Decentralized online social network primitives

Rajesh Sharma

2013

Rajesh Sharma. (2013). Decentralized online social network primitives. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54955>

<https://doi.org/10.32657/10356/54955>

NANYANG TECHNOLOGICAL UNIVERSITY

Decentralized Online Social Network Primitives

by

Rajesh Sharma

A thesis submitted to the

School of Computer Engineering

Nanyang Technological University

in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

January, 2013

Acknowledgments

I would like to express my sincere appreciation to my supervisor Dr. Anwitaman Datta for his unending patience and guidance. He not only inspired me by sharing his sharp insights on the problem, but also encouraged me by his passion and enthusiasm in pursuing research.

I am fortunate enough to get the blessings of the Supreme Power which is omnipresent. No word can express my thanks to it which has always guided and helped me. My wholehearted gratitude goes to my parents, for their unconditional love and continuous encouragement in my studies. I owe them a lot for the sacrifices they have made for me. I would also like to thanks my sister and brother-in-law.

I am also grateful to my colleagues and friends, in and out of NTU who have helped me in one way or another during my PhD stay at NTU. Tu Trong Hieu thanks for spending so much time on my papers and your inputs were really helpful. I learnt a lot from our discussions. You are a really cute, kind and nice person. Be like that ! Nguyen High Long, Stefano, Anthony, Prof. Sarunas, Deepak Subramanian, Liu Xin, Kokil, Prof. Krz - discussions with you guys helped me in refining my various works and research in general. Shirin, Manvi, Piyush, Amrith and Divya - thanks for proof reading my papers and thesis. I would also like to thank my friends for their support and friendship - Kid, Manisha, Shubs, Jing Dear, Guangxia Li, Anand, Tran Huy Duc, Rajat and Arun. Thanks Mohan and Cedric, parties at your house were indeed refreshing. I

am also thankful to my sister Jaspal and aunty Sayna for the love and care they have shown. Many thanks to them for taking care of my diet especially during YOG.

I would also like to thank Thomas for his administration and technical support at CAIS center. You were really very helpful in booking the meeting room on short notices.

Abstract

Centralized online Social networks such as *Facebook* are highly popular [1]. Despite their success, a prominent concern about such centralized systems is privacy [2] [3] from the service providers themselves. In order to address these shortcomings, researchers have proposed decentralized architectures for social networks. Examples of such Decentralized Online Social Networks (DOSNs) include Peerson [4], Vis-a-Vis [5] and Safebook [6].

Typically in a DOSN each autonomous entity has a local view of the network based on social links it has. In other words, in the absence of global knowledge because of decentralized architecture, a node can only explore its local information, i.e., its own knowledge as well as knowledge shared by its neighbors related to it using some social relations (friends, relatives etc.). This is challenging due to the fact that problems need to be solved without global coordination of local information and local links.

We explore “distributed social computing” to solve various challenges in a DOSN. Social computing is an area where the social behavior plays an important role for any computational systems. Distributed social computing is an emerging area which focuses on solving problems in distributed systems by leveraging social relations (such as interest, trust, expertise, etc.). In a centralized system, collective wisdom of the users is relatively easy to gather to solve a problem. However in a decentralized system, it is not trivial to achieve the same because of the typical characteristics of the environment of decentralized systems - (1) network is decentralized and scalable, and (2) no node can have

global knowledge about the system because of the decentralized nature of the network (or because of resource constraint).

In this report, we investigate issues and challenges in a DOSN that can be termed as more difficult than that in a centralized online social network as there is no central authority, which can cater users for various services. We solve various problems in DOSN in a distributed manner with constraint of social links. Specifically we aim at social network applications like storage, information dissemination and ranking. The main contribution of our research is that in our proposed solutions, nodes exploit only their local knowledge (i.e. their own information and information gathered from their immediate neighbors) by using distributed social computing approach. We present several results which show the effectiveness of our approach. We also highlight various future directions with respect to our current work.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	ix
List of Tables	xi
I Preliminaries	1
1 Introduction	2
1.1 Overview	2
1.2 Problem Statement	5
1.2.1 Information Dissemination:	5
1.2.2 Data Availability in DOSN:	6
1.2.3 Ranking in distributed/decentralized settings:	6
1.3 Research scope and solutions/contributions	7
1.3.1 Information Dissemination:	7
1.3.2 Data Availability in DOSN:	9
1.3.3 Ranking in distributed settings:	10
1.3.4 Connection among three primitives.	11
1.4 Dissertation outline	12

2	Literature Survey	14
2.1	Distributed Systems	15
2.1.1	Peer-to-peer Systems	15
2.2	Challenges in DOSN	18
2.2.1	Information Propagation	19
2.2.2	Data Availability in DOSN	22
2.2.3	Ranking in Distributed Systems	24
2.2.4	DOSN Economics	26
2.2.5	Privacy and security in DOSN	29
II	Challenges in DOSN	35
3	Information Dissemination	36
3.1	Introduction	36
3.1.1	High level objective	37
3.1.2	Motivation	37
3.1.3	Difference between GoDisco and GoDisco++	38
3.1.4	Contribution	39
3.2	Approach	39
3.2.1	System Model	40
3.2.2	The intuitions for GoDisco++'s design	40
3.2.3	Concepts	41
3.2.4	GoDisco++	46
3.2.5	Complexity of the algorithm	49
3.3	Evaluation	50
3.3.1	Results	52
3.4	Conclusions	60

4	Data Availability in DOSN	62
4.1	An empirical study of availability in friend-to-friend storage systems . . .	62
4.1.1	Introduction	62
4.1.2	Ego-Centric Network Characterization	64
4.1.3	Data Placement at Friends	66
4.1.4	Results	68
4.1.5	Conclusions	74
4.2	SuperNova	74
4.2.1	Introduction	74
4.2.2	Approach	75
4.2.3	Evaluation	86
4.2.4	Conclusions	96
4.3	Epilogue	97
5	DRank: Decentralized ranking mechanism for semantic community overlays.	98
5.1	Introduction	99
5.2	Approach	101
5.2.1	System Model	102
5.2.2	Algorithm 1: Simple Case	102
5.2.3	Algorithm 2: Complex Cases	106
5.3	Evaluation	108
5.3.1	Dataset	109
5.3.2	Results	110
5.4	Conclusions	117

III	Concluding remarks and future directions	119
6	Conclusions and Future Work	120
6.1	Summary	120
6.2	Future Directions	123
6.2.1	Information Propagation	123
6.2.2	Data availability in DOSN	125
6.2.3	Ranking	127
6.3	List of Publications	127
IV	At the periphery of main research	129
7	Appendix	130
7.1	A tunable graph model for incorporating geographic spread in social graph models	130
7.1.1	Introduction	131
7.1.2	Related Work	134
7.1.3	The Social Graph Generation Model	136
7.1.4	Evaluation	139
7.1.5	Conclusion	150
7.2	Acronym Table	150
	References	152

List of Figures

1.1	Relationship among different primitives.	12
3.1	Examples of different types of synthetic graphs.	52
3.2	Message dissemination	54
3.3	Recall & precision	56
3.4	α and γ comparison on the random clique based network.	57
3.5	Saved duplication	58
3.6	Volume of duplicates	59
3.7	Qualitative dichotomy of spamming effect	60
4.1	Ego-centric network characteristics.	70
4.2	Microscopic analysis.	71
4.3	Macroscopic analysis.	72
4.4	Neighbors used for data placement.	73
4.5	Relation between different entities.	78
4.6	Performance of different parameter combinations for Deviation (D) and Non Deviation (ND)	90
4.7	Comparison for Friend's Time(FT) and Total Time(TT) for Deviation (D) and NonDeviation(ND)	94
4.8	Comparison for Flat(F) vs. Super-peer(SP) for Deviation (D) and Non- Deviation(ND)	95

4.9	Comparison for Best(B) vs. Worst(W) parameter combination for Deviation (D) and NonDeviation(ND)	96
5.1	Figure showing the sample graph.	105
5.2	Iterations vs. % of nodes	117
7.1	Node migration as well as creation of remote edges are leveraged in our social graph model. After migration, nodes may also discard some edges from the previous region. Such an aging process is also incorporated in our model, though it has not been shown in this figure.	134
7.2	Out-location distribution in various scenarios	139
7.3	Geographic-clustering compactness distribution in various scenarios . . .	145

List of Tables

2.1	Past literature work with respect to information dissemination.	32
2.2	Past literature work with respect to data availability.	33
2.3	Past literature work with respect to ranking.	34
4.1	Different combination of parameters	89
5.1	Algorithm's execution on a pseudo network	105
5.2	Iteration values in the simple case	112
5.3	Iteration values in the complex case	112
7.1	Geographical distribution phase parameters	135
7.2	Macroscopic characteristics for different scenarios	146
7.3	Acronyms	151

Part I

Preliminaries

Chapter 1

Introduction

1.1 Overview

Web 2.0's applications such as social networking and blogging are quite popular among internet users. The online social networks (OSNs) such as *Facebook*, *LinkedIn*, *Twitter* have large user bases [7], [1], [8]. Users post messages, upload photos, videos and various other personal information on OSNs. Thus, these OSNs act as storekeepers for the users' data which is growing rapidly with each passing day. OSNs such as *Facebook* and *Google+* are centralized in nature and are managed by a single authority like *Facebook* and *Google* respectively. In spite of being successful, these centralized systems have some inherent issues associated with them, such as privacy issues (not only from other end users, but also from the service providers themselves) which are very prominent and serious [2], [3], [9]. Along with privacy issues, the service providers of centralized infrastructure are often accused of unwanted censorship [10], [11], [12]. The service providers may act as Big Brother for the users. As an alternative to centralized systems, efforts have been made by researchers to come up with decentralized online social networks (DOSNs) such as [13], [14], [6] where the objective is to let users remain as the owners of their data instead of relying upon third party service providers. The proposed architectures mainly advocate users to collaborate among themselves by using peer-to-peer (P2P) approaches.

The path to the decentralization of online social networking itself is however not an easy one to tread. Using end user resources in a collaborative/peer-to-peer manner does not naturally mean scalability or robustness in terms of quality of service, on the contrary, an internet-scale DOSN run by autonomous and individually unreliable peers is bound to have poorer performance when compared with a well provisioned, centrally managed dedicated infrastructure based service. Also, the functionalities being provided by *Facebook* and other centralized architecture based OSNs are difficult to implement because of the inherent decentralized architecture.

One of the functionalities provided by OSNs is to find users based on similar interests. A user might be interested in finding other users based on similar interests because either he/she is interested in getting more friends or wants to disseminate information related to a particular interest. In centralized OSNs, a user can query the centralized server (which has information about all the users) to find users with a particular interest. A centralized server can facilitate the querying user by returning all the users having that particular interest. This returned information can then be used by the querying user for dissemination purpose. However, the problem of focussed dissemination is non-trivial in a DOSN as there is no central authority which has information of all the users which can help in focussed dissemination. One of the problems on which we worked on is how to disseminate the information in a focussed and efficient way in a decentralized network with the aim of maximum coverage among nodes which might be interested in a particular topic and with minimum spamming among the nodes which have no interest in the topic.

Also, many centralized OSNs provide (this is not a universal rule, but just a prevailing practice) free storage to all the registered users at the expense of privacy [2, 3]. A natural benefit is that a user's data stays available to his friends even if the user is offline. In a DOSN, there is no central server(s) which is(are) ready to store users' data. If a

data resides only with the user himself/herself then as soon as the user is offline, the data becomes unavailable. Thus, the problem of data availability is another important issue which needs to be addressed. One intuitive solution would be that the users can collaborate amongst themselves, that is, they can replicate their data on other users' machines to increase availability of their data. The direct advantage of storing the data at multiple sites is that even when a user is offline his data can be accessed from other authorized users' machines. In this thesis we have worked on mechanisms to ensure the availability of users' data.

These two problems are the basic functionalities which need to be addressed and are closely related with each other. A mechanism is needed so that the availability of a user's data is not compromised in the absence of a central server. To increase data availability, a user has to share his data with other users. To know other users - a basic mechanism is needed for communication which is efficient so as to broaden the social circle. New and more friends can probably help in increasing the data availability.

Another closely related problem is that of finding top-k nodes with respect to an interest/topic in decentralized settings. A user might be interested in finding top-k nodes in the network for various reasons. For instance, consider a network of email ids of researchers where each email id denotes a researcher and collaboration between researchers denotes the edge between two nodes. This is an example of decentralized and distributed network. In such a decentralized network, a user might be interested in finding the top influential researchers. The search might be motivated with an intention for collaboration in future. However, it is not easy to get information about top-k researchers with respect to a particular topic. In centralized systems such as *Twitter* and *Google scholar* [15], it is easy to find top influential users and researchers respectively. In *Twitter*, one way to find the most influential users can be based on the number of followers of the node. Finding top influential authors is also straightforward using *Google scholar*.

One can check the citation count using *Google scholar* to check the most influential authors.

1.2 Problem Statement

In a community network - either centralized or decentralized, social links are the primary means to bind the network. These social links contain vital information (sometimes hidden) within them and can be explored to solve a problem. For decentralized networks, we advocate that, approaches have to be developed which (i) use local information and (ii) may benefit from exploiting the social links as part of the solution. Such solutions should be scalable enough to cater to a large number of users in the network. In general our research problem is how to solve various challenges of a large decentralized network when global knowledge is not available. To this end, we explored principles of distributed social computing to propose different solutions.

1.2.1 Information Dissemination:

The first problem we studied is that of selective information dissemination in a decentralized semantic social network. The goal of this problem is to find an efficient approach to cover maximum relevant nodes and to spam minimum irrelevant nodes in the network.

In a distributed environment the solutions are mainly based on the publish-subscribe mechanism [16] or uses gossip techniques [17] or explore epidemic based flooding techniques such as [18]. The publish-subscribe mechanism has its own limitations. If the topics for the subscriptions are defined in an abstract way and cover a range of subjects in each topic, then the subscribers might be overloaded with lots of information. On the flip side if the topics are too refined then a user might miss a vital information if he/she is not subscribed to some particular topic. The traditional gossip based approaches assume any node can communicate with any other node. However, in our network settings,

the nodes can only communicate with their neighbors. Although the flooding based approaches can reach out to all the desired nodes, however, they also create high spam in the network which is not desirable. We propose a social principle inspired approach [19] which explore the social links between nodes for targeted dissemination and explore multi-dimensional networks which exist in the network.

1.2.2 Data Availability in DOSN:

The second problem caters to the data availability problem in a decentralized online social network. In the absence of a central storage provider, nodes have to replicate their data at other nodes to increase their data availability. When a node's machine is down, other nodes can act as data providers for data seeking nodes.

Our analysis of real traces of an European instant messaging server [20] has given us the insight that merely replicating the data on friends'(from same locality) machine cannot provide 24X7 data availability. Thus we proposed a super-peers based architecture (SuperNova) to facilitate decentralized online social networking [21]. In SuperNova we also analyzed mix of user strategies on the sustainability of DOSNs. In comparison to our approach, the approaches defined earlier basically present the architecture based on friends [6], [14] or hybrid architecture [4].

1.2.3 Ranking in distributed/decentralized settings:

In the third problem, we try to find top-k nodes in a distributed and decentralized network with respect to a particular interest. The problem of finding top-k users (or nodes) with respect to a particular topic is important. Be it a co-authorship graph where an author is looking for other top-k authors with respect to a topic, or the problem of finding top-k influential nodes with respect to an interest (or topic) in a social network. In large networks, global knowledge is difficult to collect and maintain at individual

nodes because the networks are (i) dynamic in nature and (ii) usually scale to very large numbers. Hence there is a necessity to design algorithms based on locally available information and through local interactions.

In a centralized system it is easier to find the top-k nodes as the centralized server can have all the information. The server can easily find out the top-k based on some attributes. For example, once the google crawler has all the information, the page rank algorithm [22] can find the top web pages based on in-links and out-links. In a distributed settings, each node can only compute the top-k node based on the knowledge it has about itself and its neighbors. This problem has not been explored extensively in decentralized settings. It is thus essential to develop mechanisms to develop a simple algorithm for finding top-k nodes in decentralized settings.

1.3 Research scope and solutions/contributions

In this section, we present the approaches and methodology for the problems discussed above. We also highlight the contributions of our research and how the contributions can be integrated.

1.3.1 Information Dissemination:

- In a DOSN, information dissemination is a problem where a user initiates dissemination with the aim of maximum coverage and minimum spamming across the network without any help from a central authority and by solely relying on its local information. We first propose two techniques for information dissemination, (i) a gossip based and (ii) another using swarm particle. We then propose our third technique in which we take the best out of the above two approaches.
- We propose and investigate a gossip based (GoDisco) [23], social principles and behavior inspired, decentralized mechanism to disseminate information in a dis-

tributed setting, using exclusively social links and exploiting semantic context to keep the dissemination process selective to relevant nodes. We do so by using local information by taking a distributed approach. A node selects a particular neighbor if neighbor's interest matches with that of the message. A node also forwards the message to the potential forwarders selected based on history as a forwarder, activeness, and degree of the nodes. The approach tries to cover maximum coverage of the relevant nodes.

- As an alternative to [23], a swarm particle based propagation algorithm [24] is proposed for efficient message propagation. We exploit multiple relations which exist in networks to get optimal set of forwarders¹. A node selects relevant nodes based on interest and also explores multiple relations for potential forwarders. Swarm particles are used to carry the messages (emulating as agents) and are capable of communicating with each other for maximizing the relevant nodes and minimizing the spam.
- We then propose a more generic solution by taking the best out of the above two approaches in [19]. In this thesis we present only the GoDisco++ approach since it is the ultimate solution we determined based on the multiple stages of development.
- The GoDisco++ approach is generic enough, which can be used in variety of applications, like a Mobile adhoc network or a Delay Tolerant Network. It can also be used for the call for papers problem in a network of emails. It can also be used in Job Market network consisting of employers and employee and the task is to look for suitable employers and employees.
- The approaches are distributed in nature and uses local information for decision making, thus not requiring the global knowledge of the network.

