource indexes if their identifiers are in the same identifier space, which are shown as in Figure 3. In more detail, this network uses DHT algorithm to locate contents of peers in the network result in each of contents is provided a unique ID. Hence, this network supplies contents for peers through the queries mapping, so it will ensure to provide correctly data to required peers. However, disadvantage of this network is hard to maintain structure P2P network in a long time because peers are joining and leaving at a high rate. Some typical structured P2P networks referred as CAN [11], Chord [12], and Kademlia [13].

Unstructured P2P network is loosely controlled and organized in random model. This network uses mainly flooding-based search (breath-first and depth-first search), random query or expanding Time-to-live mechanisms for searching resources of networks. However, a disadvantage of this network is that it has to send many queries to the other peers when a peer wants to query a resource, because it is not bonding between topology and data location. Some typical unstructured P2P networks referred as Gnutella [15], Freenet [16], and BitTorrent [17]. Unstructured P2P network

is divided into three kinds of architectures: centralized, decentralized, semi-centralized architecture.

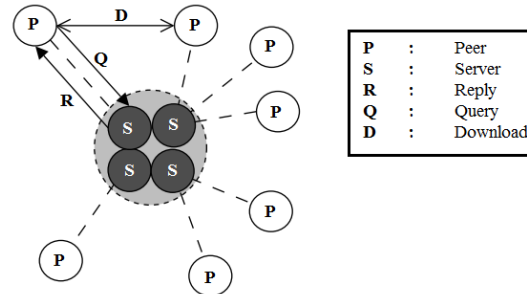


Figure 4- Centralized peer-to-peer architecture [29]

Centralized peer-to-peer architecture looks like a client-server model as in Figure 4, which is centralized architecture of a file sharing system. Each of client stores shared resources with the rest of the network. All clients always connect directly to a central repository server that managing list of user connection IP address and list of data information that each user is holding and sharing. When a client searches resources, it will send a query to the server. After that, the server will search for matches in its index, and return a list of users' information that holding the mapping files. Next, the requested peer will connect directly to the peer, which the server just specified. All further communications are directly between these two peers while the server was only vital for searching resources. In fact, advantages of centralized peer-to-peer architecture are simple to implement, and resources are located quickly and efficiently. However, it still exist some disadvantages that they are vulnerable to censorship, legal action, surveillance, malicious attack, and technical failure, because the shared content are controlled just by the single institution, company, or user that face to single point failure of central server occurring.

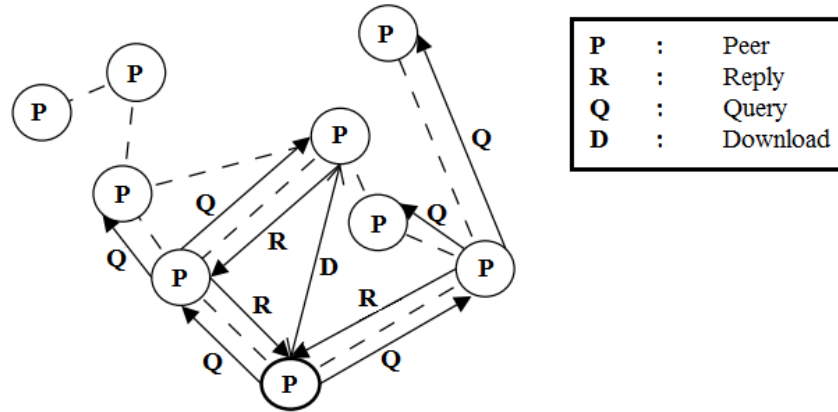


Figure 5- Decentralized peer-to-peer architecture [29]

Decentralized peer-to-peer architecture is also called the pure peer-to-peer architecture because there is no central coordination of the activities in the network as in Figure 5, which is decentralized infrastructure of a file sharing system. Also, users are referred to as servants, which are both as a client and server that directly connected to each other. In addition, when a peer is storing the requested resources, it will reply to the original requested peer via the same routed path initially. Therefore, searching resources in pure peer-to-peer networks will forward queries from node to node using a flooded request algorithm as in Gnutella 0.4 [15] which broadcasts effectively the query to limited scope. Whereas, some more intelligent routing method as in Freenet [16] that the query is routed towards the host most likely having the requested file. In these searching, the Time-to-Live (TTL) field in the queries header is used to limit the transmitting queries that avoiding traffic network. However, scalability issues in the original purely decentralized systems arose from the fact that the use of the TTL effectively segmented the network into “sub-networks”, imposing on each user a virtual “horizon” beyond which their messages could not reach [6].

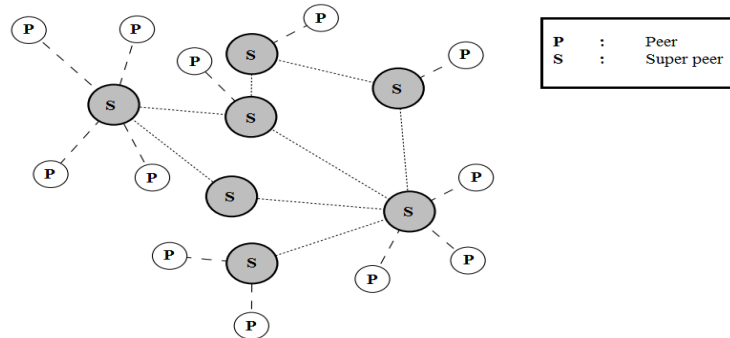


Figure 6- Hybrid architecture [28]

Semi-centralized architecture is also called the hybrid peer-to-peer architecture as in Figure 6. This architecture is a combination of the centralized and decentralized peer-to-peer architectures. The hybrid networks consist of several super peers and peers. Each of peers will connect to at least one super peer while these super peers will connect to each other to form a group of super peers. Therefore, if one super peer disconnects from the network, the peer can still reach the peer-to-peer network via the other super peers. In addition, peers are automatically elected to become super peer if they have sufficient storage, bandwidth and processing power. In semi-centralized architecture, the super peers will hold resources information of connecting peers, while peers will advertise their sharing resources to super peers. In addition, the key difference between hybrid and centralized architecture is that each of super peers only manage connecting peer's information to it without the peers are connecting with other super peers. While centralized architecture will manage all of connecting peers information.

As these architectures above, each of peer-to-peer architectures has different properties. The scalability, resiliency, search efficiency and ability to control depend

on the architecture of the network. For example, the pure peer-to-peer architecture is facing scalability issue to a large number of nodes. However, it is the most resilient against node failures. Also, a failure of the centralized server eliminates the whole centralized peer-to-peer network. While the hybrid architecture is a combination between the pure and centralized peer-to-peer architectures. In fact, it scales quite well and has good resiliency as long as. The hybrid architecture is also situated somewhere between the other two architectures when considering the search coverage in the network. When a peer sends a query out, the request is forwarded to other super peers that then forwarded the request onwards. The TTL field of the request message limited the search coverage, but the coverage is larger in the hybrid architecture than the pure P2P architecture.

Comparisons between peer-to-peer architectures are presented in the Table 1.

Architecture	Pure	Hybrid	Centralized
Scalability	Low	Very high	Medium
Signaling overhead in super peer	–	High	Very high
Signaling overhead in ordinary peer	High	Low	Low
Resiliency	Very high	Medium	Low
Operator control	Low	High	Very high
Search coverage	Medium	High	Very high

Table 1- Architectural comparison

2.2 – Related work

A study of Boyd et al. [14] has referred to social networks as several networks that support online social activities including making friends online, publishing user data online through posting, chatting, media sharing, etc. Whereas, decentralized online

social networks provide various social computing services on distributed environment. These services improve the limitations of the centralized servers by using the decentralized servers. Some studies have applied P2P technology to building decentralized social networks as Likir [23], Tribler [21], PeerSoN [22], etc. These studies are shown as follows:

2.2.1 – Likir

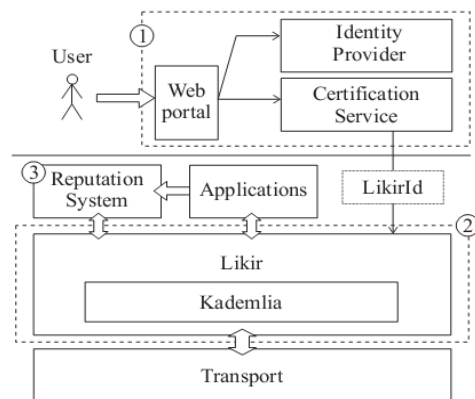


Figure 7- Likir architecture [23]

Likir [23] is a Kademlia-based Distributed Hash Table, a p2p distributed hash table and getting famous among other distributed hash tables due to its quick search, aimed to protect the overlay against attacks common to these systems, by embedding a strong identity notion at overlay level.

The main motivation of Likir is to avoid a centralized storage of personal information, for both the reason to avoid a single point of failure and aggregation of user data. In addition, the applications built on top of this substrate consequently are relieved from identity management, as it is already provided by the underlying Distributed Hash Table. Moreover, the architecture of Likir is structured in as in Figure 7, which is designed to achieve full confidentiality of data as well as

anonymity of the users. Also, it additionally provides access control in a granularity to applications: authorized disclosure ensures that malicious applications installed at some individual's device is unable to access data disclosed to other applications.

Likir builds its core properties on introducing cryptography in a plain Distributed Hash Table-based approach. All nodes are furnished with an identifier in the form of an OpenID by a certification service. Also, subsequent communication events consequently are encrypted and authenticated by both communicating parties. In addition, a supplemental access control scheme is integrated that requires all service providing nodes to check grants that are appended to each request before returning any data. Lastly, a prototype of the Likir middleware has been published and is available for download, including a simple chat application for the purpose of demonstrations. Besides decentralized counterparts of general purpose online social networking applications as described above, there are also initiatives to realize decentralized counterparts of niche applications such as video sharing, social bookmarks and libraries, and micro-blogging to name a few.

2.2.2 – Tribler

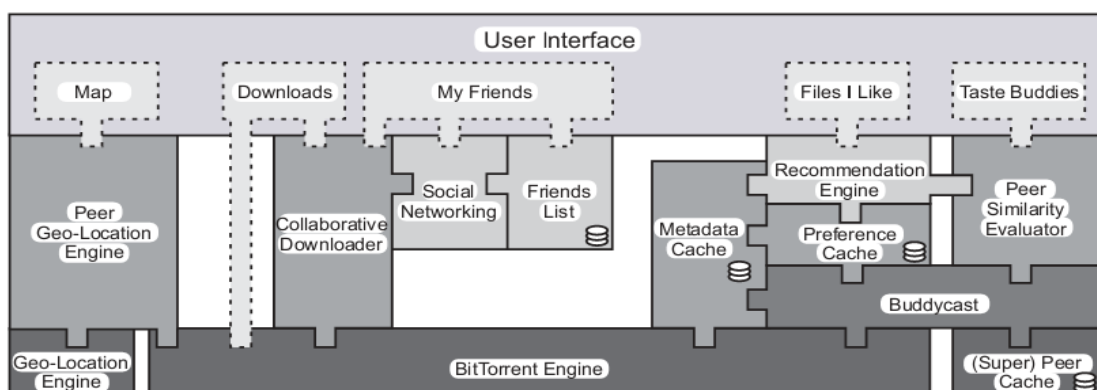


Figure 8- Tribler architecture [21]

Tribler[21] based on purpose Social-based P2P File Sharing, which is basically a P2P content sharing system that leverages the existing social relationships and taste similarities among its users for fast discovery and recommendation of digital contents. Moreover, Tribler has been extensively developed and provides a rich set of functionalities for its users, e.g., discovery and recommendation of friends with similar preferences, as well as providing fast and low overhead delivering video on demand such as television programs. While the system designed the architecture as in figure 8 and implemented as a social-based extension of the BitTorrent engine, is freely available and has been evaluated at small and medium scales. To facilitate the social group formation between users, Tribler uses a de-anonymized approach to manage the peer identities. During the registration phase, each participating user is given by email a secure, unique, permanent identifier (PermID). For the PermIDs, they are obtained via a public key generation scheme with challenge response mechanism to prevent spoofing. In addition, for existing contacts of a user, they can also be imported from other social networks such as MSN and GMail. Whereas, for bootstrapping the networks that is done via an epidemic protocol, any newly joined peer can contact one in a list of pre-known super peers, from which to obtain a list of other peers already available in the systems.

A peer in Tribler uses many types of caches to store locally any contextual information relevant to its interests and tasted. These so-called Mega caches (each less than 10MB) of a peer may store various information related to its friend lists, the altruism levels, the preferences of its friends, and meta-data of the files and contents posted in the network. For existing BitTorrent networks, the

number of files injected per day is sufficiently small such that the caches of file meta-data can be fully replicated among all peers. Thus, there is no need to perform network wide searching for interested content, but only the browsing of the local meta-data cache. Whereas, peers in Tribler are clustered into many groups, each of which contains those peers with similar preferences and interests, or taste buddies group. The interest commonalities between two peers are given by the similarity in their preferences of in the same or related content.