¹The optimal set of forwarders are selected based on two criteria, namely (i) interest and (ii) relations for potential forwarders to help in reaching out to distant communities for that message.

- These designed mechanisms can be useful not only for the distributed social information systems for which we develop them, but may also have wider impact in the longer run - such as in engineering word of mouth marketing strategies, as well as understanding naturally occurring dissemination processes which had inspired us the algorithm designs at the first instance.

1.3.2 Data Availability in DOSN:

- Another interesting problem that we investigated is that of storage. In the absence of central servers, a user has to select nodes for storing its data, so that when he is offline his data is still available to his friends.
- We analyzed real traces of an European instant messaging server to verify if replicating the data with friends is enough for a good availability of users' data[20].
- We defined some key characteristics of an ego-network which influence the achievable availability in an F2F storage system.
- We observe that identification of a minimal set of friends to achieve the maximum achievable coverage is in fact analogous to the set cover problem, and hence NP-hard.
- We thus propose greedy heuristic data placement algorithms for increasing the data availability.
- We evaluate the heuristics using trace-driven simulation. The traces contain the ego-centric network and the up/down-time information of the nodes in the network.
- Based on our analysis of real traces, we propose a new architecture for realizing decentralized online social networks (DOSNs) which is a superpeers based network of volunteer agents (hence the name, SuperNova). The super-peers act as brokers

to find suitable nodes which can act as storekeepers for other nodes and to take care of the new joiners in the network. A user shares his data with his friends and also with some strangers (if desired with the help of super-peers) for maximum availability.

- We investigate (based on simulations) the performance of the proposed system for a static snap-shot, which results from a mix of strategy choices made by the constituent user population. Such extensive exploration (albeit of only a part of the DOSN) of design and parameter spaces is in contrast with most existing DOSN literature, which have focused on the system design and security protocols or have evaluated small scale deployments in controlled environment.
- We propose mechanisms to judiciously utilize user resources, given a mix of user strategies and contributed resources, by detailing the various resource management tasks played by the super-peers.

1.3.3 Ranking in distributed settings:

- We propose a decentralized ranking algorithm for finding top-k users in a semantic social overlay based network. For a particular topic or interest, users can calculate top-k nodes based on their local knowledge and information about their neighbors. Assuming a trusted and fair environment, each node performs this operation and then shares its top-k findings with other neighbors. In this way, each node updates its findings based on the information being gathered from the neighbors. Ultimately the process converges and every node knows about the top-k nodes.
- Our proposed algorithm exploits social links and uses local information only.
- The algorithm scales upto any size of the network.

- We evaluated our approach on real and synthetic dataset to show the effectiveness of our approach.

1.3.4 Connection among three primitives.

In this section, we present the connections among various DOSN primitives. Communication is the most basic primitive of all in any kind of social network. The problem of disseminating a particular message to a specific kind of community becomes more challenging as no user can maintain (or have knowledge) about all the nodes of the system because of resource constraint. We presented a gossip based solution where a node exploits non relevant (which are not interested in the message) nodes to find distant nodes (not connected directly to the originator of the message) which might be interested in the message. As part of the solution, the proposed gossip based algorithm exploits some top-k non-relevant nodes in reaching out to distant relevant nodes. The gossip algorithm can exploit solutions of ranking primitive, which tries to find top-k nodes. Ranking primitive can in turn exploit gossip based random walk approach to converge about the top-k nodes in the system. The third primitive of data availability is also well connected to the problem of ranking and information dissemination. The nodes can use the information dissemination mechanism to make more friends and thus can find more nodes which are ready to act as storekeepers, and among all the potential storekeepers, a node can select top-k nodes using the ranking approach. All the proposed solutions, for all the primitives, are based on social computing and uses local knowledge. Thus these solutions can easily be integrated into a DOSN.

Also, the three primitives being investigated in the thesis can leverage solutions from DOSN economics (dealing with trust and free riding) and, privacy and security² primitives to make the solutions more robust. Figure 1.1) shows the relation among five primitives

²These two primitives are part of our immediate future works.

and the work on the graph model (which is not a primitive), which is inspired by our work on data availability primitive. Each primitive being investigated is shown in shaded box with the chapter name. The label on the incoming edge to a particular primitive box describes how/what that particular primitive can leverage from the other primitive (box, from where the edge is going out).

Figure 1.1: Relationship among different primitives.

1.4 Dissertation outline

This report is organized as follows.

In the first part of this thesis, we provide the preliminary information. After this chapter, where we presented an overview along with motivating context and a brief summary of our research, we next review related works in Chapter 2. We review the existing works related with information dissemination, storage and data availability and ranking. We also present literature about DOSN economics and, privacy and security related issues in a DOSN.

In the second part, we present our approaches and algorithms for various problems that we investigated in distributed and decentralized networks using distributed social computing primitives. In chapter 3, we present GoDisco++, our gossip based information dissemination algorithm. Chapter 4 includes two works. In the first work, we analyzed the real traces of availability of friends. One of most important insight we obtain from this work is - friends from the same locality are not enough for meeting storage system needs. In the second work, we present our approach of super-peers based mechanism to solve the problem of storage and data availability in a DOSN. The algorithm for solving the problem of finding top-k nodes in a decentralized environment is presented in chapter 5.

In the third part, with chapter 6 we conclude this thesis with a summary of research. Future extensions to this research work are also outlined in this chapter.

Finally in part 4 (in chapter 7), we present our work which lies at the periphery of our main research. The work is the outcome of our investigation on storage and data availability. We present a tunable graph model to incorporate the geographic spread which occurs in the networks.

Chapter 2

Literature Survey

In this chapter, we summarize background knowledge and review related works. We first discuss distributed systems in section 2.1. We describe its characteristics taking peer-to-peer systems as the example in section 2.1.1.

In section 2.2, we identify various design issues which needs to be addressed for the realization of a DOSN, specifically (i) section 2.2.1 describe the literature survey related with information dissemination (ii) section 2.2.2 presents the related work for the problem of data availability in a decentralized social network which is another important problem, (iii) section 2.2.3 presents the related work for the problem of finding top-k nodes in distributed and decentralized settings.

We also present related works with respect to two more challenges which we have not explored in this thesis however they are our immediate future works - (i) DOSN economics and (ii) privacy and security in DOSN. In section 2.2.4 (DOSN economics) we present various literatures related with trust and incentive schemes which have been proposed to motivate the DOSN users for active participation and hamper unfair. The related work with respect to privacy and security is presented in section 2.2.5.

Other than the above five challenges, there exist many other technical challenges which need to be addressed. In this thesis, we are not presenting the related works with respect to those areas as they do not fall under the premise of our research. Topology,

searching for new friends in the network, openness to new applications and locality related issues are few challenges which cannot be ignored [25].

2.1 Distributed Systems

Computing has made its presence felt in every walk of human life. Most of today's applications such as mobile or banking services are based on distributed computing paradigm. According to Wikipedia a distributed system is a *collection of multiple autonomous computers that communicate through a computer network* [26]. A well-known distributed system is the World Wide Web, in which everything looks like a collection of documents (web pages). When a distributed system is required to scale to a size comparable to the Internet then it is said to be a massively distributed system.

As a representative of massively distributed systems, peer-to-peer systems have been very successful (e.g. Napster [27], Gnutella [28], BitTorrent [29]). Despite the fact that the success mainly comes from the distribution of copyrighted content, peer-to-peer systems have been also intensively studied in academia these years. Many applications built on top of P2P overlay are proposed. However, besides file sharing, only a few applications like Skype [30] for IP telephony, SETI@Home [31] for grid computation, etc. are practically deployed. Such situation is mainly due to system unreliability, which stems from various properties of P2P systems.

2.1.1 Peer-to-peer Systems

2.1.1.1 Peer-to-peer Basics

In peer-to-peer systems, resources are deployed in a distributed fashion and the tasks are performed in a decentralized manner. The benefits of P2P include improved scalability, reduced need of costly infrastructure, resource aggregation, etc. Currently, there is no specific definition of P2P, but we loosely adapt the one from [32]: “*Peer-to-peer*

systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.” The main properties of peer-to-peer system include:

- Decentralization: There is no strong centralized administration.
- Scalability: P2P architecture is easily scalable and with no trouble can handle large number of users and addition of resources to the network.
- Dynamics: Every peer is allowed to join and leave the system arbitrarily (also called churn).
- Self-Organization: P2P systems are self-organized without control of any external system.
- Ad-Hoc connection: Due to the openness and underlying network protocols, connections among peers are ad-hoc.
- Servent: Each peer acts both as a client and a server.

We discuss taxonomy of P2P systems in the next subsection to deeply understand P2P systems.

2.1.1.2 Peer-to-peer Classification

Using different criterion, P2P systems can be classified in different ways.

2.1.1.3 Network Centralization

Theoretically, P2P systems are fully distributed, that is, there is no central entity in the system. However, in practice this is not always true. Various degrees of centralization are present.

- Fully Decentralized P2P: All the peers in the system perform the same tasks, acting both as server and client. An example of such a system is Freenet [33].
- Partially Centralized P2P: In these types of systems, some peers (supernodes) are responsible for more tasks than other peers. For instance, maintaining local central indexes for files. Different from a central entity in traditional centralized system, these supernodes prevent a single point of failure since they are selected dynamically. Example of such a system is Kazza [34].
- Hybrid decentralized P2P: In this type of architecture, a central server maintains the metadata for identifying peers and as well as for looking up of data. By this way the central server facilitates the peers' interactions. That is, when a peer wants to download a file, it requests the server. The server reply contains the addresses of peers that own the requested file. Then direct connection is established between the data requestor and data holder(s). Obviously, these systems suffer from the single point of failure and thus are vulnerable and unscalable. Examples of such P2P systems are Skype and Napster.

2.1.1.4 Network Structure

Peer-to peer systems can be classified as structured or unstructured based on how peers are linked to each other.

- Unstructured: In these type of P2P networks the overlay links are formed arbitrarily and the data is randomly distributed over the peers. The peers initiate a flooding

mechanism to query for a piece of information. Flooding guarantees the retrieval of rare data. However, it also generates high amount of traffic in the network. Gnutella and Kazaa are the examples of unstructured P2P networks.

- **Structured:** In structured systems, a tightly controlled topology and placement of data at specific locations ensure the efficiency and ease in search for a desired file. Structured systems offer a scalable solution for queries in a distributed environment. The disadvantage of the structured systems are that they are difficult to maintain if the nodes are highly transient. Chord [35], Pastry [36], CAN [37], P-Grid [38] are examples of structured peer-to-peer systems.

2.2 Challenges in DOSN

The decentralized architecture in a DOSN invites various challenges. In this section, we present works related to five challenges in a DOSN which are part of our current and as well as immediate future work. Out of the five we have investigated three problems in our thesis, namely - (i) information dissemination, (ii) data availability and (iii) finding top-k nodes (with respect to a particular interest) in a DOSN. The other two problems, which are the focus of our immediate future works are (iv) DOSN economics and (v) privacy and security in a DOSN.

In a DOSN, a user might select top-k nodes to store their data based on the maximum resources the nodes have. Schemes for information dissemination are essentially needed for finding top-k nodes or for spreading the need of assistance from other users for increasing the data availability. A DOSN also needs special incentive or reward mechanisms and trust mechanisms, built-in into the system, so that users are motivated to participate actively and fairly. The decentralized architecture of a DOSN also invites security threats and privacy issues for users from other users.

2.2.1 Information Propagation

Targeted dissemination of information can be carried out efficiently using a structured approach as is the case with publish/subscribe systems [16] that rely on infrastructure like overlay based application layer multicasting [39] or gossiping techniques [17] to spread the information within a well-defined group. Well defined groups are not always practical, and alternative approaches to propagate information in unstructured environments are appealing. Broadcasting information to everyone ensures that relevant people get it (high recall), but is undesirable, since it spams many others who are uninterested (poor precision). In mobile ad-hoc and delay tolerant networking (DTN) scenarios, selective gossiping techniques relying on user profile and context to determine whether to propagate the information, have been used. This is analogous to how multiple diseases can spread in a viral manner - affecting only “susceptible” subset of the population. Autonomous gossiping [40] and SocialCast [41] are few such approaches, which rely on the serendipity of interactions with like minded users, and depends on users’ physical mobility that leads to such interactions. Thus, such approaches as presented in [40], [41] and [42] are not suitable for the settings considered in our work [19], where the graph connecting the nodes can be considered static for all practical purposes during the period of a single dissemination event, and changes very slowly even over multiple such events.

There are also many works studying naturally occurring (not specifically designed) information spread in community networks - including in the blogosphere [43, 44] as well as in online social network communities [45], and simulation based models to understand such dissemination mechanisms [46] and human behavior [47]. Our work, GoDisco++ [19] lies at the crossroads of these works - we specifically design algorithms to effectively (ideally achieving high recall and precision, low latency) and efficiently (low spam and duplication) disseminate information in collaborative social network communities by using some simple and basic mechanisms motivated by naturally occurring human behaviors

(similar to mouth of word) while relying only on local information. Compared to another work [23] on which this ultimate solution is heavily based, we have also explored the multiple relations which may exist among nodes. Multiple kinds of relations make the dissemination more focused by identifying and targeting the most likely audience.

Broadly speaking, GoDisco++ belongs to the family of epidemic and gossip algorithms [48]. However unlike GoDisco++ where we carry out targeted and selective dissemination, traditional approaches [49, 17, 50, 51] are designed to broadcast information within a well defined group, and typically assume a fully connected graph - so any node can communicate directly with any other node. In GoDisco++, nodes can communicate directly only with other nodes with whom they have a social relation. While this constraint seems restrictive, it actually helps in directing the propagation within a community of users with shared interest, as well as in limiting the spread to uninterested parties. We specifically aim to use only locally available social links, rather than forming semantic clusters (e.g., using mechanisms like T-Man [52]), because we want to leverage on the natural trust and disincentives to deliberately propagate bogus messages that exist among neighbors in an acquaintance/social graph.

Similar to GoDisco++, design of JetStream [51] is also motivated by social behavior - but of a different kind, that of reciprocity. JetStream is designed to enhance the reliability and relative determinism of the information propagation using gossip, but it is not designed for selective dissemination of information to an interest sub-group. A recent work, GO [53], also performs selective dissemination, but in explicit and well defined subgroups, and again by assuming a fully connected underlying graph. The emphasis of GO is to piggyback multiple messages (if possible) in order to reduce the overall bandwidth consumption of the system. Some of the ideas from these works [51, 53] may be applicable as optimizations in GoDisco++.

2.2.1.1 Communities in Distributed Systems

In all the distributed (decentralized or centralized) systems we often observe community phenomena. In OSN, which is an example of centralized systems, the community phenomena is very evident and explicit, for example groups in *Facebook*. In some other distributed networks like network of emails, there is no clear and explicit community. However, a group of emails ids of users working in a particular interest (for example database) can be termed as a community (which is not explicit defined by the network). We often observe that users working in the field of a particular interest generally know (mostly if not all) researchers working in that field and thus the communication among them establishes an implicit concept of community (with respect to that interest) in the network. Our proposed algorithm explores the concept of virtual communities which exist in the network for message propagation. Since the concept of community is a key idea in our proposed algorithms, we present various works from different social (semantic) networks where the concept of virtual community exists and is thus relevant to our work.

In multiplayer computer gaming the concept of virtual communities is a relatively new phenomenon and is growing rapidly. Various works have analyzed the communities in the past which formed in the massively multiplayer online game (MMOG). By using self-organizing maps Lia C. Rodrigues [54] tries to identify clusters from MMOG. To improve the performance of the algorithm, authors further designed a fuzzy system. However, the main disadvantage of the approach is that it requires training for every MMOG. Activities are one of the important triggering factor for the formation of communities. Pang and Chen [55] try to identify the community structure of the relationship network formed in one activity. In a relationship network, generally, a person might belong to different communities. The presence of a node in multiple communities leads to the formation of a shorter path in the network. In a different study [56] authors study the social interactions within the virtual communities. Considering world wide web as an

online community, they analyzed as to what extent characteristics of a tribalism exist in such communities.

2.2.2 Data Availability in DOSN

Work on peer-to-peer storage systems date back to the OceanStore [57] initiative to achieve archival storage using end-user resources. More than a decade of P2P storage related research later, we have several hybrid or peer-to-peer storage and backup cloud like services in actual deployment, e.g., Wuala¹ [58], where storage task is centrally coordinated, or as academic prototypes such as FriendStore [59], where storage is carried out at nodes' friends. While allowing for options for sharing and socializing, the original design intent of such systems is not social networking. At a very high level, other instances of hybrid (central coordination assisted) peer-to-peer virtual community networks include internet telephony service like Skype [30], P2P massively multiplayer online games and virtual worlds [60], and social peer-to-peer file sharing systems like Tribler [61]. The need and challenges of realizing DOSNs were formalized recently [13], and since then, there has been a flurry of academic initiatives to realize both complete systems, as well as to surmount individual challenges. A more exhaustive survey of existing DOSN specific research covering various aspects of DOSN designs can be found in [25].

Some of the prominent (and architecturally representative) prototypes of DOSN include PeerSoN [4], SoJa [62], SafeBook [6] and Vis-a-Vis [5]. The PeerSoN design, in principle, assumes a global storage service realized using user resources based on a DHT (Distributed Hash Table), but storage location and responsibility are disentangled from social relations. Data security is the primary focus of Safebook [6], where the degree of trust among nodes is used to create a multi-ring system called Matryoshkas. In Vis-a-Vis [5], each individual user host its data on a virtual individual server (Vis), which is run on a cloud service.

¹The first version of Wuala had a hybrid architecture, but now it has a centralized one.

SoJa is a DOSN based social library application, which identifies a dichotomy of storage services - namely global and ‘socially’ local storage, where the global storage is again realized using a user resource contributed global DHT, and stores interesting information such as user directory, as well as allows storage on encrypted offline messages for users (similar to as in PeerSoN), while nodes store data which will typically interest only friends in the local storage realized using friend nodes’ resources.

Social relation agnostic storage approaches like PeerSoN do not allow for incentive mechanisms for users to contribute resources. Approaches leveraging on only immediate social and trust relations (as in SoJa, FriendStore, SafeBook), or based on reciprocity [63] provide incentive mechanisms with limited scope, since users benefit only within the confines of their individual social neighborhoods, rather than at the level of a community at large, and such limitations lead to poor level of availability [64, 63]. In Vis-a-Vis, each peer needs to provision for its storage needs, possibly by making monetary payment to a storage cloud service. In SuperNova, we try to leverage on the advantages of each of these approaches, by allowing a multitude of storage and strategy choices - and by adapting an architecture which allows both bilateral as well as community wide incentive mechanisms. The super-peers in SuperNova help realize a peer provisioned storage cloud, by both providing storage themselves, as well as by helping in matching the nodes to meet storage needs. In turn, the super-peers themselves may, similar to Vis-a-Vis, be run either on commercial cloud services, or by running (relatively) dedicated personal computers by the end users. Users can store their data at friends, super-peers or at ‘stranger nodes’, the later matching, coordination and accounting being carried out by super-peers. Likewise nodes may choose to allow friends or strangers store data on them, in order to gain reputation and influence in the community, which can culminate into super-peer status, which in turn allows opportunities to monetize the service. Other extrinsic incentives or reasons for nodes to run super-peers may be altruistic, or vested interests such as

corporate entities running super-peers to manage their employee ‘accounts’. Thus, what we achieve with the SuperNova architecture is in some sense a generic hybridization of the existing architectures, allowing the users to participate in an ecosystem, where each individual user has the flexibility to choose different trade-off points.

Another recent approach looks at using user resources to help scale centralized OSNs [65] by using user resources to distribute large files among friends, but in such a scenario, the users’ role is more that of a peer-to-peer content distribution network, and the privacy and autonomy issues in centralized OSNs persist.

Besides the fundamental architectural and design differences with existing DOSNs, the presented work has several other differences with respect to the existing literature. We provide explicit mechanisms for new nodes to join the system, and leverage on the resources provided by super-peers and strangers, while they wait to establish or discover adequate friendship relations. The super-peers based architecture was motivated by our analysis of real world online-offline behavior of an egocentric network [20]. To the best of our knowledge, this is the first work of its kind which analyze the real traces for evaluating the storage schemes based on friendship network. Recently, in another work [66], the authors study the feasibility of DOSN using real traces of *Facebook* and *Twitter* to study the trade-offs between data availability (user online time) and replication degree.

In SuperNova, we carry out experiment based evaluation of different parameter choices to determine the impact of different mix of user strategies on the sustainability of DOSNs, which contrasts with the evaluation of existing prototypes, which mainly focus on functional correctness in controlled settings, rather than exploration of the design parameters and system environment (which we do).

2.2.3 Ranking in Distributed Systems

The algorithms that have been proposed in the past such as [22], [67] and [68] assume that all the information is present on a central server. The crux of these algorithms is

to compare each pair of nodes and provide the result after analyzing the whole graph. All these algorithms are computationally expensive since it needs to calculate similarities between every pair of objects. These studies have been done on networks from a global perspective - that is when a decision is taken based on the whole knowledge of the network. However, with the amount of information growing per day, it is not easy to keep all the information at one place. In contrast, our algorithm takes a decentralized approach and uses only local information which is available with the nodes.

Although the classic pagerank algorithm [22] is centralized in nature, later a distributed page rank algorithm was proposed by Brinkmeier [69] by splitting the graph into its strongly connected components. Based on similar lines Shi *et al.* proposed distributed page ranking approach for structured P2P networks [70]. The approach works by dividing the whole network into different groups and then each group's ranking is calculated by a ranker of the group. Later, these rankers collaborate for final conclusion. They also added a mechanism to reduce overhead communication between the rankers. In both the algorithms, authors try to solve the problem by breaking the problem into small subproblems. Even though these algorithms are distributed in nature, they still cannot be applied in decentralized settings.

In the past, extensive and thorough research has been done with respect to searching for top-k items on the web like the seminal PageRank algorithm [22] or like that of finding top-k entities in a P2P [71]. In [22], information is collected about all the web pages by a central system, and then based on indegree and outdegree, the ranking of the pages is done. In [71] super-peers collect information of local ranking from individual peers. Based on the collected information, rankers calculate global ranking of the entities. The approach explored the peer ranks to optimize routing. This leads to shortening of answer set size as well as lowering of network traffic. Our work is totally different from these as instead of ranking web pages or peers (based on resources) in centralized or hybrid

settings, we propose an algorithm to rank users based on the relevancy score in a pure decentralized network.

Pei and Wang proposed TPUT algorithm [72] for distributed algorithm to find top-k objects in a distributed network. There is a central manager concept (similar to a central server) which is used to start the process of ranking. After collection of all the information, pruning is done to filter out ineligible objects as much as possible to avoid unnecessary calculations. In this work, there is an assumption of central manager/sever which has all the information and can take a decision based on lower bound value. However, in our case, central entity is missing. Thus all nodes have to cooperate with each other to get the information about top-k users in the network.

One of the very close related works with that of ours is that of Jelasiy *et al.* [73] where authors proposed a gossip based aggregation of maximal values. Bai *et al.* [74] presented a gossip-based technique for top-k query processing namely P3Q. The key idea is that the queries are gossiped among users with similar tagging behavior and the process is performed iteratively to get refined results. Compared to gossip based approaches which assume that a node has a knowledge of some n number of random nodes, in our case a node can only communicate with its neighbors. Thus, the setting of social constraint in our case is more realistic.

We have summarized the related work with respect to three primitives being investigated in this thesis in Tables 2.1, 2.2, and 2.3.

2.2.4 DOSN Economics

Distributed systems are designed for a large number of users, usually numbering in millions. To cater to such number of users many distributed systems employ scalable decentralized peer-to-peer architecture. Such distributed systems, which operate on public networks are often susceptible to a variety of potentially malicious behavior [75], [76] from

free-riding [77] to Denial of Service (DoS) attacks [78]. Thus, in distributed and decentralized systems, trust and incentive based algorithms are essentially required. Studies such as [79] and [80] on Gnutella reveals that approximately more than two-third of the peers don't contribute in terms of file contribution and thus most of the peers in the system are free-riders. A well-designed system might fail if there are no economic considerations such as incentives for the users to behave well and punishments for bad behavior.

In this section, we present related works for problems like free riding and trust. DOSN is very susceptible to these problems as there is no central authority which will strictly enforce the fairness among the nodes in the system. Also, a DOSN must employ either reputation based trust mechanism or provide incentives as a source of motivation to the nodes to act fairly. Grandison and Sloman defined trust as [81] – *“The firm belief in the competence of an entity to act dependably, securely and reliably within a specified context”*. According to Abdul-Rahman and Hailes – *“A reputation is an expectation about an agent's behavior based on information about or observations of its past behavior”* [82]. From the definition, we observe that reputation can be derived from either trustor's past experience with target agent (direct trust) or opinions reported by third parties (indirect trust). In incentive based approach, a node gets more incentives by not acting maliciously and thus provides right information (or resources) in a distributed system. It is important to distinguish selfish nodes in the network which can reduce the quality of service of the over all system.

In P2P based file sharing community, reputation or incentive based mechanisms are strongly required to force the users to act fairly. Shneidman *et al.* did a faithfulness analysis for existing algorithms running on the Internet (BitTorrent) in [83] and used manual backtracing to find rational manipulation problems. To discourage free-riding, Tamilmani *et al.* presented a credit based incentive approach [84]. To capture the repeated

interactions in systems like BitTorrent, recently, Rahman *et al.* [85] used a simulation based approach (which they call Design Space Analysis -DSA) for analysis of incentives in distributed protocols. In the context of P2P multicast streaming services in [86] Ngan *et.al.*, proposed a simple technique to distinguish between fair and unfair nodes. A node makes its judgment about selfish nodes by comparing the upstream peers' behavior. For peer-to-peer media streaming systems in [87] Habib and Chuang proposed a mechanism for rank-based peer-selection and provide incentives for cooperation through service differentiation. In P2P Massively Multiplayer Online Games (MMOGs), cheating by players is highly possible because of absence of (i) centralized control and (ii) specialized trust management mechanisms. To identify trustworthy clients, a distributed reputation management system called as REPS was presented in [88]. The approach included mutual rating among users. Additional reputation query mechanism helps in knowing the trustworthiness of other users.

Various models have been proposed to study and understand trust in different distributed systems mainly in P2P networks where the problem is quite evident. A study of rational peers and their interactions was done by Chiranjeeb *et al.* in [89]. Along with presenting a game theoretic model, to improve the system, authors also proposed a differential service based incentive scheme. In a similar work in [90] Alberto *et al.* proposed the interactions between nodes as a random matching game, and described a simple reputation system that can provide incentives for good behavior. In a large distributed network there is no central authority to verify and validate the trustworthiness of agents. Based on mathematical theory of evidence, Yu and Singh proposed a mechanism for propagation of the ratings in a large networks [91]. An agent can then use the ratings from other agents (being propagated in the network) to combine with its own knowledge (experiences in the past) to verify the trustworthiness of other agents.

A fully distributed method for selection of the most trusted agents was proposed by Kaszuba *etal.*, in [92]. They also proposed various improvements in terms of security

of the existing ones. A Bayesian based distributed and decentralized approach was presented by Buchegger and Boudec in [93] where they presented a distributed reputation mechanism in which nodes share reputation information, which they collect from other nodes. In a decentralized system, sometimes it is not possible to cover all the social links existing in the network for aggregating reputation feedbacks. To overcome this hindrance, alternatively, a hybrid reputation system has been proposed [94].

One of the works which can fit well to our SuperNova work is that of [95], where authors proposed a super-peers based hybrid model. The framework also has a secure mechanism for the aggregation of trust reports. Another closely related work is that of [96], where Yao *et al.*, proposed super-agent based reputation mechanism in decentralized systems. Super-agents with more capabilities maintain reputation information of services and share the information with other consumer agents in the system. They also introduced a practical reward mechanism for super-agents to contribute their resources and provide truthful reputation information.

2.2.5 Privacy and security in DOSN

User's privacy and security is an important issue in OSN which has been well investigated in various studies [97], [98], [99]. Applications like flybypnight [100] and NOYB [101] which have been proposed (for *Facebook*) to make the privacy more robust, doesn't free the users from the jaws of *Facebook*. The idea in [101] involves replacement of users' data by atoms of other users' data. The technique is not secret-key share mechanism free as it requires the sharing of secrets with friends. However, this doesn't guarantees the non-exposition of relations which can be inferred through message exchange. The approach in [100] trusts *Facebook*, which acts as a middleman between users and the application server for providing encryption. The solution can easily be exploited by *Facebook*. It is difficult to achieve full proof privacy in OSN with a third party applications under the

shadow of centralized authority. Thus, decentralization of the architecture for OSN is the call of the time.

The decentralized architecture also invites issues related to privacy and security from other users. In a very recent work [102], Greschbach and Buchegger studied the friend adversary model. They examine how a friend as an attacker might exploit the social link status to target a neighbor. We present in this section the related literature with respect to privacy and security in a DOSN. Many decentralized architectures have been proposed such as Safebook [6], Vis-a-Vis [5], PeerSon [4], Persona [103], Lockr [104] and including our own [21], as an alternative to centralized OSN to eradicate privacy issues.

Safebook [6] offers privacy, integrity, and availability by using community based trust mechanism. The degree of trust among nodes is the motivation to create a multi-ring system called Matryoshkas. The presence of Trusted Identity Service (TIS) entity makes the architecture secure from DoS attacks. While Safebook exploits the real world social relations to provide a secure DOSN architecture, in a social agnostic approach in Vis-a-Vis [5], each individual user host its data on a virtual individual server (Vis), which runs on a cloud service. However, this architecture invites privacy issues from cloud infrastructure.

Various solutions have been proposed where researchers exploit peer-to-peer techniques for providing a decentralized architecture and use encryption to control privacy issues. To address the privacy problem, PeerSon [4] uses an OpenDHT as a means to build a peer-to-peer system to achieve decentralization. On top of that, it uses encryption for access control. It's a hybrid solution and is less secure as compared to Safebook [6]. On similar lines, another peer-to-peer approach was proposed in [105]. An advanced version of Kademlia [106] based DHT framework, embedded with identity management was used for achieving OSN type functionalities in decentralized settings.

Some works have highly emphasized on providing a secure environment by using various encryption techniques. In Persona [103], authors proposed attribute-based encryption

technique for creating policies for accessing the data by other users. The scheme requires a generation of a key-pair value and distribution of the public key out of band to other users. This might hamper the effectiveness of the mechanism. Like Persona, Lockr [104] is also an access control system and for transferring privacy-protecting structures it uses XML-based formats. Lockr assumes that relationships are bidirectional which sometimes are not the case in real-life. Also, it may happen that because of monetary reasons, a person might collude with malicious attackers [102].

As compared to the above approaches, totally different ideas have been presented by researchers – augmenting the decentralized framework with the addition of the new communication layer. The layer is accomplished by the introduction of a new programming language which puts extra burden on users. Homeviews [107] proposed a P2P middleware solution using a SQL-like query language for sharing personal data. Access for clients is managed through procedures which might not be easy to carry by clients. Like Homeviews, PrPl [108] also uses a SQL type language for accessing the data. However, the use of automatic account creation and the use of OpenID management systems reduces the unwanted overhead in PrPl.

Table 2.1: Past literature work with respect to information dissemination.

Work	Approach/ Methodology	Importance	Shortcomings
Pub/Sub[16], Bimodal[17]	A structured approach. Based on publish subscribe and use, gossip technique.	High recall and precision.	Not practical - assumes overlays. Topic classification (micro/macro) can affect performance.
Autonomous Gossiping [40], SocialCast [41], Bubble rap[42]	Mobile ad-hoc and DTN, gossip and epidemic based techniques.	Works well in mobile ad-hoc and DTN.	Effectiveness of the approach depends on - (1) serendipity of interactions with like minded users. (2) users' physical mobility
Jetstream [51]	Gossip based technique designed to enhance the reliability.	Socially inspired solution - exploits reciprocity for message dissemination.	Not designed for selective dissemination of information to an interesting sub-group.
GO [53]	A gossip based technique, performs selective dissemination.	Piggyback multiple messages to reduce the overall bandwidth consumption.	Non-realistic assumptions - 1) explicit and well defined subgroups and 2) fully connected underlying graph.
Blogosphere [43, 44] Flickr [45]	Study naturally occurring information spread in community networks.	Gives insight about information dissemination from real dataset.	As dataset is from centralized systems so not necessarily reflects for DOSN.

Table 2.2: Past literature work with respect to data availability.

Work	Approach/ Methodology	Importance	Shortcomings
Oceanstore [57]	Archival storage using end-user resources by using P2P technique.	First initiative for a scalable non-centralized system.	Untrusted entities and assumes an overlay connection.
Wuala [58]	Centralized coordinated storage system.	Reliability is guaranteed due to centralize architecture.	Privacy concerns because of centralized system.
SoJa [62], FriendStore [59]	Friends' based social library application.	Durable as uses both local and global techniques for achieving data availability.	As friends are collocated so cannot guarantee 24X7.
Peerson [4]	Achieves decentralization by exploiting external P2P (open DHT) system.	Open DHT provides reliability and ensures access control through encryption.	Leverages on an untrusted P2P system.
Safebook [6]	Data security is the primary focus of this architecture.	Proposes a social (and trust) based security solution for a DOSN.	Assumes friends/relatives are trusted entities.
Vis-a-vis [5]	Cloud based data storage mechanism.	Reliable as data stored in cloud.	Clouds are not free from privacy issues.

Table 2.3: Past literature work with respect to ranking.

Work	Approach/ Methodology	Importance	Shortcomings
PageRank [22], SimRank [67], P-rank [68]	Rank mechanisms for huge datasets.	Efficient for scenarios where data (even if distributed across the globe) controlled by one authority.	Requires beforehand all the information. Expensive computations. Not suitable for DOSN.
Distributed page rank [69]	Distributed version of page rank mechanism.	Hybrid mechanism. Fault tolerant because of multiple rankers.	Exploit connected components which might not be feasible in all networks especially in DOSN.
Ranking in P2P [70, 71]	Structured P2P [70] and super-peers [71] based. Distributed approaches.	Fast mechanisms as based on divide and conquer.	Architecturally dependent approaches, thus, not suitable for DOSN.
TPUT [72]	Hybrid approach. Efficient compared to centralized approaches.	Pruning to filter out ineligible objects to avoid unnecessary calculations.	Not suitable for DOSN as the central manager does not exist in DOSN.
Gossip approaches [73, 74]	Gossip based technique for calculating top-k information.	An efficient and faster mechanism for calculating top-k nodes.	Assumes freedom of communication between nodes. Thus, not realistic for DOSN scenario.

Part II

Challenges in DOSN

Chapter 3

Information Dissemination

This chapter investigates the first problem out of the three that we identified in a DOSN - information dissemination in decentralized settings. We propose GoDisco++, a gossip based approach for information dissemination in online social community networks.

3.1 Introduction

Interest communities are present in many networked and distributed environments. In online social networks there is often an explicit concept of communities or groups based on declared interests of individuals, as well as implicit communities that can be inferred from the link structures. Likewise, in other networks, such as those derived from communication patterns, or in traditional file sharing peer-to-peer systems where people with similar interests self-organize into a semantically clustered overlay [109, 110], implicit communities may be identified. Recently, various efforts to build peer-to-peer online social networks (P2P OSNs) and applications have also been underway [4, 103, 111, 21]. Such systems are being investigated to achieve privacy and autonomy from big brotherly service providers [25]. Likewise, in virtual worlds such as massively multiplayer online games (MMOGs), virtual communities are formed.