2.2.3 – PeerSoN

PeerSoN [22] keeps the features of online social networks but overcoming two limitations: privacy issues and the requirement of Internet connectivity for all transactions. In addition, PeerSoN uses encryption and access control coupled with a peer-to-peer approach to address the privacy problem and to replace the centralized authority of classical online social networks. These two measures prevent privacy violation attempts by users, providers, or advertisers. In extending the decentralized approach, PeerSoN also enables direct exchange of data between devices to allow for opportunistic, delay-tolerant networking. This includes making use of ubiquitous storage to enable local services.

The main properties of PeerSoN are encryption, decentralization, and direct data exchange. In a nutshell, encryption provides privacy for the users, and decentralization based on the use of a P2P infrastructure provides independence from online social networks providers. While decentralization makes it easier to integrate direct data exchange between user's devices into the system. Direct exchange allows

users to use the system without constant Internet connectivity, leveraging real life social networking and locality. Currently, PeerSoN implementation replicates the following features of online social networks. In the category of social links, peers can become friends and thus establish a social link between each other. Thus, digital personal spaces are provided in that users can maintain their own profile and a wall, a space for items posted by themselves or their friends. In addition, communications between users are directly peer-to-peer when both are online, and the implementation supports asynchronous messaging when this is not the case.

PeerSoN uses a Distributed Hash Table as a lookup system and then lets peers connect directly; all data is encrypted and keys for accessing an object are encrypted for the exclusive use of authorized users and stored in a separate file associated with a particular object, such as a user's profile. Lastly, the prototype implementation has been tested on PlanetLab and uses OpenDHT for the lookup service.

2.3 – Gnutella Protocol

Gnutella is a protocol for distributed search, designed originally by Nullsoft. Although the Gnutella protocol supports a traditional client-centralized server search paradigm, Gnutella's distinction is its peer-to-peer, decentralized model. In this model, every client is a server, and vice versa - called servents. In more detail, a peer wishes to participate in the network must join to the Gnutella network by connecting to any existing peers. Currently, mechanisms as known as "host caches" allow new

peers to participate in the network easily by connecting to a random provided peer's IP address.

2.3.1 - Protocol Definition

The Gnutella protocol defines the way in which servents communicate over the network. It consists of a set of descriptors used for communicating data between servents and a set of rules governing the inter-servent exchange of descriptors. Currently, the following descriptors are defined in Table 2.

Type	Description
Ping	A request for information about another node(s)
Pong	A reply carrying information about a node (e.g., number of shared files)
Query	A request for a resource (e.g., searching for a file)
QueryHit	A response identifying an available resource (e.g., a matching file)
Push	A mechanism that allows a firewalled node to share data

Table 2- Gnutella protocol messages [15]

2.3.2 - Descriptor Routing

The peer-to-peer nature of the Gnutella network requires servents to route network by ping, pong, query, queryhit, and push messages shown as in figure 9.

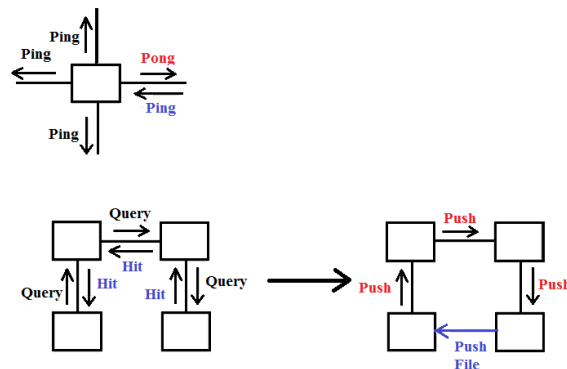


Figure 9- Ping/Pong/ Query/QueryHit/Push Routing [15]

2.3.3 - Descriptor Header

	<i>Message ID</i>	<i>Payload Descriptor</i>	<i>TTL</i>	<i>Hops</i>	<i>Payload Length</i>		
<i>Byte offset</i>	<i>0</i>	<i>15</i>	<i>16</i>	<i>17</i>	<i>18</i>	<i>19</i>	<i>22</i>

- Descriptor ID: a 16-byte string uniquely identifying the descriptor on the network.
- Payload Descriptor: identifying the message type as 0x00 = Ping, 0x01 = Pong, 0x40 = Push, 0x80 = Query, 0x81 = QueryHit.
- TTL: Time To Live field, indicating the maximum number of forwards until this message is removed from the network. Therefore it is decremented by every servent it passes until it reaches null.
- Hops: number of nodes passed; incremented by every servent passed.
- Payload Length: length of this messages' descriptor immediately following. This is a very important field since it marks also the beginning of the next message in the input stream of a servent.

2.3.4 – Detail of Descriptors

In original Gnutella protocol, there are five types of messages as in the table 2. A peer connects to multiple existing Gnutella peers to be able to reach more total Gnutella peers. Once a peer has connected to the network, it will communicate with its neighboring peers by sending and receiving Gnutella protocol messages as in table 2, and accepts incoming connections from the new peers that wish to join to the network.

a) Ping (0x00)

Ping descriptors have no associated payload and are of zero length. A Ping is simply represented by a Descriptor Header whose Payload_Descriptor field is 0x00 and whose Payload_Length field is 0x00000000. In addition, a servent uses Ping descriptors to actively probe the network for other servents.

b) Pong (0x01)

Pong descriptors are only sent in response to an incoming Ping descriptor. It is valid for more than one Pong descriptor to be sent in response to a single Ping descriptor.

	<i>Port</i>		<i>IP Address</i>			<i>Number of Files Shared</i>				<i>Number of Kilobytes Shared</i>			
<i>Byte offset</i>	<i>0</i>	<i>1 2</i>	<i>5</i>	<i>6</i>		<i>9</i>	<i>10</i>					<i>13</i>	

- Port: The port number on which the responding host can accept incoming connections.
- IP Address: The IP address of the responding host in little endian format.
- Number of Files Shared: The number of files that the servent with the given IP address and port is sharing on the network.
- Number of Kilobytes Shared: The number of kilobytes of data that the servent with the given IP address and port is sharing on the network.

c) Query (0x80)

	<i>Minimum Speed</i>		<i>Search criteria</i>	
<i>Byte offset</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>...</i>

- **Minimum Speed:** The minimum speed (in kB/second) of servents that should respond to this message. A servent receiving a Query descriptor with a Minimum Speed field of n kB/s should only respond with a QueryHit if it is able to communicate at a speed $\geq n$ kB/s.
- **Search Criteria:** A null (i.e. 0x00) terminated search string. The maximum length of this string is bounded by the Payload_Length field of the descriptor header.

d) QueryHit (0x81)

QueryHit descriptors are only sent in response to an incoming Query descriptor. A servent should only reply to a Query with a QueryHit if it contains data that strictly meets the Query Search Criteria. The Descriptor_Id field in the Descriptor Header of the QueryHit should contain the same value as that of the associated Query descriptor. This allows a servent to identify the QueryHit descriptors associated with Query descriptors it generated.

	<i>Number of Hits</i>		<i>Port</i>		<i>IP Address</i>		<i>Speed</i>		<i>Result Set</i>		<i>Server Identifier</i>	
<i>Byte offset</i>	0	1	2	3	6	7	10	11	...	n	n+16	

- **Number of Hits:** The number of query hits in the result set.
- **Port:** The port number on which the responding host can accept incoming connections.
- **IP Address:** The IP address of the responding host.
- **Speed:** The speed (in kB/second) of the responding host.

- **Servent Identifier:** A 16-byte string uniquely identifying the responding servent on the network. The Servent Identifier is instrumental in the operation of the Push Descriptor.
- **Result Set:** A set of responses to the corresponding Query. This set contains `Number_of_Hits` elements, each with the following structure:

	<i>File Index</i>	<i>File Size</i>	<i>File Name</i>
<i>Byte offset</i>	0 3	4 7	8 ...

With:

- **File Index:** A number, assigned by the responding host, which is used to uniquely identify the file matching the corresponding query.
- **File Size:** The size of the file whose index is `File_Index`
- **File Name:** The double-nul (i.e. 0x0000) terminated name of the file whose index is `File_Index`.

e) Push (0x40)

A servent may send a Push descriptor if it receives a QueryHit descriptor from a servent that doesn't support incoming connections. This might occur when the servent sending the QueryHit descriptor is behind a firewall. When a servent receives a Push descriptor, it may act upon the push request if and only if the `Servent_Identifier` field contains the value of its servent identifier.

	<i>Servent Identifier</i>	<i>File Index</i>	<i>IP Address</i>	<i>Port</i>
<i>Byte offset</i>	0 15	16 19	20 23	24 25

- **Servent Identifier:** is the same with Servent Identifier of QueryHit. This allows the recipient of a push request to determine whether or not it is the target of that request.
- **File Index:** The index uniquely identifying the file to be pushed from the target servent. The servent initiating the push request should set this field to the value of one of the File_Index fields from the Result Set in the corresponding QueryHit descriptor.
- **IP Address:** The IP address of the host to which the file with File_Index should be pushed.
- **Port:** The port to which the file with index File_Index should be pushed.

CHAPTER III – METHODOLOGY

Decentralized online social networks provide various social network services on distributed environment. These services improve the limitations of the centralized servers by using the decentralized servers. In addition, based on several studies have applied P2P technology to building decentralized social networks, I would like to propose posting message service on social network to approve feasibility of the approach P2P based social network [1] that employs a super peer P2P architecture to achieve remarkable features in distributed environment (see Figure 10). In this service, I just focused on how to users communicate with their friends, family and the other people by text message without authenticate service.

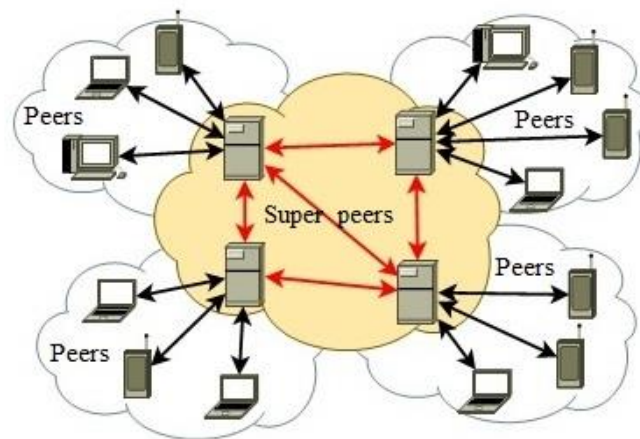


Figure 10 – Super peer P2P architecture [1]

3.1 – System architecture proposal

3.1.1 - Functionalities of peers

Figure 10 plots the architecture of the proposal P2P based social network that focuses on the posting message service of users on social network. This service helps users post statuses on their wall, which is the space where store users' content, and forward these statuses to the whole friends groups, a set of friends or one friend without losing data when a single point failure of central server occurring. There are two kinds of peers in this system: peers and super peers. This architecture coincides the architecture of the super peer P2P networks [7]. For peers, which can be desktop, laptop or mobile devices. Whereas, the super peers are only desktop or laptop because it needs not only strongly system resources and stable connectivity but also high uptime without being behind a firewall to execute many complex operations and stores large amount of data. Each component in peers and super peers has some similar and different functions when they overlay social network. In fact, the key difference between peers and super peers are that super peers support search function

including search friends and search users' statuses.

For peers as desktop, laptop or mobile devices, which are users using social network. In more detail, there are five main functions: making connection with super peers, update users' statuses to their friends through super peers, update friend group's statuses that are text messages to indicate users' behaviors as News Feed of Facebook [4], store users' statuses and profile, and update particular friend's statuses. For overlay social network, each peer needs to connect with at least one super peer to join to social network. This is very important because it helps to distribute user social data into separate super peers to avoid losing data when single point failure of central server occurring, which is one of problems in client-server model. In addition, peers will send user's statuses to the whole friends groups, a set of friends or one friend through super peers when they wrote behaviors on their wall. After that, store user's profile and statuses feature will be activated to store user social data into device's database that helps users managing their own data. Also, the main goal of update particular friend's profile feature will help users see and keep tracks all of friends' shared statuses. Last feature, update friend group's statuses will help users update information of friends groups that was shared on their wall.

In the other kind of peers, I mention about super peers, which are the most important component in this architecture. In super peers, it is quite similarity with peers that still including five main functions as making connection with super peers, update users' statuses through super peers, update friends groups' statuses, store user's statuses and profile, and update particular friend's statuses. However,

super peers consist of two new main functions as search and store data, which save social data of both super peers and some peers, which super peers are managing. These two functions only have in super peers because it has capabilities about storage, bandwidth and processing power. While mobile peers are facing with some problems as battery exhaustion, network latency and low bandwidth, less memory, weak processors and small storage. Now, we focus on the purpose of two new main functions as search and store data. In search feature, there are two main actions that this feature interacts in the system such as search peers/friends, and search statuses. For search peers, it will help users can find out where their friends are through sending request to super peers that already made connection. After that, the super peers will check list of managing sub-nodes, if it is not found the result, the super peers will search as more super peers as possible in the overlay network then forwards the request to them. Hence, this feature needs a processing power and strongly memory that normal mobile cannot satisfy. In store data, it is also similar with peers but there are some differences. For peers, store data feature is only saving data of one user who is using the device logged in the social network. While super peers are not only saving data of user logged in but also saving social data for list of managing peers. In addition, list of peers data are stored in term of a copy data and super peers forward these data to some other. Therefore, this feature needs a storage capability to store much data of list managing users that normal mobile peers cannot satisfy because they are usually limited storage. However, peers can become super peers if they have sufficient capability.

3.1.2 - Components inside peers

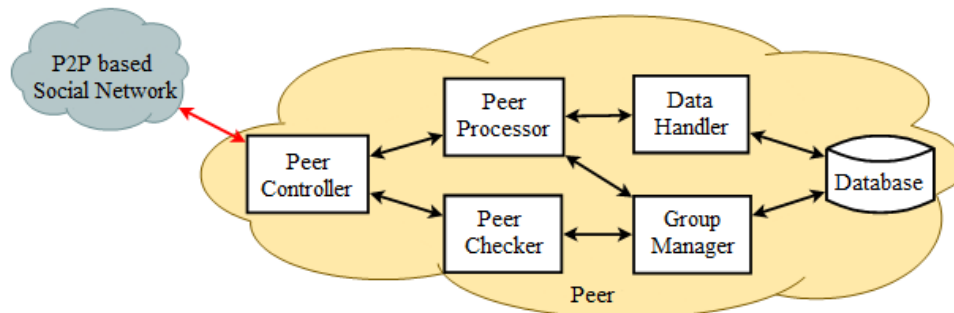


Figure 11- Peer components and communication [1]

Figure 11 presents all of components in both peers and super peers.

There are six similar components in the system such as Peer Controller, Peer Processor, Peer Checker, Data Handler, Group Manager and Database. However, each of components is exist some different functions in peers and super peers.

- *Peer Controller*: manages communication and exchanges information among peers and components, e.g., messages for checking peer identities, messages for checking peers alive, messages for joining peer groups, messages for posting peer status, etc.

- *Peer Checker*: takes a role of registration servers to authenticate peers. This component can request certain user registration information from the registration servers or super peers. In super peers, it is associated with user registration information stored on the database, while this component of the peer only contains itself registration information and a list of super peers.

- *Peer Processor*: keeps tracks on peer activities related to the posting message service that managing peer information including status and profile,

processing posting messages, communicating peer groups, etc. Moreover, this essential component closely works with the data handler and group manager components. In addition, there are 3 main functions in this component that in both peers and super peers including message filter, news feed, and user posting functions. Also, supper peers are additional search function. Now, each of functions is shown more detail as follows:

- *Message filter function*: this component used to filter various types of messages in the posting message service. This function will receive and filter messages and then forward to News feed or User posting functions. In this function, notification feature is an interesting and helpful feature, which used to inform the users that has been added to his or her wall in case of their friends wrote comments or like on user's shared statuses.

- *User posting function*: uses to publish user status messages, sends and receives user's status messages that just related to user who are using the device to logged in the social network in case of both super peers and peers. In more detail, all information related to users' statuses as like, comment that will be managed in this function, e.g., a user writes a status on their wall, and their friends like or comment on the status. In consider of user's status messages, which are including two characteristics: public and private. For private status messages, the owner see them on their wall and store it in their device only without forwarding to their friends or super peers. While public status messages are shared with their entire friends groups.

- *News feed function*: this component uses to request and receive friends groups status messages or a set of friends, or one particular friend. There are four main types of messages in this function: request newsfeed, respond newsfeed, request profile, respond profile messages, which are described more detail in expanding Gnutella protocol of next part. In more detail, it will get a list of friends that already defined friends groups in Group Manager component, or just one particular friend, and then send a request to them to get their shared statuses. However, in super peers, it is not also keeping these functions as peers, but also collects a set of copy statuses from the list of managing peers and through Data Handler component to save all of them into its database and concurrent require Group Manager component to forward these statuses to friends who defined in the message.

- *Search function*: this component only has in super peers because it requires processing and storage capabilities. There are two kinds of features that search function supporting such as search friends and statuses. For each of search features in search function, after super peers received requests from peers to search friends or statuses, they will send the request to other peers and super peers. Upon receiving a request, this component obtains and processes relevant social data from device's database and returns the response, which is described in expanding Gnutella protocol.

- *Data Handler*: uses to update and store status messages on both peers and super peers. This component communicates with *peer processor* and *database* to update and store users' statuses. In addition, this component supports optional data

function to provide to users two options storing their own social data when they have written behaviors on their wall: store in local device, or super peers' database. The data handler component of peers is much simpler than the super peers, which stores not only their own data but also managing peers data when they are required. In fact, when a user has written a status on their wall, optional data function is activated to ask the user in which their social data will be stored in local device or super peers' database. If the user chooses local database, the copy of status will be stored in the device's database. This is one of advantages to help users can keep their own social data on their own device. Otherwise, users will store their data on super peers' database; it will send a request to the connected super peers to store their own data. After that, this super peers will send this copy data to 5 neighbor super peers to distributed user social data to avoid losing data when single point failure of a super peer occurring. In addition, in case of both options, a copy data will be sent to their friends groups through super peers.

- *Group manager*: is responsible for group formation, manages and forwards user status messages. This component maintains the stability of various peers groups and facilitates the information exchange of peers. Group communication is a main feature of the posting message service. In more detail, the purpose of this feature is that reduce traffic network of communications in overlay social network. In addition, this feature in peers uses to keep tracks entire close friends' statuses and interacts with them who formed to a group. Whereas, super peers are not only having these activities as peers but also forwarding these status messages to these friends

who defined in the messages. In fact, in the social network, one user can make friends with many other people. So, there are many redundancies messages that are not interested with the user cause' traffic network. Hence, group communication feature is a solution to solve this problem that define a group of friends or set of friends. We consider this feature based on group communication defined by the previous study [29] and the proposal Applying MapReduce Framework to Peer-to-Peer Computing Applications [5]. In these proposals, the main functions of peers contain contributing computing resources and forming computing groups. In more detail, while connecting to the network, a peer maintains a list of connections to the friend peers. To form a group, the peer sends the group messages to the friend peers that respond with the join messages including the same group identifier of the group message and their identifying if they agree to join in the group, where the group and join message defined in the study [5]. In fact, for group communication feature in this system, firstly each peer needs to maintain list of friends identifying before it join to the social network. After that, when a user writes a status message, they will send the message to the connected super peers. Then, the super peers will forward the message to whole friends groups or a set of unique friends, or a particular one who be attached in the message.

- *Database*: stores all record of user's statuses and profile in the user's device. However, in super peers, database is not only store user data, but also list of peers' data that managing by the super peer.

3.2 – Progresses in posting message service

In the proposal posting message service of the social network, there are three main actions: update user's statuses, update friends group's statuses, and update particular friend's statuses. In general, for each of connections in each progresses following, they always have a return value to inform that these actions were completed or uncompleted. Now, the detail of these progresses are shown as follows:

3.2.1 – Update user's statuses progress

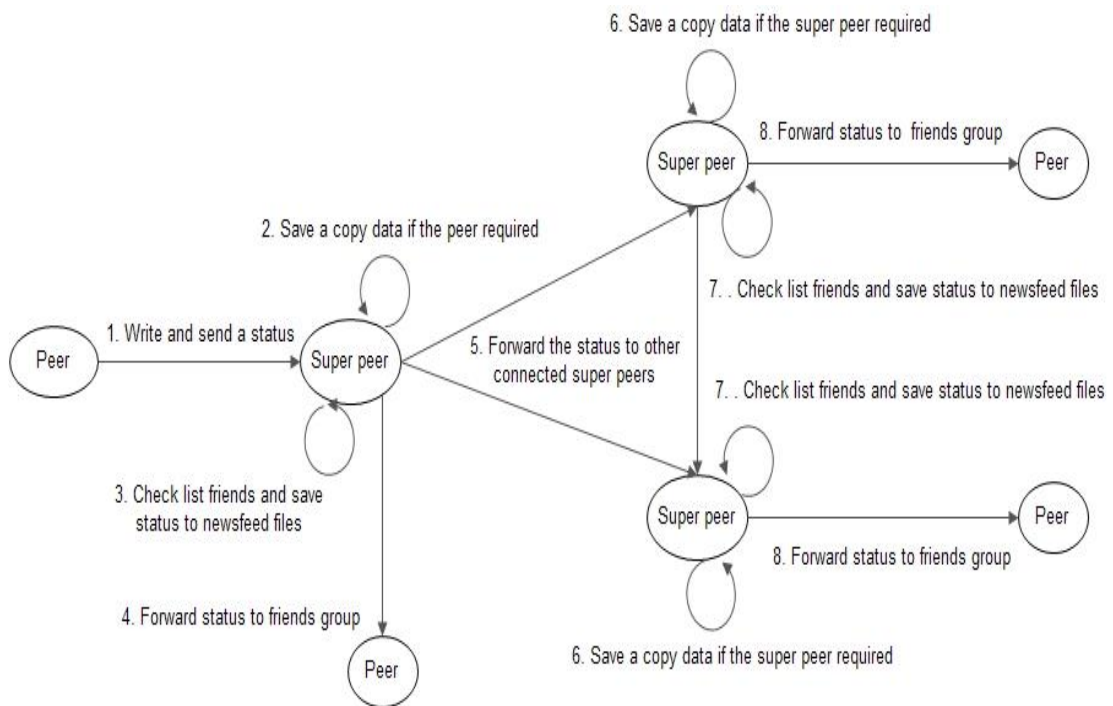


Figure 12 - Update user's statuses progress

Figure 12 describes update user's statuses progress on the proposal posting message service. The purpose of this progress are that proposes a new approach to distribute user's social data on several super peers instead of one single server as

client-server model, and forwards user's statuses to their friends . Now, step-by-step posting a status progress are shown as follows:

1. User writes a status on their home page. After that, this status will be sent to connected super peer. Also, optional data function on peer will be launched to ask the user in which their social data will be stored that in local device or super peers' database. If they choose super peers' database, step 2 and 6 will be activated. Inversely, these steps will be ignored.
2. Super peer, which connected to user, will store a copy data into database if they are required.
3. Check whether list friends' newsfeed files that friends are defined in the status, are storing in the super peers. If the super peer are storing these files, they will save the status into these files to help friends can update their statuses and others also. Otherwise, the super peer will send the status to other connected super peers to forward the message to friends group as in step 5.
4. Super peer send the status to users in the friends group.
5. Super peer will send the message to defined friends group through other super peers, which are connecting with their friends.
- 6, 7, 8. In step 6, 7, and 8 are the same with step 2, 3, and 4 respectively.

3.2.2 – Update friends group’s statuses progress

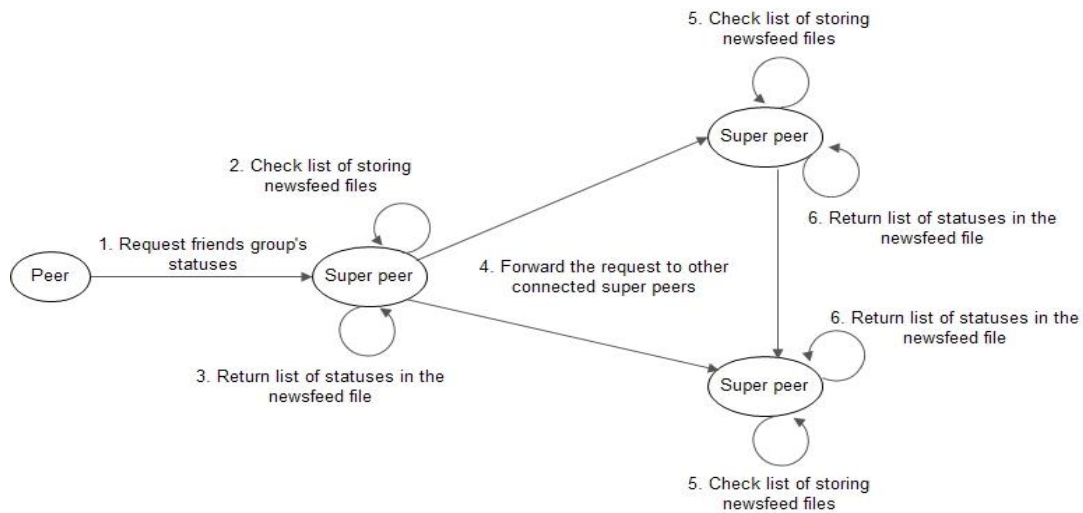


Figure 13- Update friends group’s statuses progress

Figure 13 describes update friends group’s statuses progress that uses to update all new shared friends group’s statuses, which are stored in their newsfeed file at super peers. This is a function to keep tracks their friends’ behaviors that indispensable function in a social network. Now, step-by-step update friends group’s statuses progress are shown as follows:

1. User sends a request to connecting super peer to update all of new shared friends group’s statuses.
2. Super peer receives the request and check list of storing newsfeed files whether requesting user’s data was storing at database.
3. After that, if the super peer are storing these data of requesting user, it will return a response message to requesting user only including all of statuses in their newsfeed file. Inversely, if the super peer is not storing these data, step 4, 5 and 6 will be activated.