3.1.1 High level objective

It is often necessary to have a mechanism to disseminate information in such virtual communities. Since the users who form a community (be it explicit or implicit) are driven by shared interests, a dissemination mechanism should take into consideration that ideally only users who might be interested in a particular message should receive it. Essentially, such a selective dissemination mechanism needs to

- maximize spreading (high recall): The message should reach out to as many as possible relevant users who would be interested in the information.
- minimize spamming (high precision): The message should reach out to as few as possible non-relevant nodes.

As compared to centralized settings, this is non-trivial to achieve in decentralized settings due to lack of global knowledge or coordination, as well as the system's dynamics, such as churn or change of user interests. Thus it is desirable to design mechanisms that utilize only information that is locally available, and hence, easy to update as well.

3.1.2 Motivation

Circulation of call for papers (CFP) among academic peers is a good analogy to understand the motivation behind our work. For carrying out such circulation, one may use mailing lists, which is analogous to a very simplistic 'topic based' publish/subscribe (pub/sub) system. If the mailing lists' scopes are too narrow, a user would however have to subscribe to many such mailing lists. A user might miss out some information if he fails to subscribe to any relevant mailing list. On the other hand, if there are few mailing lists, each covering a vast amount of topics, a user might receive unwanted information most of the times.

We often observe colleagues and collaborators also adopt ad-hoc, word-of-mouth ways to propagate CFPs by personal email. However the decisions as to whom to and whom not to forward a CFP is taken by individuals based on their perception of what their buddies' interests may be. Likewise, as a receiver, each individual may build local (implicit) trust metrics to decide which friends typically forward useful or useless information, providing a personal, subjective context to determine the quality of information. However, this unstructured word of mouth (gossip/epidemic) approach does not guarantee full coverage, and can also generate many duplicates. At the same time, such redundancy can however also make the dissemination process robust against isolated communication failures. Such resilience of gossip/epidemic algorithms is well studied and exploited [112, 50, 17, 52] in general.

Despite the well recognized role of word of mouth mechanisms in selective information dissemination in real life as well as over the internet (such as by emails or on the blogosphere and social media), to the best of our knowledge, there has not been any algorithmic (designed) information dissemination mechanism leveraging on the community structures and semantics available in social information systems. This motivated us to propose GoDisco [23] in the past which can be applied in systems such as P2P OSNs, [4] virtual communities [111] and MMOGs [113] where the mechanism can be naturally explored. We refine the previous ideas to propose GoDisco++.

3.1.3 Difference between GoDisco and GoDisco++

In GoDisco [23] we proposed and investigated a gossip based, social principles and behavior inspired decentralized mechanism to disseminate information in a distributed setting, using exclusively social links and exploiting semantic context to keep the dissemination process selective to relevant nodes by exploring social aspects such as degree and activeness of the users along with behavioral aspects such as history as a forwarder. This

work is primarily a refinement of our previous work [23]. The novelty of this extension is the exploration of a multi-dimensional social network (which we borrowed from our another work [24]), whose semantics can be exploited to achieve better dissemination characteristics as compared to what is achieved using a single dimensional social network.

3.1.4 Contribution

Our main contribution in this work (some shared with GoDisco [23]) is the information dissemination algorithm (we call it GoDisco++) which does not require global knowledge of the network. The mechanism explores social principles such as social triad, activeness, degree and is behavior inspired, for example - exploiting forwarding behavior of the neighbors.

The designed mechanism is expected to be useful for distributed social information systems [21], [4], [114], [115] as well as in other settings such as word of mouth marketing strategies in job market. We evaluate our mechanisms using both synthetic and real datasets of social communities, particularly looking at metrics such as recall, precision, etc. Experiments show that the proposed mechanism is moderately effective and efficient.

The rest of the chapter is organized as follows. We present our approach including various concepts and algorithm in section 7.1.3. Section 4.2.3 evaluates the approach using synthetic and real networks. We conclude in section 7.1.5 by indicating potential future directions.

3.2 Approach

In this section we summarize the system model and relevant concepts and then describe the GoDisco++ algorithm.

3.2.1 System Model

In this work we study the problem of information dissemination in large semantic social networks with the aim of maximizing spread while minimizing spam, determined based on the interest of individuals in the population, subject to the constraint of using only social links to propagate information.

Let $G(N, E)$ be a graph, where N represents nodes (or users) of the network and E represents the set of edges between any two nodes n_i and $n_j \in N$. Edge $e_{ij} \in E$ exists in the network, *iff* node n_i and n_j know each other. We define the neighborhood of node n_i as the set $\aleph_i$ where $n_j \in \aleph_i$ *iff* $e_{ij} \in E$.

In a large network, a node cannot acquire (or maintain) global knowledge of all the nodes. For a specific message, the number and location of the multiple destinations is not known to the source. We design an algorithm leveraging on local information and interactions for directed dissemination of a information to maximize coverage to relevant nodes while avoiding spamming effect.

3.2.2 The intuitions for GoDisco++'s design

We propose a feedback based adaptive selective gossiping algorithm. Its design is inspired by several social science principles. Reciprocity [51] mechanism is incorporated to encourage nodes to disseminate information. Social triads [116] are exploited to avoid duplication among nodes. The essential idea is to avoid forwarding a message to the common neighbors of the node from which a node gets the message itself. A neighbor's history as a forwarder for specific kind of message and its neighborhood (and degree) can evolve with respect to time. This information is used as feedback derived locally by nodes for neighbor selection for new dissemination based on experience from previous dissemination. The users also explore relations with respect to a neighbor for identifying most suitable information agents for spreading the information efficiently. Our approach

uses two logically separate and independent mechanisms, namely the control and the message propagation phases.

- (i) Information gathering phase: This phase runs in the background to collect neighbor nodes' information (interest, degree) and behavior (forwarding behavior, activeness).
- (ii) Message propagation phase: This is the module to actually disseminate an information.

Information collected in the 'information gathering' phase acts as feedback which is subsequently used to choose appropriate nodes for disseminating information in future, and thus improving the quality of information propagation.

3.2.3 Concepts

Before explaining the algorithm in section 3.2.4, we first introduce the concept of communities and information agents which are integral part of the proposed solution.

3.2.3.1 Community and Information Agent

Let I be the set of interests, representing a collection of all the different interests of all the nodes in a network. We assume each node has at least one interest, and I_{n_j} represents the collection of all the different interests of node n_j .

A community is collection of people with shared interest who are connected to each other, e.g., researchers working in the area of distributed systems connected to each other by co-authorships, forming a subgraph $G'(N', E')$ where $N' \subseteq N$ and $E' \subseteq E$ such that G' is a connected graph and all members of the community share some common interest, i.e., $\exists x \in I; \forall n_j \in N', x \in I_{n_j}$. According to this definition, a node can and often does belong to different communities at same instance if it has varied interests. The existence

of such communities is supported by various works in the past, for example [117, 109, 110] showed co-authorship follows the small-world phenomena.

Such a definition of community implies that there may be islands of communities disjoint from each other but sharing the same common interest. It then would be necessary to rely on some nodes from outside the communities to pass messages between these communities, but we need to minimize messages to such nodes outside the community.

To that end, we should try to find potential forwarders who can help in forwarding the message to such isolated but relevant nodes. We identify suitable forwarders based on several factors: (i) interest similarity, (ii) node's history as forwarder, (iii) degree of the node, (iv) activeness of the node and (v) by exploring the multiple relations which exist between nodes. These forwarders help spreading a message faster, even while not personally interested in the information - so we call them information agents (IAs), a term inspired from catalytic agents from chemistry literature which are needed for completion of a chemical reaction, even if they don't themselves take part in the reaction.

We elaborate further on interest similarity later in Section 3.2.3.2. High degree nodes form the backbone of many small-world networks, and provide shortcuts between different components of the graph. Next we explain how forwarding history and node's activeness in the system are relevant and used.

History as Forwarder: This parameter calculates how good a forwarder a node is. The rationale behind this approach is the principle of reciprocity from social sciences. If node n_a does a favor for node n_b by forwarding its message, then there is a good chance that n_b returns the favor in future [51]. We provide a motivating example that gives an idea how one can measure it. A user can write a script for the forward button. If the receivers forward the message to his new friends by clicking forward button, the sender can get a notification if the receiver has forwarded the message or not¹.

¹The method needs to take care of privacy concerns as well.

Activeness of a Node: An active node in the network can play a crucial role in quick dissemination of information, making it a good potential information agent. One way to measure user activeness is in terms of frequency of visits to an online social network, which can be inferred from any kind of activity the user does - e.g., post messages or update status, etc. If Δt is the average time difference between two such actions for a node n_i , measured over a window of time or number of activities, then its activeness $A_{n_i} \propto 1/(\Delta t)$

We assumed that the nodes have information about the degree, forwarding history and activeness of their neighbors.

3.2.3.2 Interest classification and node categorizations:

Interests can typically be classified using a hierarchical taxonomy. For simplicity, as well as to avoid too sparse and poorly connected communities, we restrict it to two levels and name them as (i) Main Category (MC) & (ii) Subcategory (SC). For example, web mining and data mining are closer to machine learning rather than communication networks. So data-mining, web mining and machine learning would belong within one main category, while networks related topics will belong to a different main category. Within a main category, similarity of interest among different categories can vary again by different degree, for example web-mining and datamining are relatively similar as compared to machine learning. So at next level of categorization we put data mining and web mining under one sub category (otherwise we would need too many sub-categories or further hierarchy) and machine learning under another subcategory. To summarize, two different main categories are more distant as compared to two different sub categories (within same main category).

A node's relevancy for a particular message depends on the categorization of interest in terms of MC and SC. We determine a node as relevant, possibly relevant or irrelevant

as follows. For example, if a message belongs to subcategory SC_{11} under main category MC_1 , then:

- (i) Relevant Node: is a node having interest in at least SC_{11} .
- (ii) Possibly Relevant Node: is a node having interest in set $MC_1 - SC_{11}$.
- (iii) Irrelevant Node: is a node having not even a single interest falling under MC_1 .

Information Agent Classification: Based on interest similarity, we categorize different levels of IAs with respect to tolerance for spam messages. For a particular message if a message's interest is falling under SC_{11} of MC_1 , we classify levels of IA in the following manner:

Level 1: Nodes having interest in different subcategory under same main category (e.g., a node having interest in $MC_1 - SC_{12}$). Ideally such nodes should not be considered as spammed nodes as they have, high similar interest and possibly they might also be interested in these kind of messages.

Level 2: For nodes having good history as a forwarder, an irrelevant message can be considered as spam. However they have high tolerance for such messages - that is why they have good forwarding history. These nodes will typically be cooperating with others based on the principle of reciprocity.

Level 3: Highly active nodes with no common interest can play an important role in quick dissemination, but their high level of activity does not imply that they would be tolerant to spam, and such information agents should be chosen with lowest priority.

While selecting an IA from non-similar communities maximum nodes should be chosen from level 1 since they are more likely to be able to connect back to other interested users, then from level 2 and lastly from level 3.

3.2.3.3 Relationship Matrix:

Along with exploring node's history as forwarder and activeness and degree, nodes also explore multiple relations that exists between them and their neighbors. Each node maintains a relationship matrix. A node defines what relations exist between itself and each of its neighbors and assign a weight between 0 and 1 representing the strength of the relation. Relations are of different types, some are bidirectional in nature while others are not. We consider following categories of relationships:

Bidirectional and, equally but binary weighted: Some relations are bidirectional in nature. For example colleagues or co-workers are bidirectional and binary in nature. That is, two people are either co-workers (represented by 1) or not (represented by 0).

Bidirectional and, non-equally weighted: Some other types of relations are bidirectional in nature however they are not equally weighted. Relations like friendship fall under this category. For example, from n_i 's perspective n_j could be his best friend however reverse may not be true, for n_j , n_k might be his best friend. So, for friendship relation n_i might give weighted value of 0.95 for n_j , however n_j might assign value of 0.40, representing that he is not confident about his friendship with n_i .

Unidirectional: Some relations are just unidirectional, for example n_i is a patient of doctor n_j .

Some messages are meant only for close and good friends (for example - sharing romantic trip memories to bahamas), however some messages are relevant to be shared with colleagues only (for example - call for papers) and some can be with colleagues and friends as well (for example - birthday party invitation). A life saving message can be forwarded to all contacts as people don't mind forwarding such messages to their contacts for seeking help. Thus relation matrix is used while deciding which nodes to explore as information agents.

3.2.3.4 Duplication avoidance:

We took into consideration proximity among nodes in the social network (social triads) to avoid message duplication. In a social network nodes can typically be aware of all the neighbors of all its neighboring node. This knowledge can be explored in reducing duplication. For example, when a node n_x receives a message from neighbor node n_y then before forwarding to a neighbor n_z , n_x checks if n_z is a neighbor of n_y . If n_y and n_z are mutual neighbors, then the message is not forwarded again to n_z .

3.2.4 GoDisco++

There are two logically-independent modules of GoDisco++ - the information gathering phase which runs in the background to determine system parameters and self-tune the system, while the dissemination phase carrying out the actual dissemination of the messages.

3.2.4.1 1. Information gathering phase:

Each node transmits its interests and its degree information to its immediate neighbors whenever there are any changes, and likewise maintains a relational matrix about its neighbors. In the long term, such changes are expected to be infrequent, and the volume of information exchange per change event to be marginal.

Each node also monitors (infers) its neighbor's activeness and forwarding behavior². The latter two parameters are updated after each dissemination phase. Neighboring nodes who spread/forward further to more number of neighbors are rewarded in future (based on reciprocity principle). Also, nodes that are more active are considered better IA (potential forwarders) as compared to less active nodes. Each node maintains a tuple of $\langle h, d, a \rangle$ for each of its neighbors, reflecting their history as forwarder, degree and activeness respectively.

²We assumed that the information about these three parameters is available with the neighbors.

During the dissemination phase non-relevant nodes are ranked according to a weighted sum $h\alpha + d\beta + a\gamma$ to determine potential information agents (IAs) where α, β, γ are parameters to set the priority of the three variables $\langle h, d, a \rangle$ such that $\alpha + \beta + \gamma = 1$. The parameters α, β, γ can be tuned to control the dissemination process. For example, if speed of dissemination is not critical, α can be set to a relatively higher value and γ to a lower one, as may be the case for disseminating a call for paper for a conference 2-3 months in advance. Higher α gives precedence to the history of forwarding, which adjusts to the relevance of the topics being disseminated - and thus, such a choice keeps spamming relatively lower. While user activity level is independent of the specific message in question. Higher γ is thus expected to accelerate the dissemination process at the expense of relatively higher chance of spamming some nodes and may be applicable, for example for the same call for paper dissemination example, a relatively urgent message like an extension of deadline at the last moment. In a totally different scenario, for inviting researchers (having particular interest for example, database) and their known (relatives/friends) for a conference dinner a user would like to use a high value of β (with a message life of one hop for the random walk messages.).

3.2.4.2 2. Dissemination phase:

We assume that the originator of the message provides necessary meta-information - a vector of interest categories that the message fits (*IntTags*), a vector of relations (*RelTags*) which can be explored for dissemination of information as well as dissemination *parameters* tuple $\langle \alpha, \beta, \gamma \rangle$ (their use is explained below). The message payload is thus: $\langle \text{message}, \text{IntTags}, \text{RelTags}, \text{parameters} \rangle$.

Algorithm 1 (please see Protocol 1) illustrates the logic behind sending message to relevant nodes (by exploring *IntTags*) and collecting non-relevant nodes if they match the relational tags *RelTags*, ranking their suitability and using them as information agents

Protocol 1 GoDisco++: Actions of node n_i (which has relevant profile, i.e., $msgprofile \cap I_{n_i} \neq \emptyset$) with neighbors $\aleph_i$ upon receiving message payload $\langle message, IntTags, RelTags, parameters \rangle$ from node n_k .

```

1: for  $\forall n_j$  s.t.  $n_j \in \aleph_i \wedge n_j \notin \aleph_k$  do
2:   if  $IntTags \cap I_{n_j} \neq \emptyset$  then
3:      $DelvMsgTo(n_j)$ ; {Forward message to neighbors with some matching interest
       in the message}
4:   else
5:     if  $MC(msgprofile) \cap MC(I_{n_j}) \neq \emptyset$  then
6:       With  $p_1$  probability  $DelvMsgTo(n_j)$ ; {Message is forwarded with probability
        $p_1$  to Level 1 IAs.}
7:     else
8:       if  $IntTags \cap I_{n_j} = \emptyset$  and  $RelTags \cap I_{n_j} \neq \emptyset$  then
9:          $NonRelv \leftarrow n_j$ ; {Append to a list of nodes if the node's with no (apparent)
          common interest however node's relation is quite important w.r.t the
          message.}
10:      end if
11:    end if
12:  end if
13: end for
14: if  $|NonRelv| \geq 1$  then
15:   Sort  $NonRelv$  in descending order of their score using  $h\alpha + d\beta + a\gamma$ ;  $\{ \langle h, d, a \rangle$ 
    obtained from information gathering phase,  $\langle \alpha, \beta, \gamma \rangle$  from parameters set by mes-
    sage originator.}
16:    $IANodes :=$  Top  $X$  % of the sorted nodes;
17:    $DelvMsgTo(IANodes)$ ; {Carry out information gathering phase tasks in the back-
    ground}
18: end if

```

(IAs) for disseminating a message based on multiple criterion. We have adopted an optimistic approach for forwarding a message to neighboring nodes. A node forwards a message to all its neighbors with at least some matching interest in the message. For example, a node with interests in data mining and peer-to-peer systems will get a message intended for people with interests in data mining or web-mining.

Of the nodes which are not directly relevant, we determine nodes with common interest within the main categories from message and users' profiles (using $MC(.)$), helping identify Level 1 IAs, and forward the message to these Level 1 IAs probabilistically with probability p_1 . For level 1 IAs, the message though possibly not directly relevant, is still somewhat relevant, and the "perception of spam" is expected to be weaker.