4. If the super peer is not storing these data, it will forward the request to other connecting super peers to help user keep tracks their friends' behaviors.

5, 6. In step 5 and 6 are the same with step 2 and 3 respectively.

3.2.3 – Update particular friend's profile progress

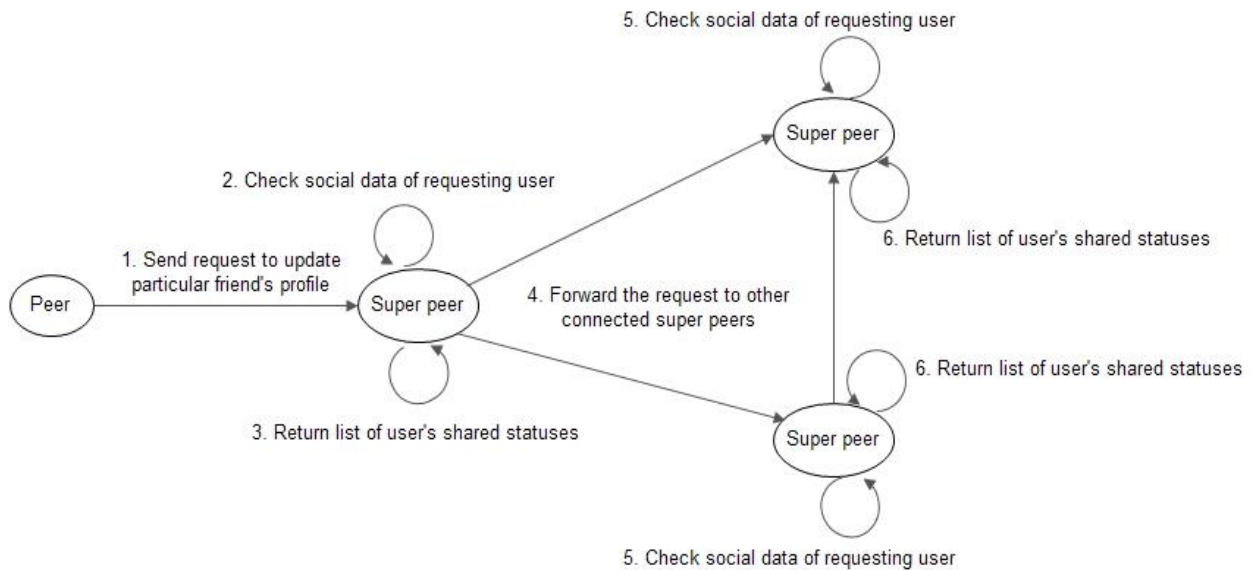


Figure 14 - Update particular friend's profile progress

Figure 14 describes update particular friend's profile progress that uses to update all profile of a particular friend. This is an activity to help users view and keep tracks all shared statuses of a specified one. Now, step-by-step update particular friend's profile progress are shown as follows:

1. A user sends requests to connecting super peer to update particular friend's profile.

2. Super peer receives the request and check list of storing files' name, which are user's identifying in the social network, whether the social data of requesting user are storing at database.
3. After that, if the super peer are storing these data of requesting user, it will return a response message to requesting user only including all of shared public statuses of requesting user in their profile file. Inversely, if the super peer is not storing these data, step 4, 5 and 6 will be activated.
4. If the super peer is not storing these data, it will forward the request to other connecting super peers to help user keep tracks particular friends' behaviors.
- 5, 6. In step 5 and 6 are the same with step 2 and 3 respectively.

3.2.4 – Acknowledgment messages progress

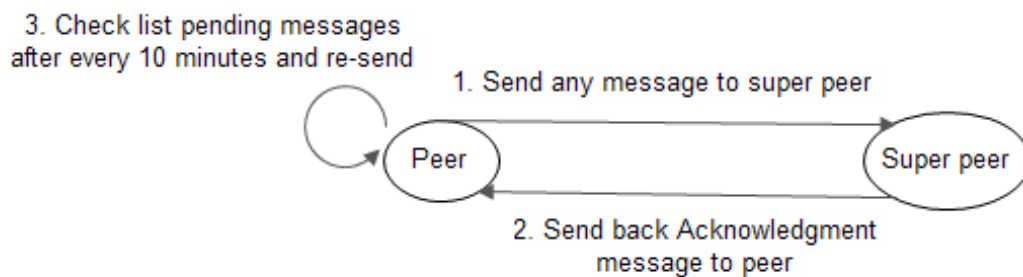


Figure 15 - Acknowledgment progress

Figure 15 describes an acknowledgment progress, which is used to confirm a super peers received the message. This is an activity to help peers avoid losing messages when they are sending to a super peer, which is unexpected shutdown. Now, step-by-step acknowledgment progress are shown as follows:

1. A user sends a message to connecting super peer such as posting, like, comment messages, etc.
2. When a super peer has received the message, they will send back an acknowledgment message to requesting peer to notify they have received the message.
3. Step 3 will be launched after every 10 minutes in peer to check whether messages are pending because they cannot send to any super peers. After that, peer will re-send pending messages to super peers.

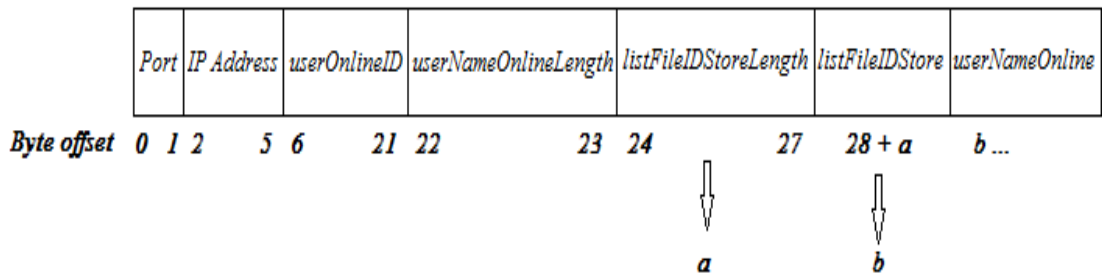
3.3 – Expanding Gnutella protocol

The Gnutella P2P protocol [4] is most suitable for posting message service due to several reasons. First, this protocol supports super peers for solving the problem of incapable peers, e.g., super peers are responsible for routing queries. Second, this unstructured and flooding-based protocol provides flexible query processing, thus facilitating both keyword and semantic search methods. Third, this protocol also bolsters data replication for better data availability. Fourth, there are multiple open source implementations of this protocol. The applicability of this protocol to the system is straightforward because this protocol has already been extended for the P2P search systems [20, 21].

In posting message service, the ping messages remain unchanged and the same with the Gnutella header, while there are 9 new types of messages that need to be implemented to enable more efficient. A Gnutella message consists of header and content. The attributes of these messages' content are shown as follows:

a) Pong message

Upon receiving the ping message from a peer, a peer uses the pong message to advertise the publishing shared files and identifying itself through `userOnlineID`, `userNameOnline` and `listFileIDStore` fields. The detailed structure of the content including byte offset is defined as follows:

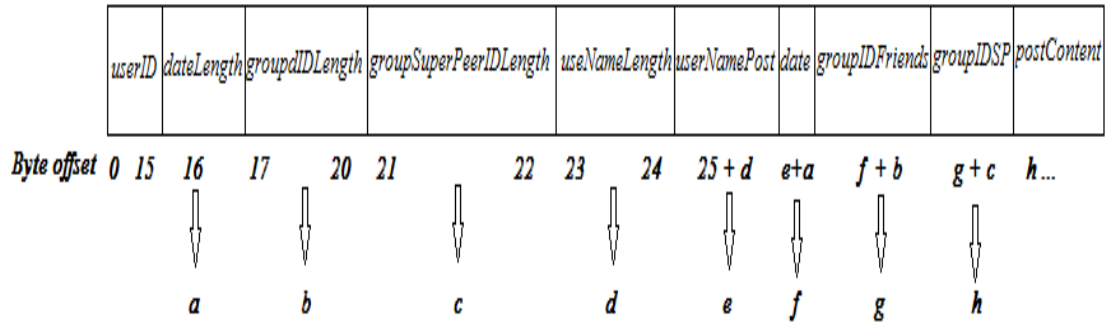


- Port: The port number on which the responding host can accept incoming connections.
- IP Address: The IP address of the responding host in little endian format.
- userOnlineID: a 16-byte string uniquely identifying the user on the social network.
- userNameOnlineLength: a length of userNameOnline field. The information of userNameOnlineLength field used to determine length of userNameOnline field.
- listFileIDStoreLength: a length of listFileIDStore field. The information of listFileIDStoreLength field used to determine length of listFileIDStore field.
- listFileIDStore: a string of storing file name on peers or super peers.
- userNameOnline: name of user are online in the social network.

b) Posting message

Posting messages/statuses are created when user have written a behavior in their home page and shared them with their friends. In addition, both peers and super

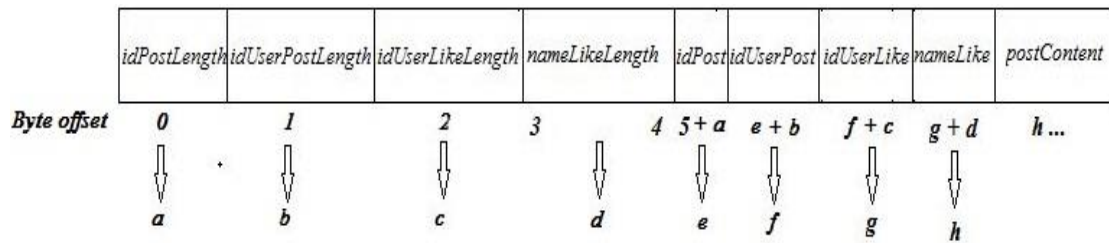
peers are sent unlimited posting messages. The detailed structure of the content including byte offset is defined as follows:



- *userID*: a 16-byte string uniquely identifying the user on the social network.
- *dateLength*: a length of *date* field. The information of *dateLength* field used to determine length of *date* field.
- *groupIDLength*: a length of *groupIDFriends* field. The information of *groupIDLength* field used to determine length of *groupIDFriends* field.
- *groupSuperPeerIDLength*: a length of *groupIDSP* field. The information of *groupSuperPeerIDLength* field used to determine length of *groupIDSP* field.
- *userNameLength*: a length of *userNamePost* field. The information of *userNameLength* field used to determine length of *userNamePost* field.
- *userNamePost*: name of user who created the status/posting message
- *date*: time that created the status/posting message.
- *groupIDFriends*: a group of friends' identifying. The information of *groupIDFriends* field used to determine who can receive this message. This field supports for group communication [5].
- *groupIDSP*: a group of super peers' identifying. The information of *groupIDSP* field used to limit which super peers will save the message to avoid one posting message can be stored at many super peers.
- *postContent*: content of the status/ posting message.

c) Like message

Like messages are created when user pressed Like button to express their interested on the status. One user only like once on one status, and number of like messages on a status will be counted by super peers, which are storing both posting and like messages. The detailed structure of the content including byte offset is defined as follows:

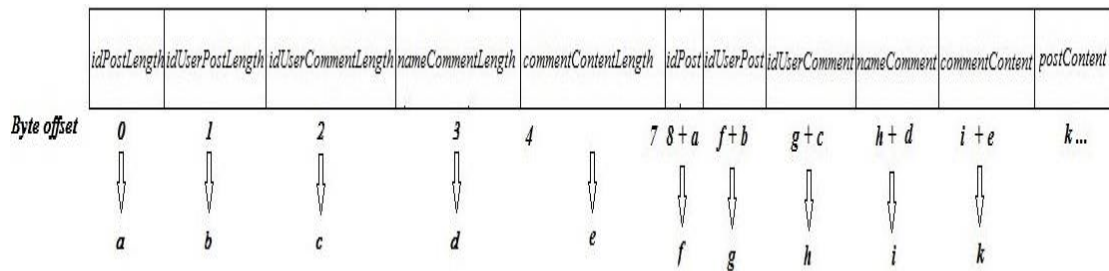


- *idPostLength*: a length of *idPost* field. The information of *idPostLength* field used to determine length of *idPost* field.
- *idUserPostLength*: a length of *idUserPost* field. The information of *idUserPostLength* field used to determine length of *idUserPost* field.
- *idUserLikeLength*: a length of *idUserLike* field. The information of *idUserLikeLength* field used to determine length of *idUserLike* field.
- *nameLikeLength*: a length of *nameLike* field. The information of *nameLikeLength* field used to determine length of *nameLike* field.
- *idPost*: a string uniquely identifying the posting message on the social network. The *idPost* field in like messages is used to firm a relationship between like and posting messages.
- *idUserPost*: a string uniquely identifying the user on the social network who created the status/posting message.
- *idUserLike*: a string uniquely identifying the user on the social network who liked the status/posting message.
- *nameLike*: name of user who liked the status/posting message.