If a node only forwards to relevant neighbors then dissemination would stop once a node encounters that all its neighbors are non-relevant. However, existence of some isolated islands or even a large community with same interests is possible. To alleviate the same, each relevant node also forwards the message to some X percentage of non-relevant nodes with highest scores $(h\alpha + d\beta + a\gamma)$ among all the non-relevant neighbors. These X percentage of non-relevant nodes can probabilistically send random walkers. We implement such random walkers with preference to neighbors which satisfy the relational tags. The message is forwarded to a node with highest score $(h\alpha + d\beta + a\gamma)$ out of these selected nodes and limit the random walks with limited time-out (in the experiments we chose time-out equal to the network's diameter). If a random walker revisits a specific node, then it is forwarded to a different neighbor than in the past, with a ceiling to the number of times such revisits are forwarded (in experiments, this was set to two).

3.2.5 Complexity of the algorithm

The algorithm 1, presented above is for individual nodes. Each node first scans all the neighbors if they are relevant or non-relevant (line no.1-13). A node n_i takes $|N_i|$ time

units for scanning the nodes. Let $sc = |\mathcal{N}_i|$. In addition to this each node also sorts the non-relevant nodes according to $h\alpha + d\beta + a\gamma$ score. Let S represent the cardinality of non-relevant neighbors (that is $|\text{Non-Relevant neighbors of node } n_i|$). The sorting takes $S\log S$ time units.

Ignoring the time taken for forwarding the message to relevant and non-relevant nodes, the complexity of the algorithm is $O(sc + S\log S)$.

3.3 Evaluation

We evaluate GoDisco++ on both real and synthetically generated social networks. User behavior vis-a-vis the degree of activeness and history of forwarding were assigned uniformly at random from some arbitrary scale (and normalized to a value between 0-1). We next describe the community networks on which our experiments were run.

1. Synthetic graphs

We used preferential attachment based Barabasi graph generator³ to generate a synthetic network of ~ 20000 Nodes and a diameter of 7 to 8 (calculated using ANF [118]). On top of this network we assigned three different types of interest assignments describe following. Also, we presented examples of each type of interest assignment in Figure 3.1 where the number inside the circle represents the interest of the node.

Random cliques based interest assignment: People often share some common interests with their immediate friends and form cliques, but do not form a contiguous community. For example, all soccer fans in a city do not necessarily form one community, instead smaller bunch of buddies pursue the interest together. To emulate such a behavior, we pick random nodes but relatively smaller number of neighbors (between 50-200) and assign these cliques some common interest. Figure 3.1.a shows example of Random Clique. Note the assignment of interest to the nodes. There are 3 cliques with interest 1 and one each for interest labeled 2 and 3.

³<http://www.cs.ucr.edu/~ddreier/barabasi.html>

Associativity based interest assignment: People often form a contiguous sub-graph within the population where all (most) members of the subgraph share some common interest, and form a community. This is particularly true in professional social networks. To emulate such a scenario, we randomly picked a node (center node) and applied a restricted breadth first algorithm covering ~ 1000 nodes and assigned the interest of these nodes to be similar to that of the center node, fading the associativity with distance from the center node. Figure 3.1.b shows example of Associativity based interest assignment. Interest labeled 1,2 and 3 are all joined continually to each of their respective community with exceptions.

Totally random based interest assignment: Interest of the nodes were assigned uniformly at random. Figure 3.1.c shows totally random assignment to the nodes of the same underlying network.

We define five types of relations, namely - friends, colleagues, co-authors, relatives, acquaintance. We create 7 different types of categories from these relations by combination of different relations - C1(friends), C2 (acquaintance), C3 (friends, colleagues), C4 (colleagues), C5 (co-authors), C6 (friends, co-authors), C7 (friends, colleagues, co-authors), C8 (relatives). For each edge, we first assign one out of seven categories uniformly at random and then we assign random weights to each relation in the assigned category.

2. Quasi-Real network - DBLP network: We use the giant connected component of co-authorship graph from DBLP record of papers published in 1169 unique conferences between 2004 to 2008, which comprises of 284528 unique authors. We classified the category of the conferences (e.g., data mining, distributed systems, etc.) to determine the interest profile of the authors. The graph is co-author based graph, so each edge has a co-author relation by default. We intentionally removed some of the co-author edges to break contiguous coauthorship aspect. On top of that, we assigned some more relations (friends, colleagues, co-authors, relatives, acquaintance) randomly to the edges.

3.1.a: Random cliques. Note 5 cliques partitioned by 4 boundary lines (A,B,C,D)

3.1.b: Associativity. Note the nodes labeled 1. All such nodes form one connected component with few exceptions, marked by pointed arrows.

3.1.c: Totally random. Note the random interest assignment to the nodes.

Figure 3.1: Examples of different types of synthetic graphs.

3.3.1 Results

In this section we describe various metrics we observed in our experiments. The default parameter values were $p_1 = 0.5$, $X = 10\%$, $\alpha=0.50$, $\beta=0.30$ and $\gamma=0.20$. Other choices provided similar qualitative results.

1. Message dissemination: Figure 3.2 shows the spread of dissemination over time in the various networks. Based on our past experiences [23], damping scheme is too

restrictive in finding the relevant nodes for random cliques and totally random type of networks. In contrast, non-damping approach is more or less like a flooding which increases the spamming in the network. In comparison to both (damping and non-damping), random-walk is a pragmatic approach for message dissemination in different type of networks and thus here we restrict our experiments to random walk scheme only.

The plots compare the percentage of relevant (R) and irrelevant (IrRelv) nodes receiving the message using GoDisco (G) and GoDisco++ (G++). As a baseline we consider a pure random walk (RW) approach in which we don't explore the relational and other behavioral characteristics of neighbors. The experiments reported here have 90 percentile confidence interval. We observe that the refined algorithm provides better coverage of relevant nodes. This shows the effectiveness of using multiple relations in our algorithm. Exploration of top node from the correct relational context makes the dissemination more focused and thus increases the relevant nodes. This effect is most pronounced in the case of the DBLP graph.

In associativity based synthetic graph, there is no improvement, since the community is tightly coupled so most of the relevant nodes will receive the message (sooner or later). For same reasons we observed that standard deviation is very low. Other than associativity we observe that there is a clear improvement in the number of relevant nodes receiving the message. The percentage of irrelevant nodes receiving the message (in both the schemes) is very low, thus clearly showing the effectiveness of the approach.

In all the different type of network settings the pure random walk (RW) approach shows poor outputs. This confirms that using relational and behavioral characteristics of the neighbors achieves better result for dissemination.

2. Recall and Precision: Recall is the ratio of the number of relevant nodes who get a message to the total number of relevant nodes in the graph. We compare the recall (R) for GoDisco (D) vs GoDisco++ (G++) mechanisms, as shown in Figure 3.3 for all the four networks.

3.2.a: Associativity

3.2.b: Random cliques

3.2.c: Total random

3.2.d: DBLP

Figure 3.2: Message dissemination

In Associativity (Figure 3.3.a), since the community is very tightly coupled so there is not much effect in the recall values with GoDisco++. In other graphs, there is a clear improvement in the number of relevant nodes when using GoDisco++ (R G++) thus the recall values have also increased. With exploration of relational concept in random cliques network model (Figure 3.3.b) recall value reaches 1 (as compare to 0.9 in GoDisco - R G) however it was relatively slower than in the associativity based network. Also in case of totally random network model (Figure 3.3.c) we notice the improvement as the value has reached almost 1 (as compared to GoDisco where the recall is slightly higher then 0.8).

In the case of DBLP network (Figure 3.3.d), the recall value is reasonably good. It's

worth pointing out that the DBLP graph comprises of an order of ~ 0.3 million nodes. Thus the magnitude of total number of nodes in DBLP is much higher as compare to synthetic graphs (actually, around fourteen times more). Also, DBLP is a good example of real life networks being a mix of associative connections among people with common interests, as well as some smaller cliques isolated from the bulk of the community.

Precision is the ratio of number of relevant nodes who get a message to the total number of nodes (relevant plus irrelevant) who get the message. Figure 3.3 also shows the precision (P) value measured in the different networks for both GoDisco (G) and GoDisco++ (G++). We observed that except the associativity based graph which is tightly coupled in all other networks the percentage of relevant nodes have increased and the number of irrelevant nodes receiving the message have decreased. Although the difference is minor, nonetheless it has increased the precision.

In general, in both the schemes, although the percentage of irrelevant nodes receiving the message is less as compare to percentage of relevant nodes, there absolute value is high and keeps on increasing with time and thus, deteriorating the precision.

The tightly coupled community (associativity network) has a good precision value in the first part of simulation however, the value deteriorates in later part of the simulation with the increase in non-relevant nodes. In case of random cliques and totally random networks because the community is not well connected, the precision is not as good as in the associativity based network.

We notice that in the DBLP network, with real semantic information, the precision is in fact relatively better than what is achieved in the synthetic associativity based network. From this we infer that in real networks, the associated semantic information in fact enables a more targeted dissemination. We also observe the usual trade-off with the achieved recall.

3. Parameter exploration: In associativity based network, because of a tightly coupled contiguous community, the quality of dissemination is not so sensitive to the

3.3.a: Associativity

3.3.b: Random cliques

3.3.c: Totally random

3.3.d: DBLP

Figure 3.3: Recall & precision

parameter choice, while in scattered networks like random cliques it is. To evaluate the effect of γ in the network, we perform experiments on the random cliques based network with (i) $\gamma=0.60$, $\beta=0.30$ and $\alpha=0.10$, (ii) $\gamma=0.40$, $\beta=0.30$ and $\alpha=0.30$, (iii) $\gamma=0.00$, $\beta=0.30$ and $\alpha=0.70$, and compare these with the scenarios with the default values of $\alpha=0.50$, $\beta=0.30$ and $\gamma=0.20$.

A greater value of γ puts a greater emphasis to highly active users as potential IAs, who can help improve the speed of dissemination, but at the cost of spamming more uninterested users. Results shown in Figure 3.4.a and 3.4.b confirm this intuition. Figure 3.4.a shows the relevant nodes (R) and irrelevant nodes (IR) receiving a message. Figure

3.4.b compares the recall (R) and precision (P) for the two choices of parameters.

3.4.a: Message dissemination

3.4.b: Precision (P) & Recall(R)

Figure 3.4: α and γ comparison on the random clique based network.

4. Message duplication: Nodes may receive duplicates of the same message. Even if application layer filtering of duplicates can be used, such that end-user does not have to see duplicates and be bothered by them, duplicates still mean higher undesirable traffic in the network. We use social proximity to reduce such duplicates. Figure 3.5 shows for synthetic (Figure 3.5.a, 3.5.b, 3.5.c) and DBLP networks (Figure 3.5.d) respectively the number of duplicates avoided during the dissemination process using GoDisco (G) and GoDisco++ (G++), both for nodes for whom the message is relevant (Relv) as well as for nodes for whom it is irrelevant (IrRelv). Since the message duplication is based on the principle of social triad thus exploration of relational concept has not influenced much. The results show that the use of proximity is an effective mechanism to significantly reduce such unnecessary network traffic.

5. Volume of duplicates: Although we leveraged on social triad principle to avoid duplication, duplication still happens for cases like where a node receives a particular message from two different nodes which are not friends. Thus we measure the volume of duplicates received by individual nodes (x -axis of Figure 3.6 for synthetic and DBLP networks) during the dissemination process. There is not much difference observed between

Figure 3.5: Saved duplication

GoDisco (G) and GoDisco++ (G++), which is expected since our relational exploration did not affect the social triads.

6. Qualitative dichotomy of spamming effect: During the dissemination process, we have used information agents (IAs), who we assumed were not themselves interested in the message being disseminated. Among these IAs, one can still define a dichotomy in terms of nodes who share the same main category (level 1 IAs) and have relatively closer interest (e.g. people interested in data mining receiving information retrieval community related news) and will thus be more tolerant to such message, versus information agents who would find the message totally irrelevant, and are likely to find the messages relatively more annoying.

Our experiments show that a very negligible amount of totally irrelevant nodes are spammed while most IAs involved during the dissemination do share the same main

Figure 3.6: Volume of duplicates

category (MC) as the message. Figures 3.7 show the results for synthetic and DBLP networks. In the figures, MC signifies that these IA nodes share the same main category as the disseminated message while $\sim MC$ signifies that the IA nodes don't share the same main category. We see that the population of nodes in the later category is marginally low, particularly when GoDisco++ (G++) is used as compared to GoDisco. This is due to the fact that as in GoDisco++ we explore multiple relations thus less spam of $\sim MC$ is observed.

Based on a rather rigorous exploration of the design choices, as well as various aspects of the system's performance, we see that GoDisco++ has clearly achieved improvement over GoDisco with its more focused approach. Thus like GoDisco, GoDisco++ is a practical solution for information dissemination in the absence of dedicated publish/subscribe infrastructure, by relying strictly on ego-centric social network and the associated locally available semantic information.

Figure 3.7: Qualitative dichotomy of spamming effect

3.4 Conclusions

We have described and evaluated a selective gossiping based information dissemination mechanism (named GoDisco++), which is constrained by communication among only those nodes that are socially connected to each other, and leverages user's interests and the fact that users with similar interests often form a community - in order to do directed dissemination of information. For various kinds of networks, synthetic as well as real, GoDisco++ demonstrated reasonable to very good dissemination characteristics vis-a-vis recall, precision, dissemination latency, message duplication, etc.

In a DOSN, along with looking for neighbors which can help in effective dissemination, users also look for nodes which are ready to store their data. Replication increases the availability of a node's data. In the next chapter, we present our work related to the

problem of storage and data availability in a DOSN.

Chapter 4

Data Availability in DOSN

In the previous chapter, we introduced our approach for solving the problem of information dissemination in a DOSN. In this chapter, we present our two works directed at solving the problem of data availability in a DOSN. We first present our empirical analysis of real traces of online-offline behavior of an egocentric network (see section 4.1). Motivated by the results from this study, in the second work, we present a super-peers based DOSN architecture, which is presented in section 4.2.

4.1 An empirical study of availability in friend-to-friend storage systems

4.1.1 Introduction

Peer-to-peer (P2P) storage systems have been studied for over a decade, starting with the OceanStore [57] project. The premise of P2P storage is crowdsourcing the storage cloud [119] to other end-users. One of the many design issues in such systems is the choice of peers at which to store data. A specific subclass of P2P storage systems has emerged based on the placement choice being constrained to ‘friends’ of the data owner, for example, FriendStore [59]. The basic characteristics of such friend-to-friend (F2F) storage systems are: (i) real-life social trust is exploited to guarantee a dependable system (*e.g.*, a friend of mine won’t erase my data); (ii) data access is predominantly confined

within a small social neighborhood. These networks, also known as “darknets” when the focus is on security, can also guarantee privacy and resistance to censorship [120]. F2F storage thus constitutes a good building block for diverse applications such as personal backup service and decentralized online social networking.

For personal backup, while data persistence is more critical, data availability is nevertheless desirable. For decentralized online social networking applications such as SafeBook [6], availability is of paramount importance. Thus, a fundamental problem that arises is determining what kind of availability one can achieve in a storage system where data placement for any specific data owner is constrained by the use of only peer nodes run by friends of the data owner.

There are several variations of this basic question that would interest an F2F storage system designer. A baseline is determined when all friends of a node store its data. This is the best in terms of availability that one can achieve subject to the constraint of using friend nodes exclusively. However, there are some obvious variations worth studying. Can the same availability (or any other predetermined threshold of availability) be achieved using only a subset of the node’s friends? How does the law of diminishing returns work in terms of availability, as the number of used friends is increased? If a stipulated number of friends are to be used, what is the best availability that can be achieved? Furthermore, the way to measure availability itself may vary. For a personal backup application, the data owner may care for the data to be available only when it itself is online - for example, with other portable devices. For a decentralized online social networking application, the data owner can serve its own data when it is itself online, but will like the friends to make the data available when it is itself offline. More generally, availability may also be determined based on whether it was available when there was any access request for the data. These various interpretations of availability may depend on the access and application specific characteristics.

The achievable and achieved performance would depend on the (temporal) characteristics of individual nodes' egocentric networks (*i.e.*, the social network consisting of those nodes and their respective immediate friends), the actual data-placement policies determining a subset of friends to store data at, as well as the interpretation of availability itself. This work is a first attempt to formalize these quantitative aspects of F2F storage systems, exploring algorithmic aspects of data placement¹ in (sub-)optimal subset of friends, and exposition of the efficacy of F2F storage systems using trace-driven simulations using real egocentric social network traces capturing additionally node availability traces over time.

The important contributions of this work include (i) defining some key characteristics of an ego-network which influence the achievable availability in an F2F storage system, (ii) observing that identification of a minimal set of friends to achieve the maximum achievable coverage is in fact analogous to the set cover problem, and hence NP-hard, (iii) propose greedy heuristic data placement algorithm, and (iv) evaluation of the heuristics using trace-driven simulation using real ego-network traces and accompanying up/down-time information.

4.1.2 Ego-Centric Network Characterization

In an F2F storage system using exclusively friends, the quality of service an individual node can have, depends solely on its own ego-centric network.² We next define some key characteristics to define a node's ego-centric network based on the uptime/downtime characteristics of nodes.

For a node n , we use F_n to represent its set of friends. We study the system for a period of τ discrete time units (seconds, minutes, or other quanta), *i.e.*, the whole

¹Data placement techniques are different from data caching techniques, where the focus is on quickly retrieval of the data. The scope of our research is limited to availability of the data and not to the speedy retrieval of the data.

²Load distribution at nodes may percolate over multiple social hops. Such load-related issues are ignored in this preliminary work.

period of study can be represented as $T_{tot} = \{t_1, t_2, t_3, \dots, t_\tau\}$. The uptime of a node n is represented with a time trace T_n where $t_i \in T_n$ iff node n is online at time t_i .