- postContent: content of the status/ posting message, which the user liked in.

d) Comment message

Comment messages are created when user pressed Comment button and wrote their behaviors on the status. One user is unlimited number of comments on a status, and number of comment messages on a status will be counted by super peers, which are storing both posting and comment messages. The detailed structure of the content including byte offset is defined as follows:

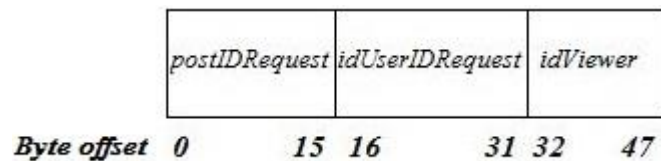


- idPostLength: a length of idPost field. The information of idPostLength field used to determine length of idPost field.
- idUserPostLength: a length of idUserPost field. The information of idUserPostLength field used to determine length of idUserPost field.
- idUserCommentLength: a length of idUserComment field. The information of idUserCommentLength field used to determine length of idUserComment field.
- nameCommentLength: a length of nameComment field. The information of nameCommentLength field used to determine length of nameCommentfield.
- commentContentLength: a length of commentContent field. The information of commentContentLength field used to determine length of commentContent field.

- *idPost*: a string uniquely identifying the posting message on the social network. The *idPost* field in comment messages is used to firm a relationship between comment and posting messages.
- *idUserPost*: a string uniquely identifying the user on the social network who created the status/posting message.
- *idUserComment*: a string uniquely identifying the user on the social network who commented on the status/posting message.
- *nameComment*: name of user who commented on the status.
- *commentContent*: content of comment on the status/posting message.
- *postContent*: content of the status/ posting message, which the user commented on.






e) Request like and comment message

Request like and comment messages are created when a user would like to see more detail of a status including like and comment. This message is sent to super peers only, which are storing the detail of requesting status/posting message, because number of like and comment messages on a status are stored at super peers only. The detailed structure of the content including byte offset is defined as follows:



- *postIDRequest*: a string uniquely identifying the posting message on the social network. The *postIDRequest* field is used to determine content of the requesting status/posting message.
- *idUserIDRequest*: a string uniquely identifying the user on the social network. The *idUserIDRequest* field is used to help super peers that determine whether this user social data was storing.

- ### f) Respond like and comment message

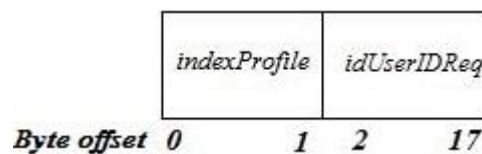
	<i>postIDReq</i>	<i>numLike</i>	<i>numComment</i>	<i>userIDReqLength</i>	<i>listNameLikeLength</i>	<i>listCommentLength</i>	<i>userIDReq</i>	<i>listNameLike</i>	<i>listComment</i>	<i>idViewer</i>					
<i>Byte offset</i>	0	15	16	17	18	19	20	21	22	23	24	25 + <i>a</i>	<i>d</i> + <i>b</i>	<i>e</i> + <i>c</i>	<i>f</i> ...
															
							<i>a</i>					<i>d</i>	<i>e</i>	<i>f</i>	

- 43

- **userIDReq:** a string uniquely identifying the user on the social network. The **idUserIDRequest** field is used to compare requested user that sent from peer with the response from super peer.
- **listNameLike:** a string of users' name likes in the status/posting message.
- **listComment:** list of content of comments that commented in the status/posting message.
- **idViewer:** : a string uniquely identifying the user who are sending the request. The **idViewer** field is used to determine whether the receiving user required the request.

g) Request profile message

Request profile messages are created when user would like to see all of shared public statuses of a specified user or themselves. This messages are sent to super peers only, which are storing this user social data. The detailed structure of the content including byte offset is defined as follows:

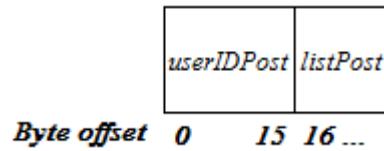


- **indexProfile:** an index of the last status/posting message in the Profile field. The information of **indexProfile** field is used to load next statuses/posting message, which are sent from the peers.
- **idUserIDRequest:** a string uniquely identifying the user on the social network. The **idUserIDRequest** field is used to help super peers that determine whether this user social data was storing.

h) Respond profile message

Upon receiving the request profile message from a peer, a super peer uses the respond profile message to provide all shared public statuses of the requested user

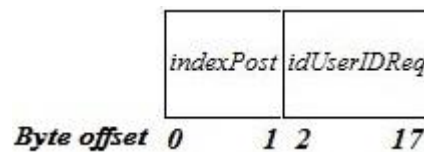
through listPost field. This message is only sent to the user who sent the request. The detailed structure of the content including byte offset is defined as follows:



- *userIDPost*: a string uniquely identifying the user on the social network who created the status/posting message. The *userIDPost* field is used to compare requested user that sent from peer with the response from super peer.
- *listPost*: list of statuses/posting messages that shared from the requested user.

i) **Request newsfeed message**

Request newsfeed messages are created when user would like to see all of received statuses from their friends to update all of their friends' behaviors and express the interested in these statuses. This messages are sent to super peers only, which are storing these statuses/posting messages. The detailed structure of the content including byte offset is defined as follows:

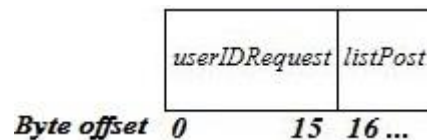


- *indexPost*: an index of the last status/posting message in the newsfeed field. The information of *indexPost* field is used to load more statuses/posting message, which are sent from the peers.

- **idUserIDReq:** a string uniquely identifying the user on the social network. The **idUserIDReq** field is used to help super peers that determine whether this user social data was storing.

j) Respond newsfeed message

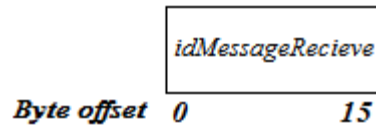
Upon receiving the request newsfeed message from a peer, a super peer uses the respond newsfeed message to provide all of received statuses from their friends through **listPost** field. This message is only sent to the user who sent the request. The detailed structure of the content including byte offset is defined as follows:



- **userIDRequest:** a string uniquely identifying the user on the social network who created the status/posting message. The **userIDRequest** field is used to compare requested user that sent from peer with the response from super peer.
- **listPost:** list of shared statuses/posting messages' content that saved at super peers for the requested users.

k) Acknowledgment message

Upon receiving any messages from a peer, a super peer sends back an acknowledgment message to the requesting peer to confirm they received the message. This message is only sent to the peer who sent the message. Also, this message helps a peer avoid losing the message when connecting super peers unexpected shutdown. The detailed structure of the content including byte offset is defined as follows:



- *idMessageRecieve*: a string uniquely identifying of message on the social network. The *idMessageRecieve* field is used to help super peer confirm that they received the message and send back to requesting peer.

CHAPTER IV – RESULTS AND EVALUATIONS

4.1 – Results

4.1.1 – Home page

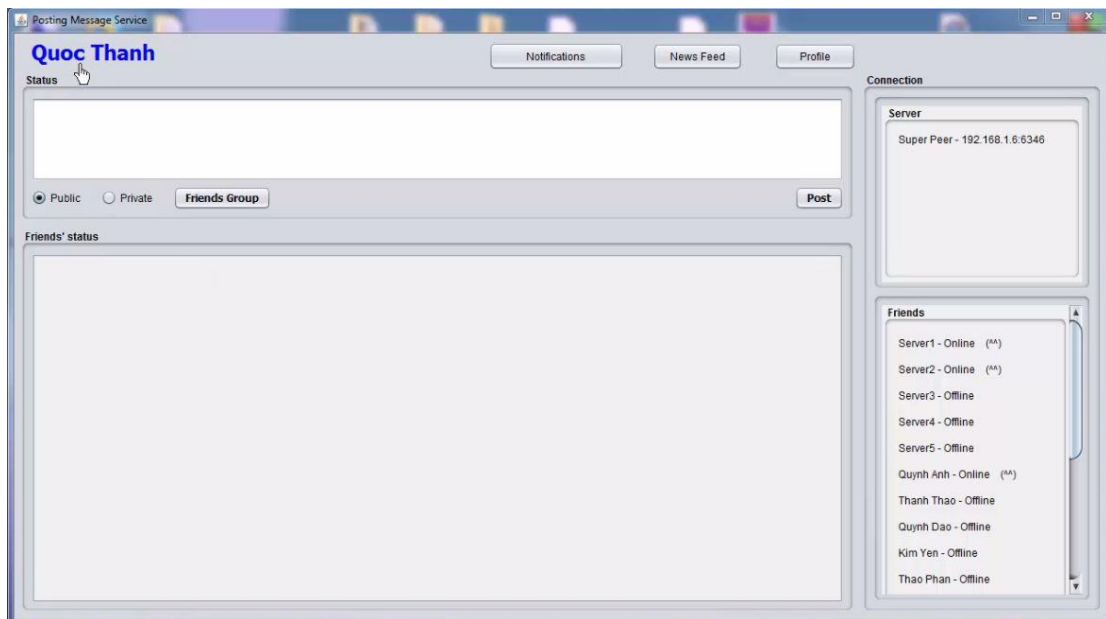


Figure 16 - Posting message home page

Home page will be open after user authentication successful as in Figure 16.

In this page, users can write and share their behaviors with their friends through Status field. In addition, users also update all of shared friends' statuses via Friends' status field. At the right side of home page, users will see all of their friends and list

of super peers that they are connecting to join the social network. Moreover, users will receive notifications when other users like or comment on their shared statuses. Also, users will update all of shared friends' statuses when they pressed News Feed button in case of they were offline before. In addition, Profile button will help users update all of statuses, which they shared before. Last but not least, Friends Group button will help users define a group of friends to determine who can receive their statuses.

4.1.2 – Super peer and Friends connections

At the right side of the home page is list of super peers and friends that a user is connecting. These connections are automatically offline when they shut down and vice versa (see Figure 17).

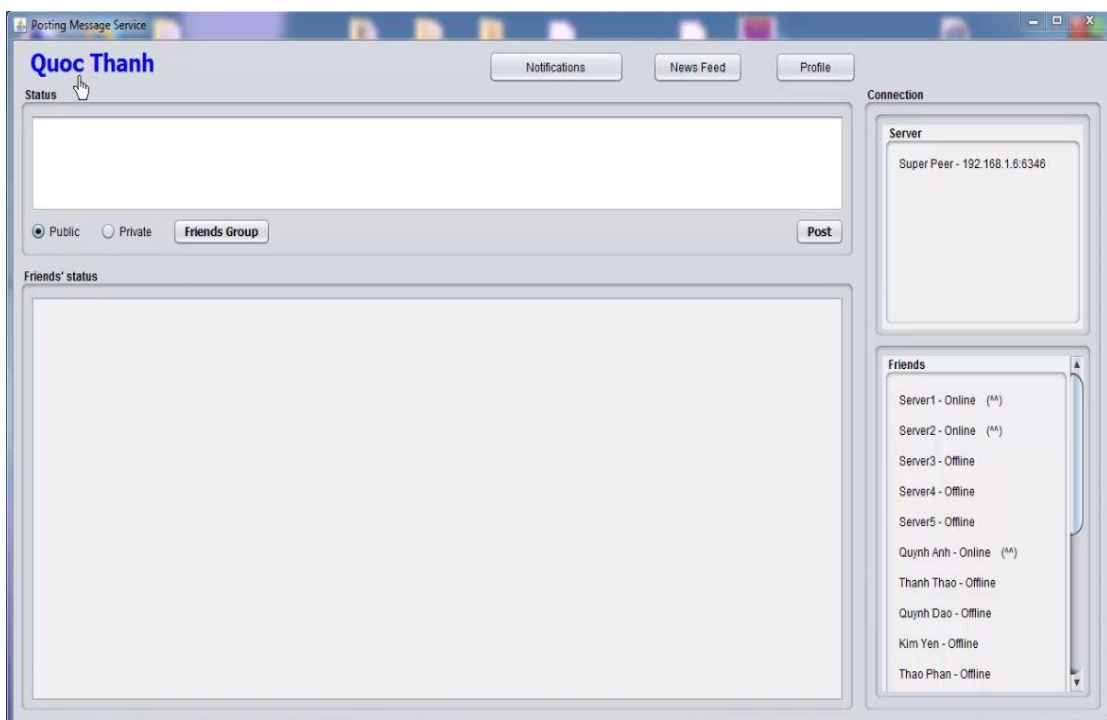


Figure 17 - Super peer and Friends connections

4.1.3 - Update user's statuses

a) Posting a public status

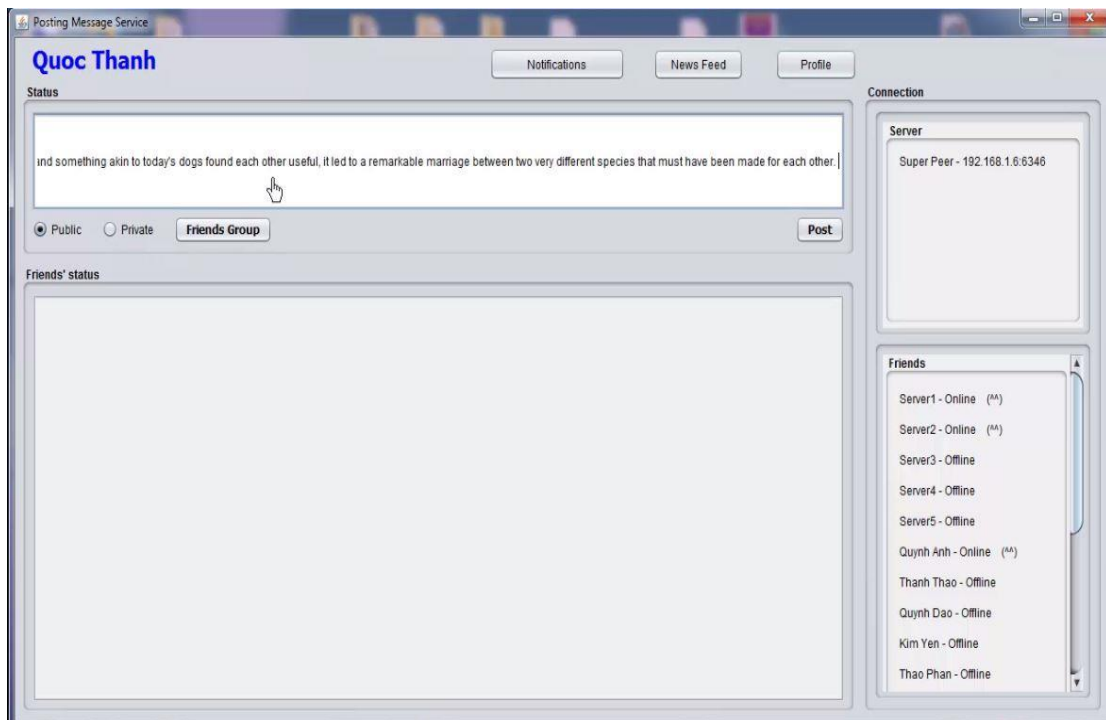


Figure 18 - Writing a public status

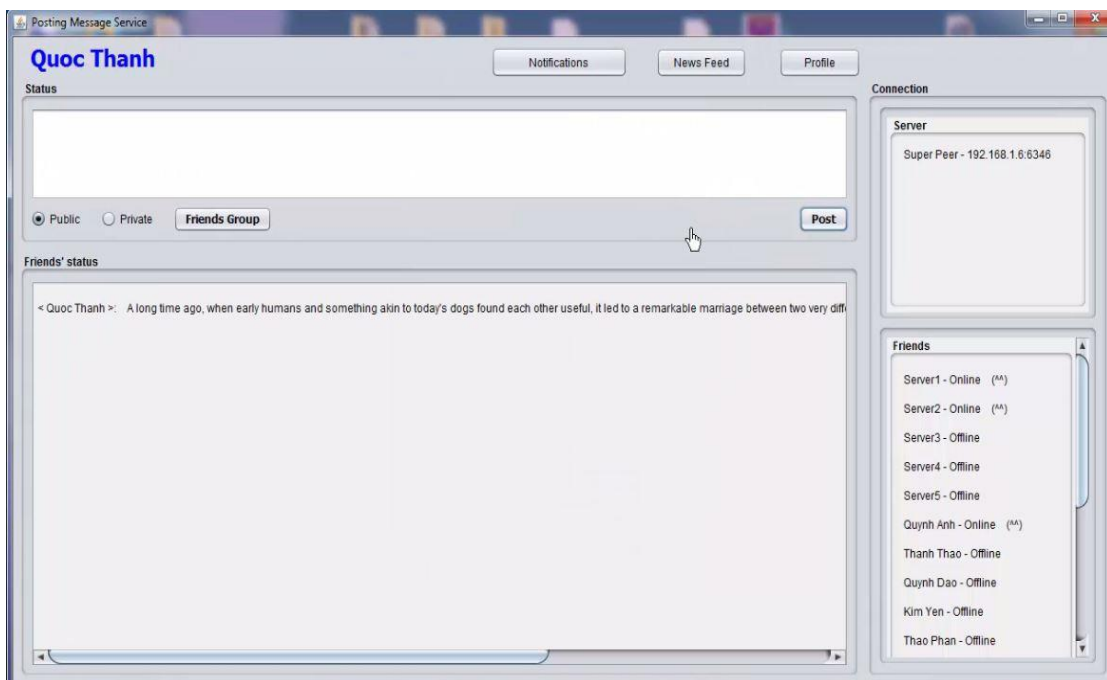


Figure 19- Showing public status after posting

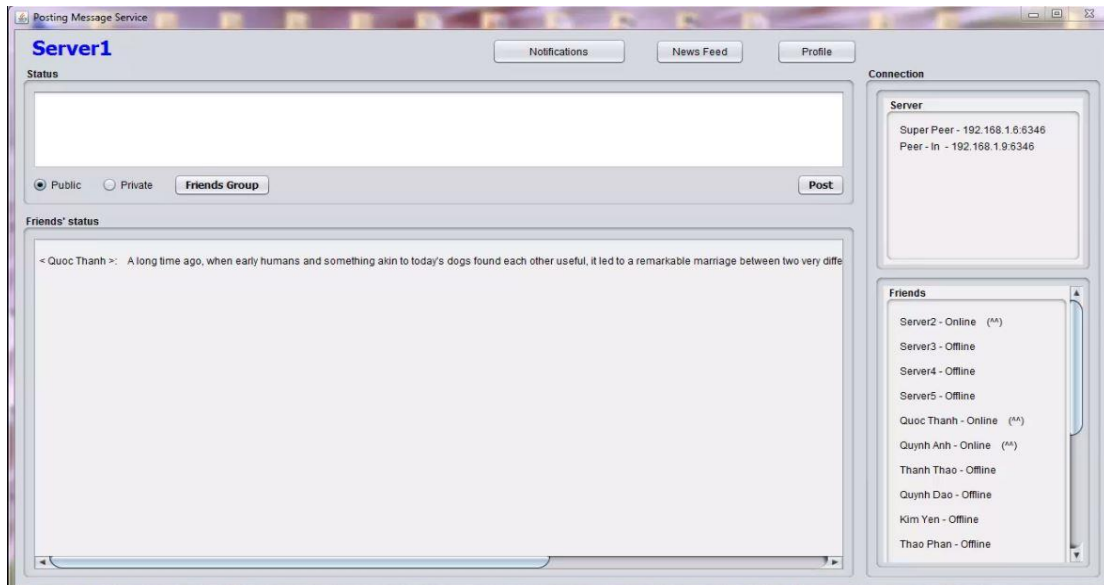


Figure 20- Friends receive and show status on Friends' status field

In the Status field, which allows a user writes and shares a public status with their friends (see Figure 18). After they posted the status on the Internet, their home page and friends' home page will receive and show the status on Friends' status field as in Figure 19 and 20.

b) Detail of a status

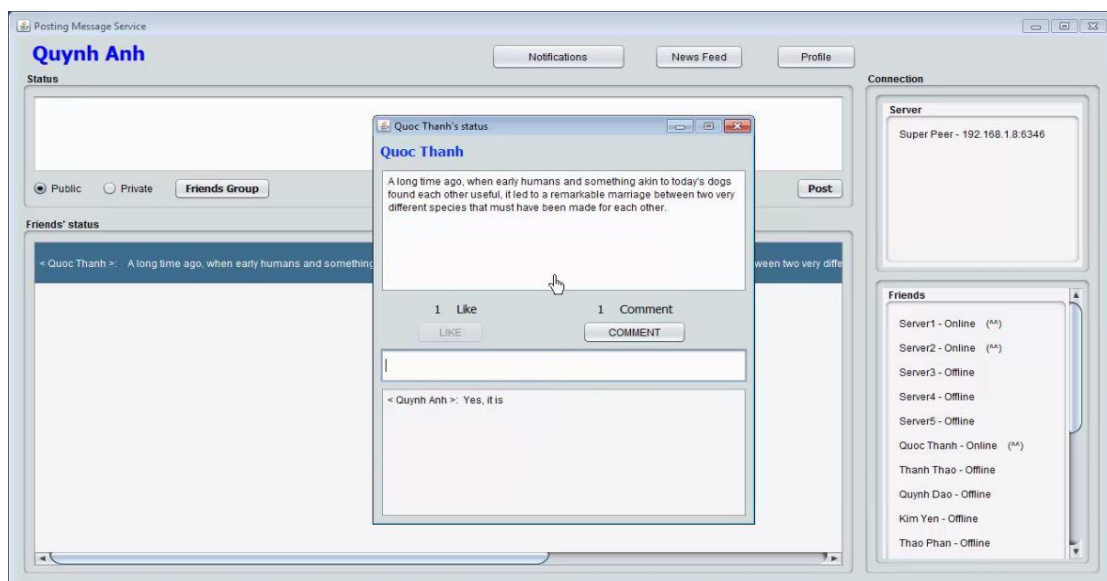


Figure 21- Detail of a status including like and comment

Figure 21 presents that the detail of specified status including content, number of likes and comments. In addition, a user can write unlimited comments and press Like button on the status to express their interested. However, one user only like once on one status.