In general, a node may want to achieve availability of its data for an arbitrary subset of time $T'_n \subseteq T_{tot}$. For example, if a node n wants 24/7 availability then $T'_n = T_{tot}$, while if it wants availability of its data only for the periods when it is itself offline, then $T'_n = T_{tot} - T_n$.

Let T'_{F_n} represent the collection of time trace of all the friends n_j of node n , *i.e.*, $n_j \in F_n$. For notational conciseness, we use $\oint$ to represent $|F_n|$.

$$T'_{F_n} = \{T'_{F_{n_1}}, T'_{F_{n_2}}, T'_{F_{n_3}}, \dots, T'_{F_{n_\oint}}\} \quad (4.1)$$

4.1.2.1 Achievable Coverage

We define *achievable coverage* AC_n as the fraction of the total time for which a node n is able to get data availability by storing data at all and only its friends, out of the total time for which it is seeking data availability. $AC_n = |T_n^{ac}|/|T'_n|$ where

$$T_n^{ac} = \bigcup_{j=1}^{\oint} T'_{F_{n_j}} - \overline{T'_n} \quad (4.2)$$

4.1.2.2 Degree of Criticality

Out of the periods where coverage is achievable, there are time slots which can be covered using an unique neighbor, *i.e.*, there is only one neighbor which is up for that period, and thus this specific neighbor is critical in providing such coverage. We denote using T_n^{cr} the time slots which are critical.³ We define the *degree of criticality* as $DC_n = |T_n^{cr}|/|T'_n|$. A high degree of criticality would typically imply that more friends are needed to achieve a particular level of availability, and also, the system is more vulnerable to faults. Conversely, low degree of criticality implies more redundancy, which in turn is expected to translate into more flexible choices, and robustness of the resulting system.

³Note that $T_n^{cr} \subseteq T_n^{ac}$.

4.1.3 Data Placement at Friends

Replicating data at all friends trivially allows a node to get maximal achievable coverage. However, this would result in very high costs for storing data in multiple copies, in particular for users with a large number of friends. An obvious question is to find the minimal number of friends that can provide the same maximal coverage. It is straightforward to show that this is in fact an NP-hard problem. This finding motivates us to propose heuristics that strive to optimize the trade-off between the number of copies and the achieved data availability.

4.1.3.1 Maximal Coverage With Minimum Replication

Let us now focus on the problem of reaching the maximal coverage for a node n (*i.e.*, T_n^{ac} in Equation 4.2) while minimizing the number of friends employed. This corresponds to the *Set Cover problem* introduced and proven to be NP-hard by Karp in 1972 [121]: using the notation we used so far, Set Cover can be formalized as follows. Given the family of traces T'_{F_n} as defined in Equation 4.1, find the minimum value k such that a sub-family of traces $C \subseteq T'_{F_n}$ of cardinality k having $\bigcup C = T_n^{ac}$ exists. This observation of NP-hardness motivates us to design a heuristic greedy policy for data placement.

4.1.3.2 Greedy Data Placement Heuristics

Let F_n^{cr} be the set of critical neighbors of node n , *i.e.*, each of them covers at least one critical time slot. To get maximum possible coverage, in our greedy heuristic the node first picks all the friends n_j s.t $n_j \in F_n^{cr}$. If critical nodes are not able to cover all the T'_n time slots for which the node n is looking for coverage, then the node n picks a non-critical node that covers the maximum number of the time slots not yet covered. This process continues until there are no nodes left which cover further uncovered time slots, or there are no more nodes left. The pseudocode of the algorithm is presented in Protocol 2.

Other variations of the greedy approach could entail (i) a cap on the quality of service, *e.g.*, a node may decide to settle for coverage not less than 85% of the maximum achievable coverage; or (ii) a cap on the maximum number of friends to be used, *e.g.*, a node may want to limit the replication factor of its data to five. We note that for such variants, other heuristics – possibly those which do not differentiate among critical and non-critical nodes – may yield better performance⁴.

Protocol 2 GreedyPlacement: Actions of node n_i with neighbors $\aleph_i$ for selecting the neighbors F'_i to replicate its data.

- 1: $F_{n_i}^{Cr} \leftarrow \text{FindCritNbr}(\aleph_i)$; {Node n_i find all the critical neighbors.}
 - 2: $F'_{n_i} = F_{n_i} - F_{n_i}^{Cr}$; { F'_{n_i} denotes the friends of node n_i which are not critical.}
 - 3: $\text{Update}(T_{n_i}^{ac})$; {Update the achievable coverage.}
 - 4: **while** $((F'_{n_i} \neq \emptyset) \wedge (T_{n_i}^{ac} \subset T'_{n_i}))$ **do**
 - 5: $F'_{n_i} \leftarrow \text{TopMostNode}(\text{Sort}(F'_{n_i}))$; {Sort the nodes based on maximum time units it is covering. Pick the node covering the maximum time units, out of leftover time units not being covered.}
 - 6: $\text{Update}(T_{n_i}^{ac})$;
 - 7: **end while**
-

Complexity Analysis of the Algorithm: While finding the critical time units, a node has to check for each time unit whether it is critical or not. Let T represent the cardinality of the total time units for which a node n_i is looking for coverage ($T = |T_{n_i}^{ac}|$). Node n_i has to traverse through all the neighbors $\aleph_i$ for finding the critical friends. Thus, complexity for finding the critical friends is $T\aleph_i$. For the while part, each time node will sort rest of the non-critical friends to find the top most node covering the maximum of rest over time units. This process is repeated till all the neighbors are checked or all the time units are being covered (which ever happens first). In the worst case, the node n_i will sort all the non-critical friends till all the neighbors are not consumed. For the while part, the time complexity in worst case scenario is $(\aleph_i - F_{n_i}^{Cr})^2 \log(\aleph_i - F_{n_i}^{Cr})$. Thus, total time complexity is $T\aleph_i + (\aleph_i - F_{n_i}^{Cr})^2 \log(\aleph_i - F_{n_i}^{Cr})$.

⁴Please note in the proposed approach we assumed there are no data storage constraints for the data placement. Introduction of storage constraint will degrade the performance of the algorithm.

4.1.4 Results

For our experiments, we carry out trace-driven simulations using instant messaging traces⁵. Numerous social network graphs are available; likewise, several traces from P2P and other distributed systems exist, providing up/down-time of nodes. However, we could not find openly available real traces containing both social relation information and the availability trace of the same nodes. We thus obtained such dataset from the operator of an instant messaging (IM) server in Italy.

Our dataset suffers from various shortcomings. First, it does not have adequate geographic diversity which can be exploited to achieve better availability. Furthermore, instant messaging usage pattern may arguably be different from the usage pattern of P2P storage systems or decentralized social networking applications. Our data is also a partial view of the complete IM network (*i.e.*, nodes have some contacts registered on different servers whose uptime information is not available).

Notwithstanding these issues, the dataset allows us to experiment on real data with a trace of both social and temporal node behavior, justifying its usage in this empirical study. In a friend-to-friend storage system, if all friends agree to cooperate, then the performance is likely to be better, since almost all the above mentioned drawbacks also happen to translate into more pessimistic estimates about the system environment. We will continue to look for more appropriate data sets in the future.

4.1.4.1 Experimental Results

The original trace comprised 3,436 nodes. After filtering the nodes that had no neighbors in the same server or were not part of the largest connected component of the social graph, we obtained a set of 848 unique nodes, each having between one and eighteen neighbors. The experiments were run for traces corresponding to a period of two weeks. Nodes'

⁵We wrote our own simulator in java for analyzing the real-traces.

uptime/downtime behavior from the first week was used to drive the data placement algorithm, and the achieved availability in the second week was studied. The rationale for this simple experimental setting is the well-known diurnal patterns [122] of individual users' uptimes.

In the following we summarize the characteristics of the ego-centric networks from our dataset and the system performance obtained using our data placement heuristics.

Ego-centric network characteristics: Figure 4.1 shows the characteristics of the nodes considered in our experiments, as derived from the whole period of two weeks of traces. A reverse CDF (cumulative distribution function) is plotted for the achievable coverage (AC). From this plot, we observe for instance, that roughly 50% nodes can achieve at least 90% of coverage (for the whole period, *i.e.*, if the ideal objective is to obtain 24/7 coverage for the two weeks studied) if they replicate their data at all their respective friends. We likewise plot the reverse CDF graph showing the percentage of the total time where coverage is achieved using critical nodes ($Crit$), that is, only one friend was up during the corresponding amount of time. If this sole friend goes offline for whatever reason, then there are no alternatives for the corresponding time slots. Ideally, it is desirable for nodes to have a low value for this metric. We note that roughly 20% of nodes have more than 90% of their time covered by critical nodes - which highlights both the importance of critical nodes, as well as the potential vulnerabilities of a F2F storage system. However, we also observe that this plot has a relatively steeper slope (for instance, w.r.to AC). This is again a desirable trait, since it means many nodes can get coverage using noncritical nodes for a large amount of time.

Microscopic analysis: In Figure 4.2 we do a microscopic study of the performance of our data placement heuristic by looking at individual, representative ego-centric networks. We show five nodes in this plot, with distinct characteristics, summarized using the tuple $\langle AC, DC, \mathcal{f} \rangle$, *i.e.*, achievable coverage, degree of criticality, and the number

Figure 4.1: Ego-centric network characteristics.

of neighbors. In these plots, we additionally get a glimpse of the marginal benefit of incrementally replicating at more friends.

Nodes with high number of neighbors, and relatively low degree of criticality, for instance, nodes represented with $\langle 1, 0.50, 18 \rangle$, $\langle 0.95, 0.25, 12 \rangle$ and $\langle 0.83, 0.38, 7 \rangle$ enjoy good level of availability.⁶ We note that, within these nodes, those with relatively lower criticality ($\langle 0.95, 0.25, 12 \rangle$) obtain high level of availability with very few replicas, in contrast with a node with a relatively higher degree of criticality ($\langle 1, 0.50, 18 \rangle$). There are however also greater overlaps among the neighbors' uptime, and thus, there is an effect of diminishing returns with additional replication, as the maximum achievable availability is approached. Finally, when the maximum availability is indeed achieved, as indicated with a marker in some of these plots, there can obviously be no further benefit in adding more replicas. This corresponds to significant storage space savings with respect to a naive strategy which would replicate at all friends. For example, for the node with 18 neighbors, nine neighbors are enough to achieve 100% availability. In contrast, the rate of increase in the achieved availability with additional replication is higher for $\langle 0.62, 0.43, 4 \rangle$ and $\langle 0.40, 0.34, 3 \rangle$ which have fewer neighbors, and degree of

⁶Recall that DC needs to be interpreted relative to the value of AC.

Figure 4.2: Microscopic analysis.

criticality is relatively higher. We also show a node with only a single node, which naturally enjoys coverage for a very low period of time.

Macroscopic analysis: While one fundamental question is to minimize the number of replicas needed to achieve the maximum possible availability, an obvious dual to this question is, if a node decides to cap the degree of replication to a specific predetermined number, for instance due to bandwidth and storage budgets, what availability is achievable? The first heuristic algorithm, which necessarily uses the critical nodes first, may not be best suited to achieve maximum availability under a cap on the maximum number of neighbors that can be used. Under such a cap on the number of replicas, a greedy algorithm which does not discriminate between critical and non-critical neighbors is expected to perform best. As a base line for such scenarios with cap on the number of replicas, we study what is achieved when replication is done at randomly chosen neighbors instead.

We report our findings in Figure 4.3, which confirm these intuitions. CF3 and CF6 represent the results obtained using our heuristic which prefers *critical nodes first*, subject to replication cap of three and six respectively. MC3 and MC6 represent the results

Figure 4.3: Macroscopic analysis.

obtained using a greedy algorithm to *maximize coverage* subject to the corresponding caps, while R3 and R6 show the results obtained using *random* placement. NL represents the result obtained with our first heuristic with *no limit* on the degree of replication, hence minimizing the degree of replication subject to reaching the maximum achievable storage. As expected, MC outperforms the other strategies when availability needs to be maximized subject to a cap on the degree of replication. Interestingly, we notice that the randomized algorithm in fact outperforms the original heuristics under such limitation on replication, and achieves only slightly less availability than the greedy approach to maximize coverage. Finally, note that even using at most six friends yields only marginally less availability than the maximum achievable.

Comparison to brute force scheme: We have not compared our approach to the brute force scheme⁷. The main goal of the paper is to find a minimum set of friends/neighbors with maximum availability. Thus, in terms of availability there is no difference when you compare brute force scheme to our greedy approach. In brute force the duplication for some time units will be present, which we are trying to avoid in the greedy approach. Thus we have not compared it to our paper. The difference is

⁷Brute force scheme is when you place the data on all the neighbors.

Figure 4.4: Neighbors used for data placement.

only in terms of number of neighbors being used. In case of brute force, we have to use all the neighbors. In case of greedy approach, not all the neighbors are being used for data placement (in most cases). Figure 4.4 presents what percentage of nodes used what number of neighbors during data placement.

Further discussions: The data set we used was extremely sparse (most online social networks have much larger number of neighbors), which explains the low availability in many of the results. One may also argue that even if people have many more online friends, most of them are unlikely to share their storage spaces, and hence a usable network may also be sparser. In all cases, our experiments expose some interesting facets about F2F storage systems. Firstly, if a node has reasonably large number of friends (say, more than 10) willing to participate in an F2F storage system, then a very good level of availability is achievable, and our heuristic does a good job in determining a minimal subset of friends to achieve maximum coverage, saving unnecessary replication. However, if there is a limit on the allowed degree of replication, and the limit is small, then this heuristic suffers a drastic deterioration because of its preference to critical nodes. In contrast, a greedy strategy which does not discriminate between critical and non-critical nodes works well. More interestingly, even a randomized placement strategy works almost as well. We hypothesize that if there is a larger pool of nodes to choose

from, it is likely that the greedy strategy will perform even better, while the random strategy's performance will deteriorate in comparison, since it will be more likely to pick some unsuitable nodes. Larger choice of nodes will also provide a bigger, and arguably temporally diverse pool to choose from - thus significantly increasing the availability.

4.1.5 Conclusions

We performed an exploratory study on the trade-offs between redundancy, data placement and data availability in F2F storage applications. We have shown that the problem of finding a data placement with maximal availability and minimizing redundancy is NP-hard, which prompted us to design heuristic data placement strategies. We then evaluated these policies through an empirical study based on a social network complemented with availability traces. In our case, some low availability results were due to the low number of connections in our particular trace. However, our results showed that reasonably good availability results could be obtained already by nodes that had around 10 friends. We also found out that – depending on the goal – different data placement heuristics are advisable. For example, if there is a limit on the applicable redundancy, a simple greedy heuristic performs better than placement strategies designed for minimizing replication for maximal coverage.

4.2 SuperNova

4.2.1 Introduction

Online social networks (OSN) such as Facebook are extremely popular, and continues to grow [1], and has grown from the early day use case of being just a tool to keep in touch with friends into something much more pervasive in both our private life and work place, including organization or promotion of group events or orchestration of collaborative tasks. With the increasing number of users, and user centric personal data online, such

networks however raise various privacy concerns. Concerns include inadequate or confusing privacy settings to share information with fellow OSN users (which can be addressed using flexible access control policies supported with simpler user interfaces), as well as more fundamental issues related to data ownership and privacy and autonomy from the social networking service providers themselves, as well as possible vulnerability of such vast repositories of personal data to identity harvesters. A slew of initiatives, originating both from academic research (such as PeerSoN [4], Vis-a-Vis [5], SafeBook [6]) as well as open-source and privacy/free speech centric communities (such as [123], [124], [125]) 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.

4.2.2 Approach

In a DOSN system, to increase availability a particular node n 's data has to be replicated among other nodes which are selected by exploring various social links - friendship and strangers (with help from super-peers). The proposed storage mechanism is to provide maximum possible availability (rather than 24/7). We assume each user has a dedicated PC or terminal from which they login. This work focuses on DOSN overlay which exist on top of P2P network.

In this section we first describe various entities of the system before describing the bootstrap process for a new node. We also describe different strategies that a node might select for increasing its data availability followed by system model.

4.2.2.1 Definitions of Entities

This section describes various entities which are parts of SuperNova architecture. A node (or user) n in a DOSN has a list of friends. Some the friends might be ready to

store friend's data thus acting as storekeepers. We also describe relation between all the entities⁸.

Profile: Each node (or user) in DOSN is represented by a profile which can be considered as a medium of expression for the user. The owner of the profile can upload new pictures or videos, and can update his status message. Each user's data is divided into three parts namely public, protected and private.

- **Public :** Anyone in the network can view this data. A user might want to make his name and profile picture and his location as public so that other users can identify him through public information.
- **Protected :** Only friends can view this information. For example a user wants only his friends to view his pictures.
- **Private :** No one, other than owner of the profile can view this data. For example a user might set his few picture as private during interval of time, instead of deleting them permanently with intention of making them protected (or public) in future.

The type of the data can be changed to another type, at any time, by the user - for example a user can change his public pictures to private anytime. Depending on the permissions set by the owner, public and protected data can have read only or read-write access for any user and friends respectively. For example, friends can read status messages and are able to see videos or pictures uploaded by the owner which is a type of read only access. Commenting on uploaded picture or video or replying to wall are examples of read-write access.

Public and Private Keys: Each user has a public and private keys associated with it. It is basically used for validating users. Each user marks his comment or message with digital signature [126] to avoid any impersonation.

⁸We interchangeably use (1) node and user and (2) friends and neighbors in this work.

Friendlist: Each user's profile is linked with his friend's profile, which user adds either based on acquaintances or on interest similarity. The collection of all the linked profile is termed as friendlist. We assume friendship is a symmetric relationship.

Storekeepers: Storekeepers are list of users who have agreed to keep a replication of another user's data so that when a particular node n is down, then n 's friends can contact storekeepers to access n 's data. In other words, n 's storekeepers give visibility to n , when it is down. We describe the process of selection of storekeepers by node n in section 4.2.2.1 in more detail. Not every friend is willing to act as storekeeper because of prior load or due to personal reasons. Every storekeeper maintains a list of nodes for which they are doing a storekeeping for data synchronization. Unlike friendship, storekeeping is not a symmetric relation. For a particular node n , every friend in his friendlist knows the list of all the storekeepers (described next) for the profile (users). During initial period of time when a node n , doesn't have enough friends it requests one of the super-peers to store his data (see section 4.2.2.1). Thus for new nodes super-peers act as storekeepers.

Figure 4.5 shows the relation between all the above entities, with respect to node n . Each node n 's profile data is divided into three parts namely - Public (Pb), Protected (Pt) and Private (Pv). Each node n has list of friends $F_1, F_2, \dots, F_f$ with which his profile is linked. Each friend F_x ($1 \leq x \leq f$) has information about the list of storekeepers ($SK_1, SK_2, \dots, SK_{sk}$) which are storing node n 's protected data. Each storekeeper SK_y ($1 \leq y \leq sk$) has information about all the other storekeepers which are storing node n 's data. Friends can access Protected and Public data. Public data can be accessed by any node in the network whereas private data is solely for node n itself.

Super-peers: Super-peers are one of the most important entities in the system. Any node can become a super-peer which is ready to provide his services to the system most importantly to new nodes. Super-peers provide storage to the new nodes who

Figure 4.5: Relation between different entities.

don't have enough friends in the network for some initial period. They also maintain and manage different types of services for DOSN by cooperating among themselves for example maintenance of user-list. Different super-peers provide different kind of services. Some of them might be offering high bandwidth others might be offering large storage space and some others both. There might be few others which might be offering extensive up-time but with limited space. Depending on need, a new node can select one of the super-peers. Although we have not put a proper reward mechanism in proposed architecture however this is essential part of the system which will boost the morale of the super-peers and thus giving a fair ranking of super-peers for each service ⁹.

Communities: Groups or communities are formed based on interest. Any user can create a community. The creator of the community will act as moderator for the community. Creator (or owner) of the community is the main responsible person for storing all the data and for settling any dispute if it arises in the community due to any posting or because of any other reason. A creator (and owner) may share his ownership

⁹This is part of our future work.

with other nodes if he wishes so. Depending on rules and regulations set by creator, community members will have different access for the community data. The latest data related with community will be shared by other members of the community as well. However community owner takes care of all data (including old and new). There are two ways to define the latest data either w.r.t. time or storage space. In terms of time, latest data could be of last 1 week. In terms of space it could be latest 2 GB of data.

Synchronization: Apart from physical clock, each node n maintains a logical clock for itself and share it with all the friends and storekeepers. Every friend and storekeeper maintains a separate clock for every user with which it's associated. Also each group owner maintains a logical clock for the community. Every community member maintains a separate clock for every community of which he is of member. Logical clock is used for keeping a synch between a node n and his friends and storekeepers. On the same line it used for keeping a synch between all the community members. It is also used to resolve multiple writes and updates (explained more in section 4.2.2.2 either in community or for friend's comment for a node n).

Services/Information: There are two kinds of services namely - public and protected available in the network. Anyone can avail a public service. List of super-peers (and the clause of their agreement) is an example of public service. Protected services are more restrictive in a sense that only those for which it is meant can avail it. For example only group members can access the information related with a particular group or community.

List of Directories: Following is the list of directories being maintained in the DOSN system.

- **User List :** Every user's id and his public information pointers (the nodes which are storing the public data). This list is maintained by super-peers.

- **Super-peer List:** List of super-peers and their respective services and agreement details. This list is maintained by super-peers themselves.
- **Group (or community) List:** List of communities and pointers for community owner and public content of the community. This list is jointly maintained by community owners and super-peers.

Bootstrap Process A new node n_{nn} joining DOSN, initially relies on super-peers to increase its data availability. Later it may find friends (or strangers) in the DOSN with help of super-peers which can act as storekeepers for it. We explain and classify different phases of a new node n_{nn} from the point it enters the network till the point it settles down in the network.

1. Initial Phase: As soon as new user joins DOSN, user creates a profile. We call this initial period as “Initial Phase” (IP). Time period of this phase could be from few hours to one day. We assume (although not necessary) user will set his username, location and interest as public data, as these are the most used attributes to find a user in the network. A public data is a piece of information through which a user might be interested in advertising about himself to make new acquaintances. As soon as profile is created by the user, an entry is added to userlist which is maintained by super-peers. Since a new node might not get enough friends or acquaintances, so it might contact a super-peer for storing his data. When a new node joins the network, he can view a list of super-peers (which is public information) and the list of services they are providing. A super-peer advertises following list of services for the new nodes.

- **Storage:** How much data they can store. In lieu of offering space to new node, in future super-peer might take new node space for storing its personal data or storing some other node’s data.

- Agreement time: For how long they can store (example they can store it for one month).
- Time availability: What time of the day, they can cover for a node.
- Content type: What type of content is acceptable. A super-peer might not be interested in storing any copyrighted material or pornographic material.
- Advertisement: what kind of advertisements will be shown on user's profile from a super-peer.

A new node selects a super-peer and sends a request for storekeeping. Once an agreement is reached between a super-peer and new node, data is replicated from new node to super-peer. New node n_{nn} now enters Take Care Phase (TCP) where a super-peer takes care n_{nn} 's data. The data at super-peer is stored in encrypted form - thus super-peer can't read or modify the content of data. We explain this security mechanism in section 4.2.2.3.

There are number of incentives for a node to become a super-peer and thus facilitate new nodes joining the system. Super-peers get more insight in the network. It could be a self satisfaction for others, as a contributor in making DOSN a successful system. Revenues are the other sources which might attract users to become super-peer as they will post the advertisement on new joinee's profile. Also users give feedback about super-peers at the end of contract which can affect a super-peer's reputation, thus reputation is another source of incentive for super-peers.

2. Take Care Phase: Once an agreement is reached between n_{nn} and super-peer, n_{nn} enters Take Care Phase (TCP), in which a super-peer tried to enhance n_{nn} 's data availability and thus act as storekeeper. There are various ways through which a node can add new friends. Node n_{nn} can try to search for his acquaintances or friends from userlist

or will ask them physically or through emails to know if they are available on network. By exploring community's membership a user can make friends based on interest. A node tries to find suitable storekeepers among all the friends.¹⁰

As part of agreement super-peers keep track of new node's geographic location and up-time statistics. Each super-peer maintains two kinds of pool as describe below with objective to facilitate users in finding other nodes, which are complete stranger to them and can act as storekeepers and thus help in increasing data availability.

- Time Track pool (TT):As part of agreement, during TCP, super-peer tracks new node's up and down time.
- Stranger pool: Every super-peer maintains a stranger pool along with TT pool. A node can make an explicit request to a super-peer to ask for suggestions about the nodes which can act as storage for it. During request a node submit it's up and down time and ask for respective time units during which it wants the other nodes to cover for it.

All super-peers share TT and stranger pool information among themselves. If a node doesn't get enough uptime availability from its friends, it can request super-peers to make some suggestion for it by using above two pools. Based on two pools, a super-peer can suggest requesting nodes about the (stranger) nodes which can probably act as storage¹¹. Once the suggestions are received about other nodes, a node can request these suggested nodes for storing data. Once an agreement is reached between two nodes, data is replicated and stored in secured way.

3. Settled Phase: A node can extend its TCP with a super-peer if he is not able to get enough storekeepers to get data visibility. Once a node has enough storekeeper(s)

¹⁰One compelling factor for choosing storekeepers could be their uptime, however in general this is not always possible. As in real life we can't compel our acquaintances/friends to store data.

¹¹Nodes suggested by super-peer based on both the pools are termed as stranger nodes.

(other than super-peer) to manage his data, it can relinquish super-peer's services. For every node n , every storekeeper agrees to store a limited amount of data which requires space s . So only latest s data is stored with storekeepers and once the limit is reached, the old data is deleted from storekeepers.

4.2.2.2 Data Synchronization and Updates

This section describe merging, copying and updates of data between storekeeper(s) which is one of the important aspects of DOSN.

Data Storage: When a node n goes down it pushes all its latest data to all the up storekeepers. When node n comes up, it requests all the storekeepers (whoever are up) for the latest data. It accepts the data from the storekeeper, which is having latest data. Node n could be acting as storekeeper for other nodes as well. As a storekeeper role, when a node comes up it synchronizes the data (for each of node, for which it is acting as storekeeper) with other storekeepers or with owner of the data (whoever is up) for which it is acting as storekeeper.

If two storekeepers don't have enough up cross time, then a node can request a super-peer to act as temporary data storage for the latest data. Thus in this case a super-peer acts as a mount point and keeps the latest data from the last up storekeeper which is going down before another storekeeper comes up (with respect to node n). We assume that, before logging off, a storekeeper do transfer completely the latest data on another machine (either super-peer or storekeeper). In this case super-peer doesn't take part in any updates or cater any request from friends accessing the data. When node n comes up and if no storekeeper is up, then node n requests the designated super-peer for the latest data.

When node n is down, friends of n contact respective storekeepers to access the latest data for node n . Any of the storekeepers who are up can be contacted for the request

for the latest data. A particular storekeeper is chosen randomly from up storekeepers for data access.

Updates: When a friend f , posts a comment on node n 's status (or about new uploaded picture or link), an update request is send to node n (if it is up) with a timestamp (logical timestamp). If n is down, requests are made to all the storekeepers who are up, about the updates. Every update request comes with the timestamp ¹². A node n (or storekeeper(s)) might receive multiple updates at same item, in such cases, node n (or storekeeper(s)) uses timestamp to resolve any conflicts.

If neither node n is up and nor any of its storekeeper, then comments/posts on the wall are stored on the commenter side, and resides till node n (or any of its storekeeper) comes up.

4.2.2.3 Data Security

Social responsibilities and mutual agreements are influential in maintaining data integrity. However sufficient procedures should also be taken to ensure confidentiality and integrity of the data. A suitable model has been defined below to satisfy these parameters.

The analogy of the access can be defined as a locker in a bank. The owner of data can be considered as an account holder in a bank, whereas bank can be considered as storekeeper. The key of the locker is with the account holder that is owner in our case. As the key remains with account holder, so even bank can't open the locker thus storekeeper in our case. We presented the solution on same thoughts in our solution which is simple and effective approach that avoids any data-tampering.

This can be achieved using with the concept of embedded database. Participation initiates an authorized management system to maintain data for the user on the storekeeper system. The authorized management system creates four tables, with three tables

¹²Timestamp is the time of the logical clock w.r.t. node n which is shared by friends and storekeepers of node n . Since the updates are w.r.t. node n , so logical clock of node n is used.

for user's data - (1) one for public data (2) another for protected data and (3) one for private data. The fourth table is the authorization table - which stores the information about the friends who can access protected data and their respective public key, to check the digital signature to avoid any misuse. The authorized system also has authorization credentials known only to the owner of the data. For any updates, owner sends a updated program with same credentials to override existing tables. A new program can only execute if its matches the old credentials.

The model includes encryption of data with user specific encryption keys for authenticated access. The model also takes into consideration the various aspects of data security as provided by the database. In this case the concepts of embedded database have been tested to be adaptive to both portability ¹³ and restricted access ¹⁴ as required by the model. The model dictates that once a participation agreement is reached, an authorized management system is initialized to maintain data for the owner on the storekeeper's system.

When a friend sends an update request, as part of request a digital signature is also associated to the request. Once authorized, the system performs the update.

4.2.2.4 System Model

Let $G(N, E)$ be DOSN graph, where N represents nodes (or users) of the network and E represents the set of edges between any two nodes n_i and $n_j \in N$. An edge $e \in E$ exists between n_i and $n_j \in N$ if each n_i and n_j profile's are linked with each other as friends.

Let Ψ represent the global stranger pool managed by all super-peers and SP as a list of super-peers. To increase the availability, a new node can share his data with a particular super-peer n_{sp} s.t $n_{sp} \in SP$. A node can get good availability with the help of friends if many of them are ready to act as storekeepers. We define friends of node n_i

¹³<http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html>

¹⁴<http://www.oracle.com/technetwork/java/javadb/securitywhitepaper10-159253.pdf>

as the set $\aleph_i$ where $n_f \in$

$\aleph_i$ iff $e_{if} \in E$. A node might be able to convince some strangers for storekeeping to boast his availability. Let σ_i represent the set of storekeepers for node n_i , then formally it can be represented as

$$\sigma_i = \{ n_i, \bigcup n_{sk}, s.t \ \forall n_{sk} \ n_{sk} \in \aleph_i \text{ or } n_{sk} \in \psi \text{ or } n_{sk} \in SP \}$$

Let T_i represents the set of time units for which node n_i is online. In decentralized system if the data is not stored other than owner of the data, then data availability is equal to the time units for which node n_i is up, that is $Avl(n_i) = |T_i|$. If a node is able store data on other nodes as well, then availability of node n_i can be defined as

$$Avl(n_i) = |\bigcup_{al=1}^{AN} T_{al}| \ s.t \ \forall n_{al} \in \sigma_i$$

4.2.3 Evaluation

In this section we evaluate using various metrics how a DOSN based on SuperNova architecture will perform. First a description of various parameters for user's behavior is presented followed by actual results.

Network: We use the social network graph from a subset of the DBLP co-authorship graph as a representative social network. Specifically, we use the giant connected component of co-authorship graph from DBLP record of papers published in conferences between 2004 to 2008, comprising of 273798 unique authors. Each author represents a node in the network. The graph has an average degree of 6 with diameter between 6 and 7 (calculated using ANF [118]).

Trace Generation: Node's up and down time is generated synthetically¹⁵, taking into account typical bimodal, and weekly behavior cycles. A trace was generated for two weeks time. We considered following three different factors for the trace generation :

¹⁵We wrote our own simulator in java for analysis purpose.

- **Location:** We consider twelve different (time zone) locations, with each location differing by 2 hours from its adjacent locations.
- **WeekDay Behavior:** We consider six types of behaviors during the weekday.
- **Weekend Behavior:** We consider four types of behaviors for the weekend.

Weekday's and Weekend's behavior were motivated by different working hours which users might follow on their workstations and thus will drive the online time during weekdays. Also some users work during weekends and others simply leisure and don't switch on their workstations. Each user is associated with one of each three different factors and a resultant trace was attached with that user for one week. In the second week, we consider two scenarios: (i) all users have same behavior as in the first week, and (ii) for some users (27 percent) we introduce minor variations in their traces and for few others we generate a major variation (3 percent), while for the rest (70 percent) the behavior remains unchanged.

Various parameters for Storekeepers: In our study we consider three different parameters related with acceptance of node's request for storekeeping the data. Different users have different behavior and to capture individual behavior is difficult so we generalize it for the whole network.

- P_{fd} : This parameter decides with what probability a friend/neighbor is ready to accept the data of a node.
- P_{sd} : The percentage of nodes which wants to participate in stranger pool.
- P_{as} : With what percentage a stranger is ready to accept another stranger's request for storing the data.

Each parameter can take two values representing lowest and highest percentage values taken by the nodes for particular storekeeping behavior. Based on these values we consider 8 different combinations. Table 4.1 summarizes the various combinations that we analyzed by selecting different percentage values for each parameter. The first column represents the combination number to identify the parameter combinations, and 2nd, 3rd and 4th columns represent the percentage values of each parameter.

Various node strategies for storage: A node can request social neighbors or/and strangers for storekeeping. It is less likely that a new joiner will make enough friends/strangers to store its data during few initial days, thus new nodes are more likely to request super-peers for data replication and thus will choose one of the super-peers for sharing of data¹⁶. Nodes will try their best to have maximum availability by replicating their data on various nodes across the network. A node in general can store its data using either of the four following schemes depending on various factors like if its social neighbor's are already overloaded and whether he is (not) willing to store data on nodes run by strangers.

- S_{nns} : A node is able to store his data on some of his neighbors as well on strangers.
- S_{nn} : A node is able to store data only at his neighbors. A node might not be interested in storing data at strangers because either he is not relying or he is not able to convince strangers.
- S_{nsp} : A new nodes store his data on one of the super-peers after agreement.
- S_{ns} : A node might not be able to get help from any of its neighbors for storage, e.g., because all the neighbors are already overloaded because of prior storage agreement. However a node might go for a scheme where he is able to convince some strangers for storing the data.

¹⁶For simplicity in our experiments we assume a node only requests one super-peer for storing its data.

Table 4.1: Different combination of parameters

Combination	P_{fd}	P_{sd}	P_{as}	S_{nms}	S_{nn}	S_{ns}	S_{nsp}
C1	50	50	100	22.4	36.6	16.2	24.8
C2	50	70	100	31.5	27.7	22.7	18.1
C3	20	70	100	14.5	11.9	41	32.6
C4	20	50	100	10	15.9	29.2	44.9
C5	20	50	40	4	22	12	62
C6	20	70	40	5	20	16	59
C7	50	70	40	12	46	9	33
C8	50	50	40	9	45	6	40

Super-peers are assumed to have enough storage space to cater to node's request (specially the new joinees). For each of the parameters combination, we measured the percentage of nodes falling under each category. In Table 4.1, the 5th, 6th, 7th and 8th columns represent the percentage values for each scheme.

We also measure the availability of nodes in each combination. We define availability time as the ratio of the number of time units for which a node n 's data is available to that of total time unit of the simulation. We categories the availability (Avl) into 6 classes namely (names are intuitive for performance) - (1) Excellent ($Avl \geq 95$) (2) Very Good ($95 > Avl \geq 80$) (3) Good ($80 > Avl \geq 65$) (4) Mediocre ($65 > Avl \geq 50$) (5) Poor ($50 > Avl \geq 30$) and (6) Very Bad ($Avl < 30$). For each combination we measure the percentage of nodes falling under each availability category. Figure 4.6 represents the percentage values for each combination of parameters namely from C1 to C8 (refer Table 4.1 for both cases when there is deviation (D) and when there is no deviation (ND)).