c) Posting a private status

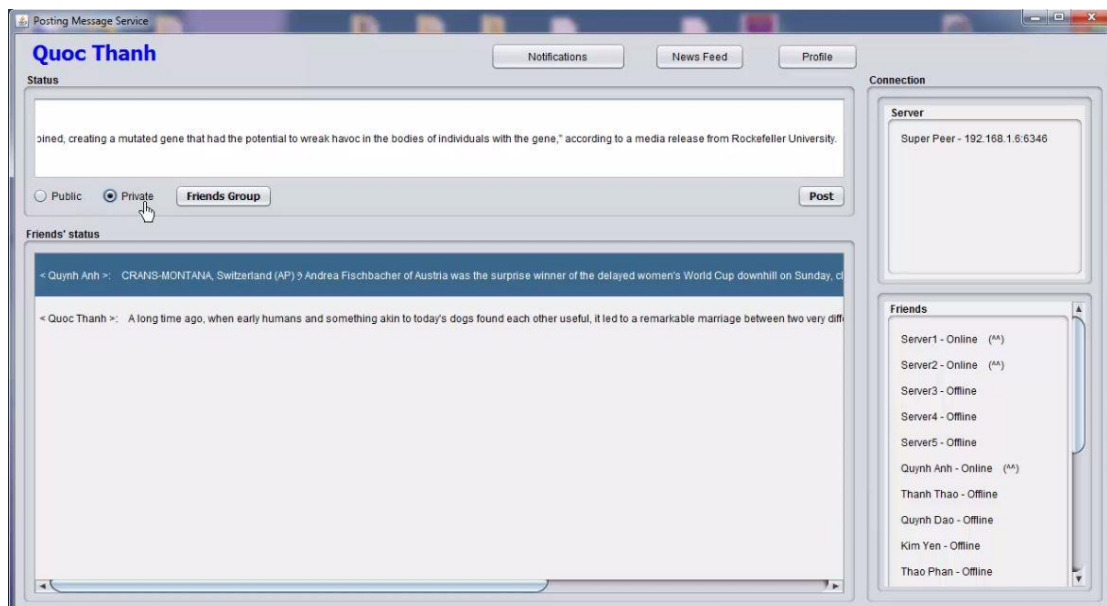


Figure 22- Writing a private status

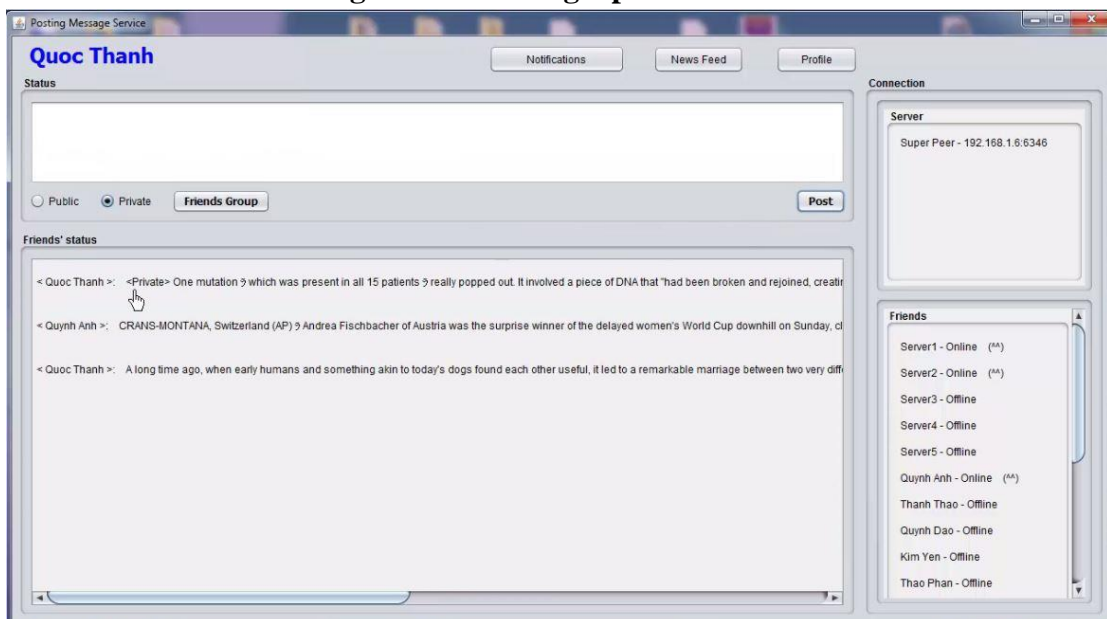


Figure 23- Showing private status after posting

In posting message service, there are 2 types of statuses/posting messages such as public and private posting messages. For public statuses/posting messages, they will be sent to defined group of friends and forwarded by super peers. Whereas, private statuses/posting messages are only see by the owner without forwarding to friends or super peers. In addition, private statuses are stored at user's device only (see Figure 22 and 23).

d) Update all user's profile

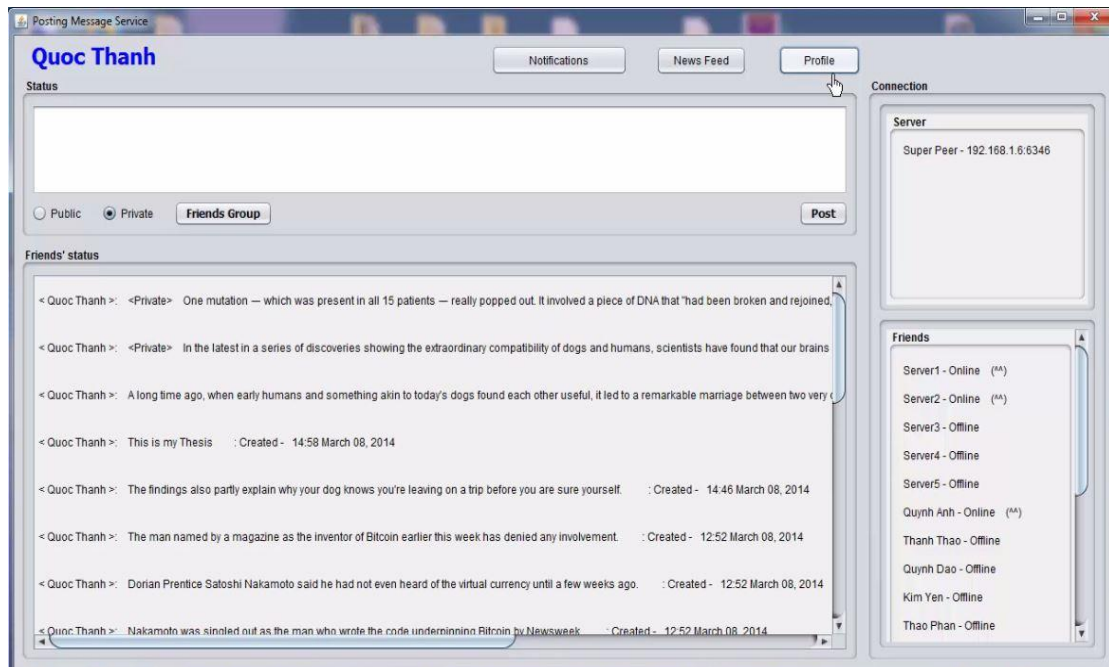


Figure 24- After click Profile button

A user can review all of their shared statuses including public and private statuses when they pressed Profile button as in Figure 24. For public statuses, they will be loaded from super peers, which are storing this user social data. Whereas, private statuses will be load from user's device database, which stored the private statuses.

4.1.4- Update friends group's statuses

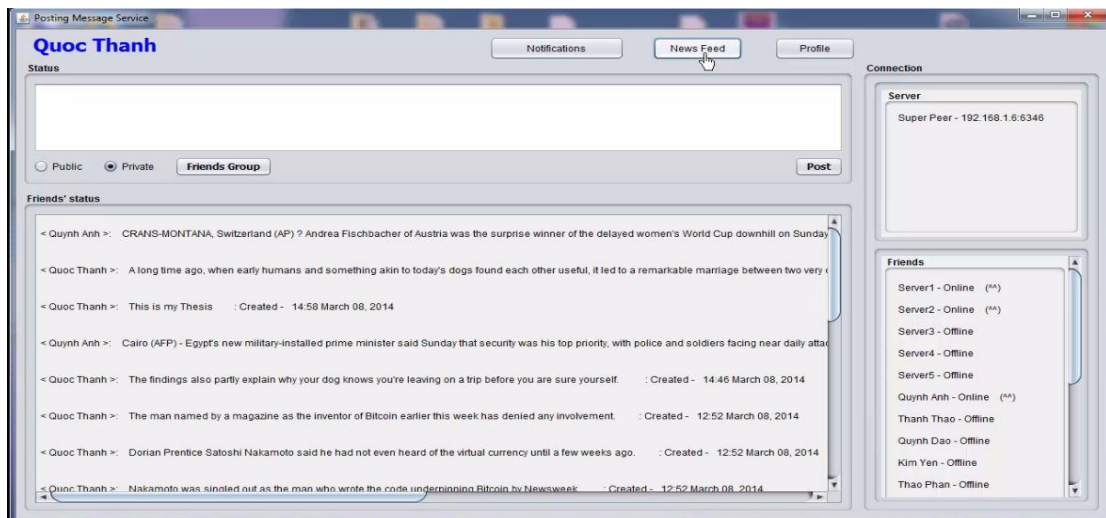


Figure 25- After click News feed button

Figure 25 describes that all of received statuses from user's friends, which looks like News Feed page of Facebook [4]. When user pressed News Feed button, it will load all of shared friends' statuses from super peers. In addition, a user can communicate with their friends through like or comment on friends' statuses to express their interested.

4.1.5 - Update particular friend's profile

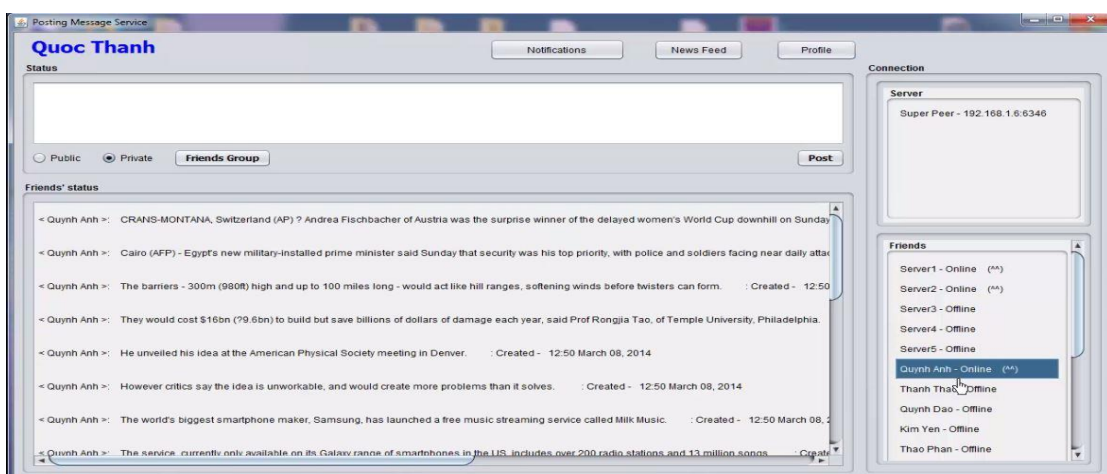


Figure 26 - After click Friend's name

Update particular friend's profile is a very useful function to help users can keep tracks all of shared statuses from a specified friend when they pressed in friend's name as in Figure 26. This function will send a request to super peers, which are storing this user social data. Then, super peers will send the response back to the user who sent the request only.

4.1.6 – Optional functions

a) Notification function

Notification function is very necessary and familiar with users. This function uses to notify to users who have liked and commented on their statuses/posting messages as in Figure 27.



Figure 27- Notification

After a user clicked on Notification button, the home page will show a dialog to describe briefly who liked and commented on user's status as in Figure 28.

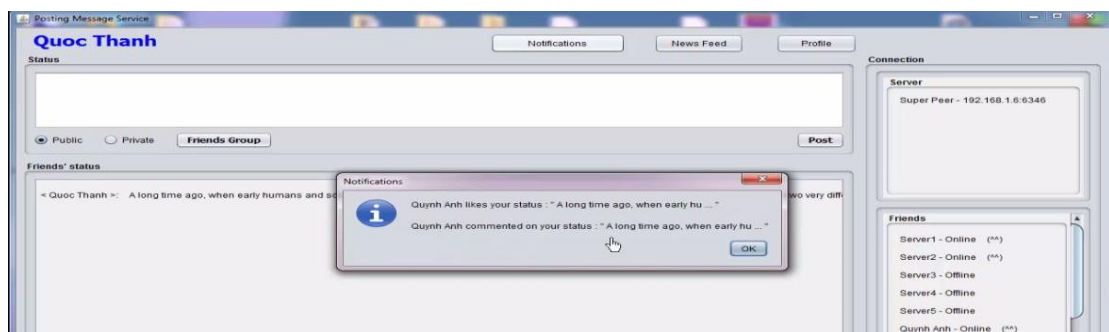


Figure 28- After click Notification button

b) Setting group of friends function

This function will be open when user click on Friends group button to define their group of friends that will receive their statuses to avoid their statuses are forwarded to unfamiliar users (see Figure 29). This function supports for group communication [5].

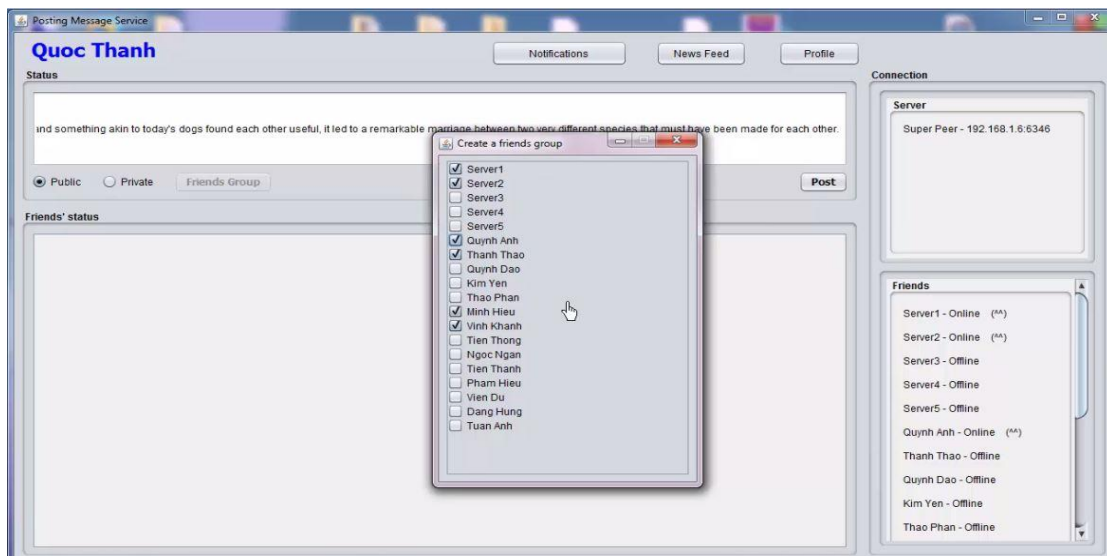


Figure 29- After click Friend Group button

c) List of users liked a status

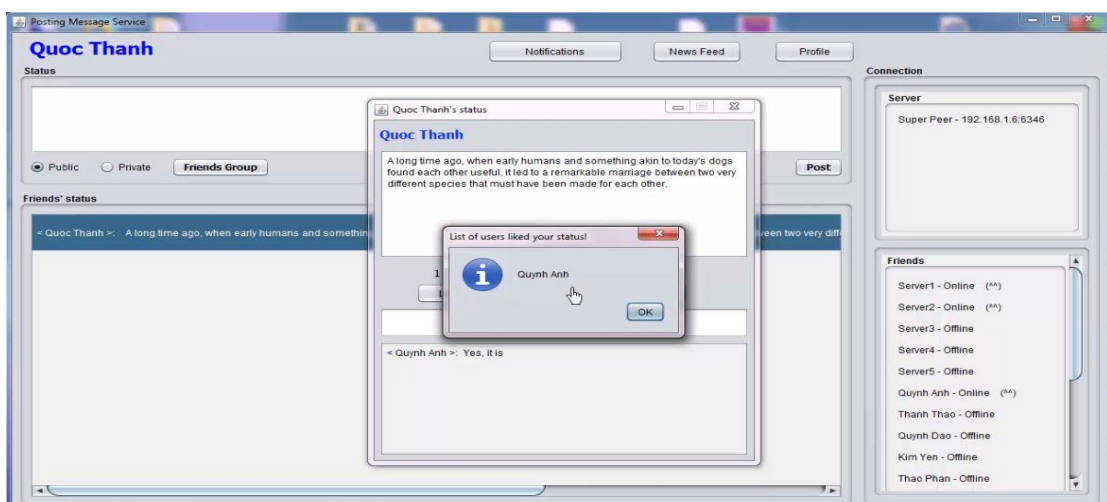


Figure 30- List of liked users

Figure 30 presents a dialog that shows list of users' name who liked the status. This dialog will be open when a user clicked in the number of likes on the status. In fact, this function is very useful to help users know who friends liked on their status.

4.1.7 – Database structure

In posting message service, there are 3 types of messages that will be stored at peers or super peers such as posting, like, and comment messages. For super peers, these messages will be stored based on list managing peers while peers are only store user's posting messages. In fact, super peers need to store like and comment messages to help synchronization data of a status. Super peer will provide number of likes, comments and content of comments as well on the specified status to users who are requesting. In addition, these messages will be stored in term of a record in a text file; and file name is user's identifying, who is the owner of these statuses/posting messages. This structure helps super peers that be easy searching whether the specified user social data was storing. The detailed structure of these messages are defined as follows:

a) Posting message

<i>PostID</i>	<i>NamePost</i>	<i>StatusContent</i>	<i>GroupFriendID</i>	<i>CreatedDate</i>

- PostID: a string uniquely identifying the posting message on the social network. The PostID field is used to detect whether a posting message has already created in the social network.
- NamePost: name of user who created the status/posting message.

- StatusContent: content of the status/ posting message.
- GroupFriendID: a group of friends' identifying. The information of groupIDFriends field used to determine who can receive this message.
- CreatedDate: time that created the status/posting message.

b) Like message

<i>PostID</i>	<i>LikeID</i>	<i>IDUserPost</i>	<i>IDUserLike</i>	<i>NameUserLike</i>
---------------	---------------	-------------------	-------------------	---------------------

- PostID: a string uniquely identifying the posting message on the social network. The PostID field is used to firm a relationship between like and posting messages.
- LikeID: a string uniquely identifying the like message on the social network. The LikeID field is used to detect whether a like message has already created in the social network.
- IDUserPost: a string uniquely identifying the user on the social network who created the status/posting message.
- IDUserLike: a string uniquely identifying the user on the social network who liked the status/posting message.
- NameUserLike: name of user who liked the status/posting message.

c) Comment message

<i>PostID</i>	<i>CommentID</i>	<i>IDUserPost</i>	<i>IDUserComment</i>	<i>NameUserComment</i>	<i>Content of Comment</i>
---------------	------------------	-------------------	----------------------	------------------------	---------------------------

- PostID: a string uniquely identifying the posting message on the social network. The PostID field is used to firm a relationship between comment and posting messages.

- **CommentID**: a string uniquely identifying the comment message on the social network. The CommentID field is used to detect whether a comment message has already created in the social network.
- **IDUserPost**: a string uniquely identifying the user on the social network who created the status/posting message.
- **IDUserComment**: a string uniquely identifying the user on the social network who commented on the status/posting message.
- **NameUserComment**: name of user who commented on the status/posting message.
- **Content of Comment**: content of comments on the status/posting message.

4.2 – Evaluations

Based on several evaluation requirements of the P2P based social network [1], I choose to focus on connectivity and reliability rather than performance and security. First, we have applied the Skype authentication service [20] to this social network with a similar super peer P2P architecture, security issues can be reduced considerably. Second, performance issues are less important than connectivity and reliability considering the posting message service. These characteristics are shown as follows:

4.2.1 – Connectivity

Number of connection times	Time (s)
1	0.264
10	0.3787
100	0.73486

Table 3- Connectivity

Table 3 reports time statistics for one peer connecting to one super peer that try to join in the social network. In this statistic, we can see time to connect in duration from 0.3 to 1 second. In fact, this connection time is fast and stable though 100 connection times. Hence, users will be easy and fast to join in the social network with highly stable connectivity.

4.2.2 – Reliability

Number of messages	Time for Super peer receiving (s)	Time for 3 peers receiving (s)
1	0.402	1.1907
500	0.39208	1.1891
1000	0.373444	1.1873
1500	0.355062	1.1854
2000	0.338291	1.1838
2500	0.322152	1.1821

Table 4 - Reliability

For reliability, we consider whether all of messages delivered to target peers and measure the time for messages that are delivered to 1 super peer and 3 other peers as in Table 4. During time of evaluation, all of messages are delivered to target peers, so we just focus on time for messages delivery. From these statistics above, it shows time for 1 super peer receiving large of messages from peers are about 0.5 second, while time for the super peer forwards these messages to other peers about 1.2 seconds. In fact, these time are fast and stable for users who would like to share these statuses to their friends.

Because of the limited time and experiences, the evaluations are not quite accurate. If I have time, I will run posting message service in a number of laboratory workstations to test effectiveness when the service works on many peers. However,

with current evaluations, they are also presenting apart of feasibility of the posting message service on decentralized online social network.

CHAPTER V – CONCLUSION AND FUTURE WORKS

I have proposed an architecture and implemented a prototype for posting message service on decentralized online social networks, which users can have more control over their privacy and dissemination of their information. Moreover, this service possesses a P2P architecture that contains peers and super peers. Users will participate the network as peers that have an optional to store a copy of user social data. Whereas, peers with sufficient capability of storage, bandwidth and processing power can become super peers that are responsible for complex operations as searching social data, host of list peers social data, etc. In addition, super peers can store and forward messages to groups of peers. I have used the Gnutella protocol to implement the prototyping posting message service. Because of the limitation of time and knowledge, the completion of this study focuses on proposal an approach for P2P social network and implements a prototype to confirm feasibility of the approach. So, I am not able to complete a fully application for posting message service. However, the prototype are presenting feasibility of posting message service on decentralized online social network.

Future work considers the possibility of extending posting message service to bind with authentication service on social network. Also, posting message service will need to synchronize and encrypt data between super peers. In addition, current posting message service is running on local network, but it will be passed to the

cellular and internet network in the future. Also, this service will improve graphical user interface to be familiar with users as Facebook by adding some functionalities such as sharing photo, video, link or tagging, check-in location, making friends, etc. Last but not least, this service will be expanded to other platforms as mobile and web that help users joining to the social network easier and more convenient.

LIST OF REFERENCES

- [1]. Ha Manh Tran, Khoi DuyVo, Thanh Quoc Le, Thao Thi Thanh Phan. “Peer-to-Peer Based Social Network”. NICS conference: <http://www.nafosted.gov.vn/nics2014/>
- [2]. Laura Solomon. Teens, Tweens and Social Networking. [Online]. <https://infopeople.org/civicrm/event/info?id=89&reset=1>. Accessed December 14th, 2013.
- [3]. H.M. Tran, K.V. Huynh, K.D. Vo, S.T. Le: Mobile Peer-to-Peer Approach for Social Computing Services in Distributed Environment. 4th International Symposium on Information and Communication Technology (SoICT 2013), DaNang City, Dec. 2013. ACM
- [4]. Facebook features http://en.wikipedia.org/wiki/Facebook_features#News_Feed
- [5]. H.T. Dang, H.M. Tran, P.N. Vu, and A.T. Nguyen: Applying MapReduce Framework to Peer-to-Peer Computing Applications. 4th International Conference on Computational Collective Intelligence Technologies and Applications (ICCCI 2012), Ho Chi Minh City, November 2012. Springer LNAI 7654. [BibTex].
- [6]. Jovanic, M. 2000. Modelling large-scale peer-to-peer networks and a case study of Gnutella. M.S. Thesis, Department of Electrical and Computer Engineering and Computer Science, University of Cincinnati, Cincinnati, OH 45221.
- [7]. B.Yangand H. Garcia-Molina. Designing a super peer network. InProc.19th International Conference on Data Engineering (ICDE’03), page 49, Los Alamitos, CA, USA, 2003. IEEE Computer Society.
- [8]. Facebook Social Network. <http://www.facebook.com/>. Last access in Oct. 2013.
- [9]. Myspace Social Network. <http://www.myspace.com/>. Last access in Oct. 2013.
- [10]. Twitter Social Network. <http://www.twitter.com/>. Last access in Oct. 2013.

- [11]. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content Addressable Network. In Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01), pages 161–172, New York, NY, USA, 2001. ACM Press.
- [12]. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In Proc. Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01), pages 149–160, New York, NY, USA, 2001. ACM Press.
- [13]. P. Maymounkov and D. Mazie`res. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS '01), pages 53–65, London, UK, 2002. Springer-Verlag.
- [14]. D. M. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.
- [15]. Gnutella Protocol Specification version 0.4. <http://rfc-gnutella.sourceforge.net/developer/stable/index.html>, 2001. Last access in Mar. 2013.
- [16]. Clarke, O.Sandberg, B.Wiley, and T.W.Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Proc. International Workshop on Design Issues in Anonymity and Unobservability, pages 46–66, Heidelberg, Germany, 2000. Springer-Verlag.
- [17]. B. Cohen. Incentives Build Robustness in Bittorrent. In Proc. 1st Workshop on Economics of Peer-to-Peer Systems, 2003.
- [18]. Leucio Antonio Cutillo, RekMolva, and Thorsten Strufe. Safebook: A Privacy-Preserving Online Social Network Leveraging on Real-Life Trust, *IEEE Communications Magazine* • December 2009
- [19]. Jian Liang, Rakesh Kumar, Keith W. Ross. The KaZaA Overlay: A Measurement Study. September 15, 2004
- [20]. Salman A. Baset and Henning Schulzrinne Pouwelse. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. September 15, 2004
- [21]. J.A., Garbacki, P., Wang, Jun, Bakker, A., Yang, J., Iosup, A., Epema, D.H.J., Reinders, M.J.T., Steen, M. van & Sips, H.J. (2011). TRIBLER: a social-based peer-to-peer system. *Concurrency and Computation: Practice & Experience*, 20(2), 127-138. in Web of Science Cited
- [22]. Sonja Buchegger, Doris Schiöberg, Le H. Vu, AnwitamanDatta. PeerSoN: P2P social networking: early experiences and insights. In *SNS '09: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems (2009)*, pp. 46-52, doi:10.1145/1578002.1578010

- [23]. Luca Maria Aiello, Marco Milanese, Giancarlo Ruffo, Rossano Schifanella: “An identity-based approach to secure P2P applications with Likir”, Computer Science Department C.SoSvizzera, 185 - 10149 Torino, Italy
- [24]. Decentralized Online Social Networks. Anwitaman Datta, Sonja Buchegger, Le Hung Vu, Thorsten Strufe, and Krzysztof Rzażda.
- [25]. Decentralization: The Future of Online Social Networking, Ching-man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, Tim Berners-Lee school of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK, Decentralized Information Group, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
- [26]. TechMediaWorld.com. (2009) TechMediaWorld.com. [Online]. <http://social-networking-websites-review.toptenreviews.com>. Accessed December 14th, 2013.
- [27]. J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Commun. ACM, 51:107–113, January 2008.
- [28]. Rüdiger Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In Proceedings of the First International Conference on Peer-to-Peer Computing (P2P’01). IEEE Computer Society, 2002.
- [29]. Stefan Saroui, P. Krishna Gummadi, and Steven D. Gribble. Measurement Study of Peer-to-Peer File Sharing Systems. Technical report, Department of Computer Science & Engineering, University of Washington, 2002.