Parameter combinations representing the user's behavior has clear effect on the percentage of nodes falling under each scheme. In an ideal situation (Row C2 in Table 4.1 - based on our parameters), half of neighbors (or friends) ($P_{fd} = 50\%$) are ready to accept storage request and more than half of the nodes in the network ($P_{sd} = 70\%$) are ready to participate in stranger pool and all the strangers ($P_{as} = 100\%$) are ready to accept

Figure 4.6: Performance of different parameter combinations for Deviation (D) and Non Deviation (ND)

all the requesting for storage. In this situation there is minimum load on super-peers (comparing with all other cases). Also the number of nodes falling under excellent availability category is at its maximum. Although deviation has introduced a sharp decline in percentage of excellent nodes however the total numbers of excellent and very good nodes are still higher. When the percentage of nodes ready to store data at strangers declines (in c1) there is a very small effect on the excellent nodes (compare ND1 and ND2 in Figure 4.6).

However when the participation of number of nodes willing to participate in stranger list declines (from 70 % to 50 % for P_{sd} in Row C1) we notice a immediate load on super-peers. This is better than C7, which represent the situation when the percentage of nodes willing to store stranger's data decreases. Thus presenting the fact that parameter P_{as} is more influential than P_{sd} for increasing the load on super-peers.

Maximum load (62%) on super-peers is observed when neither neighbors are interested in storing other friend's data and nor the participation of nodes in stranger's scheme is overwhelming (Row C5). Although presently, storage is not costly anymore, however if super-peers have limited space, system might explode.

There should be a balanced behavior among nodes for a system to be successful. When the percentage of neighbors, who are willing to store neighbor's data is low (Rows C3 and

C4), the system can still sustain itself if the participation of strangers is overwhelming. This depicts a scenario if all new joiners act as storekeepers for each other and thus increasing availability. This collusion of stranger support (in need of time) may eventually lead to change in relationship from strangers to friends.

We noticed high load on super-peers when percentage of the nodes falling under stranger scheme is high and when percentage of nodes ready to accept stranger data is low (Row C6). As soon the participation of neighbors is increased there is immediate decline on super-peers load as evident from load Row 7 which is attributed due to the fact that nodes rely more on neighbors than strangers. The system shows a very poor performance when all the parameters are at their lowest (Row 5) - clearly showing the decline in percentage of excellent and very good availability (D5 in Figure 4.6). By comparing C6 and C7 we infer that it's not merely participation of nodes in stranger scheme is important. The more important factor is how much percentage of nodes are ready to accept stranger's data.

Although nodes are more willing to store at friends. However for new nodes and the nodes who are not able to find storage among neighbors and strangers, super-peer can be of great help. From above values we observed that system gives best results when everyone cooperates (C2) and is at its worst when users are not ready to contribute to the system (C5) and act meanly. After careful consideration, we selected parameters combination in Row 8 as realistic scenario where probability of accepting neighbors are ready to store other neighbor's data, also 50% of the nodes are ready to participate in stranger scheme however out of total nodes participating in scheme, 40% are ready to accept stranger's data.

We compare availability in three different scenarios. (1) To high light the effect of super-peers we compare super-peer vs flat scheme. (2) A comparison between an ideal (or best) case vs worst is also presented to reflect the effect of cooperation and meanness

(meanness means when the nodes become selfish or in other words, when the nodes are not willing to store other users' data on their machine.) respectively. (3) Few nodes might be interested in getting availability in terms of the time when their friends are online vs. 24/7 availability. In evaluation of all the three cases - along with measuring cumulative availability we try to gather information about the percentage of availability among various nodes. As a micro-measurement, we also compare the individual scenarios in terms of availability in terms of 6 categories defined above.

1. Availability w.r.t total time and time when friends are online: A node could be interested in getting availability only when his friends are online instead of getting 24/7 availability. Interesting results were gathered while comparing two different cases. Figure 4.7 compares availability when total time (TT) is considered and friends up time(FT). Figure 4.7.a gives the cumulative availability when deviation (D) is introduced as well as when system is evaluated with no-deviation (ND). We present our results for a more realistic case (C8 in Table 4.1) out of all the combinations.

When a node is only concerned about the availability when his friend's are online, and there is no-deviation in the node's uptime behavior (FTND), system outperforms the total time with no-deviation (TTND). When only friend's up time is considered there are more than 50% of nodes whose availability is 100% as compare to total time (TT) where the value is less than 20% (refer Figure 4.7.a). However, with introduction of deviation, in case of friends up time (FTD) system sharply declines in terms of performance as compare to total time (TTD), where deviation doesn't put much effect on the performance. However in both cases 100 % of nodes has at least availability greater than 40%. With the help of Figure 4.7.b we capture more microscopic information in terms of percentage of nodes getting what percentage of availability. This information is important to gather individual behavior as some information is missing in cumulative. In case of FTND the highest percentage of nodes is around 53 % however it declines to around 5% when de-

violation is introduced. From the Figure 4.7.b, one can also infer that there is not much deviation in terms of individual percentage after deviation is introduced.

It's common that friends are generally collocated. So if a node stores the data among his friends, because of location factor, most of the friends will be online during the same time and thus availability will be high for such nodes. For same reason, most of the nodes get excellent availability. However the value might be low if we compare the total time of the system which is much higher than the friend's online time and thus a decreases in availability value. When deviation is introduced there is sharp decrease in availability time since if friends deviate from their usual uptime behavior they are responsible for decreasing the data-availability as compare to a node which has scatter his data, like by selecting some strangers as well. This observation can be verified with the help of Figure 4.7.c, where the total percentage of nodes for "excellent" plus "very good" in case of friends after introducing deviation (FD) becomes less than the total excellent nodes without introducing the deviation (FND). If we consider the total time as measure to find availability we do notice that there is not much difference in performance (excellent plus very good) between no deviation (NDTT) and when deviation is introduced (TTD). This also attributes to the fact that if a node spreads his data other than friends then he will be least affected by the deviation and thus he can either choose strangers, and if strangers are not willing to do so he can take help of super-peers.

2. Compare Flat Vs. Super-peer Technique: The effect of super-peer (SP) scheme Vs. flat scheme (F) is also compared. A flat scheme has no concept of super-peers and all nodes are of equal resources and have equal up time. Although it's hard to keep strangers' list and it's difficult to trust strangers without super-peers which are trusting authorities through which peers can find other stranger nodes to increase availability. To give flat scheme a fair chance we did assume that nodes can convince strangers to store their data at least by tit for tat scheme (tit for tat means that one party is ready to

Figure 4.7: Comparison for Friend's Time (FT) and Total Time (TT) for Deviation (D) and NonDeviation (ND)

store other party's data contingent on other party storing its data.) and strangers can be trusted for doing so.

We observe that because of the absence of super-peers the new joiners are not able to get enough space for data-replication. As in the bootstrap period for a new node it's difficult to find friends and to convince strangers to store their data. This is a major cause of poor performance in the case of flat scheme. Figures 4.8.a and 4.8.b show that there are very few chunks of nodes whose availability is above 60 %. Deviation (FD) downgrades availability for 30 % of nodes (refer Figure 4.8.a - between 30% to 60 % nodes on x axis). Also we observe there are very few percentage of nodes (8 %) whose availability is excellent which further becomes less than half of non-deviation (FND), in case of deviation (FD). Results are evident to prove that super-peer scheme outperforms the flat scheme tremendously.

3. Best VS. Worst case: We also compare the best parameter (B) combination and worst (W) one (C2 and C5 respectively in Table 4.1). In Worst case, the low parameter values indicate that every peer is acting meanly and evidently super-peers are having maximum load in this scenario out of all (please refer Table 4.1, Row C5 and

Figure 4.8: Comparison for Flat(F) vs. Super-peer(SP) for Deviation (D) and NonDeviation(ND)

column S_{nsp}). Super-peers have uptime comparatively higher than that of normal peer (however not 24/7). As large numbers of nodes are relying on few nodes (super-peers) and when super-peers are down, the data availability for large number of nodes decreases thus lowering the performance of the system. Figure 4.9.a measures the cumulative values in worst case for both deviation (WD) and non-deviation (WND) and compare them with Best case for both deviation (BD) and non-deviation (BND). These values were compared w.r.t total time.

The low parameter shows nodes are unwilling to act as storage for others nodes. From Figure 4.9.b one can clearly note that the highest availability is around 90% for little more than 5 % of nodes as compare to the nearly 25 % when every node cooperates in case of non-deviation (in best case). The cooperation phenomena is also evident in terms of percentage of nodes getting "excellent" plus "very good" availability both in deviation and non-deviation for best case in Figure 4.9.c (BD and BND respectively). In worst case after introducing deviation the system is behaving more in a mediocre range where storekeepers might not be able to exchange latest data thus leading to inconsistency. Also from Figure 4.9.a it's easy to infer that most of the availability is 60 % or less for 60 %

of the nodes. Also for non-deviation most of the nodes are not getting good availability as around 50 % nodes are having availability 70 % or less.

Figure 4.9: Comparison for Best(B) vs. Worst(W) parameter combination for Deviation (D) and NonDeviation(ND)

4.2.4 Conclusions

We proposed a super-peers based architecture to facilitate decentralized online social networking. The proposed architecture provides flexibility to individual users to choose their strategy in terms of where to store their data, as well as whose data to store. While we do not investigate the role of trust and reputation explicitly in this work, recent trust model [127], which correlate node's high reputation with them being more influential custodians of the community's tasks, provides a premise to carry out such a dynamic analysis of node's behavior in a SuperNova like super-peer based DOSN. We investigate the 'static' characteristics of the system. Specifically, we study how the influence of different user strategies affects the performance and sustainability of a DOSN. Existing initiatives, both academic and open-source ones, have in recent years focused on developing the enabling technologies (some of which, particularly related to security, could be incorporated in SuperNova as well). However, a quantitative estimate on whether such

a system is sustainable or not, given diverse participation behaviors and capacities has so far not been investigated. Thus, besides proposing a more pragmatic system design, this work distinguishes itself from existing work in the rigor of the evaluations.

4.3 Epilogue

We first analyzed the real traces of an egocentric network to validate if friends are enough for a 24X7 availability. Based on the insights from this work, we proposed a super-peers based architecture, SuperNova. Super-peers especially help new joiners in the system, in introducing strangers, when friends are not enough to increase data availability. Among all the strangers, nodes are always looking for the top best (in terms of resources or online behavior) strangers which can help in increasing the data availability. In the next chapter, we present a solution for another important problem that is how to find top-k nodes in a decentralized settings.

Chapter 5

DRank: Decentralized ranking mechanism for semantic community overlays.

After presenting the solutions for the problems of information dissemination and data availability in chapters 3 and 4 respectively, in this chapter, we present a decentralized ranking algorithm for finding top-k users in a semantic social overlay based network. In large semantic networks the problem of finding top k users (or nodes) with respect to a particular topic is important. Be it a co-authorship graph where a author is looking for other top-k authors with respect to a topic, or the problem to find top-k influential nodes with respect to an interest (or topic) in a social network. In large networks, global knowledge is difficult to keep at individual nodes because the networks are (i) dynamic in nature and (ii) usually scale to very large numbers. Hence there is a necessity to design algorithms based on local neighborhood. Our proposed algorithm exploits social links and uses local information only. The experimental results on both synthetic and real-world datasets show the effectiveness of our approach.

5.1 Introduction

In a semantic social network, users may need to find top-k nodes with respect to a particular interest. For example, in a coauthorship network a user (author) might be interested in finding top-k users with respect to a particular topic. In *LinkedIn* [7] a user (consultant) might be interested in finding top-k professionals with a specific expertise. The quest for finding top-k users might be based on a (i) business aspect, or (ii) to make more social contacts with influential people, or (iii) for doing collaborations with people having similar interest.

The problem of finding top-k expert users in this work is inherently different from set of problems that were exhaustively studied in the past where the aim was to find top-k items in distributed systems such as on the web [22], or in a P2P network [71]. These algorithms were centralized in nature. PageRank [22] algorithm works well on a repository that has already been scanned and processed by a centralized system. A hybrid approach [71] was used where the authors explored super-peers to calculate top-k resource rich nodes. In this work, we investigate how to find top-k users in a semantic social network with respect to a particular topic (or interest). We consider the underlying network as decentralized and dynamic in nature. We have compared our proposed work with existing literatures in chapter 2, section 2.2.3 in detail.

In *LinkedIn* which is a large bidirectional semantic social network for professionals working in different fields, users are often looking for other users who are considered best in their field. It is feasible to find top-k users in a particular field using a centralized server. A user can find top professionals based on the number of social links and experience. *Twitter* [128] is another example of a large¹ semantic network where relations or links are directional in nature, that is a user A might be following a famous personality B on *Twitter*. However, reverse might not be true. Users follow other users based on interest or

¹<http://www.pcmag.com/article2/0,2817,2371826,00.asp>

acquaintances. In *Twitter* users are always looking for top-k influential (or information) users (or providers) in the network with respect to a particular interest. Because of a large user base a user has to manually decide about top-k nodes based on various parameters such as degree (number of followers), interest matching and activeness. However, still the process is facilitated with the help of a centralized server.

Some of the social networks try to solve this problem with the concept of communities. In centralized systems like *Facebook*, there is an explicit concept of communities ² where a user can join a particular community and look for history of the users to decide about top-k users. Such a community concept cannot be used as it is in large decentralized networks, as no single user can have the global knowledge of the whole network because of limited resources and lack of global information. In question-answering system (for example yahoo answers [129]), which is an another example of a centralized system, a user can look for best answers/users using ratings or normal search options provided by a central service provider. These centralized mechanisms make the task easier as central server has all the information about all the users. However the task to find top-k users is non-trivial in decentralized networks.

LinkedIn, *Twitter* and *Facebook* are examples of explicit semantic social networks in the sense that it's the common interest which attracts and motivates users to join these particular networks. In contrast to explicit social networks there are some implicit social networks like the network of email contacts formed by the interaction of different users. Co-authorship graph is another example of implicit social network. These networks are decentralized in architecture as there is no central authority to cater to users for any service. Our proposed algorithm can be used in such decentralized networks such as in coauthorship graph to find top-k authors working in a specific domain.

²<http://www.facebook.com/help/?page=1067>

In this work we propose a decentralized algorithm which uses only local information. We are partially motivated by GOSSPLE [130] in which users periodically gossip digests of their interest profiles and compute their score in terms of interest with respect to other nodes. Compared to GOSSPLE, where nodes can gossip with other nodes (known or unknown) based on interest matching, in our approach nodes exchange information only with their neighbors. The proposed algorithm can scale for any number of nodes in a network. The algorithm is generic enough and it can be applied to different kinds of problems. For example it can be used to find (i) influential nodes in the network, or (ii) the most vital player in MMOG, or (iii) top authors in a coauthorship network. The algorithm also has its application in recommender system to find top-k experts for a particular domain or users looking for a news in a completely decentralized network [131]. We performed our experiment on real (DBLP) as well as on different kinds of synthetic networks including small-world and scale free networks to show the effectiveness of our approach.

The rest of the chapter is organized as follows. We present our approach including various concepts and algorithm in section 5.2. Section 5.3 evaluates the approach using synthetic and real networks.

5.2 Approach

In this work we study the problem of ranking in the decentralized setting of a large semantic social networks subject to the constraint of using only social links. In other words, nodes use local information available for calculating top nodes with respect to a topic in the network. In this section we describe the approach to calculate the rank in a decentralized mechanism in semantic community overlays.

5.2.1 System Model

Let $G(N, E)$ be a graph, where N represents the set of nodes (or users) of the network and E represents the set of edges between any two nodes n_i and $n_j \in N$. Edge $e_{ij} \in E$ exists in the network, if node n_i and n_j know each other. We define the neighborhood of node n_i as the set $\aleph_i$ where $n_j \in \aleph_i$ iff $e_{ij} \in E$. Let t represents a particular topic and T as the collection of the whole topics of all the users in the network. A node n_i could have an interest in more than one topic. Let T_i represents the collection of topics in which node n_i is interested.

5.2.2 Algorithm 1: Simple Case

We first describe our algorithm in a simple scenario by making some assumptions. Then in section 5.2.3, we will relax these assumptions and explain the algorithm to accommodate more complex situations. The proposed algorithm is designed for a pure decentralized mechanism where each user tries to find top-k users with respect to a topic. Each user computes a relevance score with respect to a topic and shares it with its neighbors.

We first describe the relevance score of a user with respect to a topic before going into the details of the algorithm. Relevance score basically indexes the user for a particular topic t . The higher the score, the more relevant a user is with respect to the corresponding topic. The relevance score can be calculated in different ways depending upon the network being considered. For example, in a co-authorship graph one way to calculate the relevance score ($Relv_n(t)$) for an author n and topic t could be the number of papers published by the author (user) for a topic t . As the main focus of this work is about calculating the top-k nodes' information, we assume that all the users calculate their relevance score with the same formula.

We make the following assumptions for a simple scenario. We do provide an algorithms for more complex scenarios by removing these assumptions later in section 5.2.3.

- (i) Each node fairly calculates its relevance score for a topic t .
- (ii) Each node exchanges relevance score in a fair manner with its neighbor.
- (iii) Every node is interested in every topic.

To simplify the presentation, we explain the algorithm to find the topmost node ($k=1$) for a topic t , which can be easily extended for $k > 1$. Initially each node n assumes it is the top most node for a particular topic t . We call this iteration 0. All the nodes share their relevance value with all its other neighbors. At the next iteration (iteration 1), node n updates the relevance value (and corresponding entry for the node which has the highest value) if any of the neighbor has the higher relevance value than its own. In the next (and corresponding) iteration(s) nodes again share their updates (which they gather from their neighbors in the previous iteration) about top- k node information with all the neighbors. The process of sharing the relevance value is a continuous process.

For a better understanding, we explain the algorithm with the help of a synthetic network in Table 5.1. Please see the corresponding network of the example graph in Figure 5.1. In row 1, from column 2 to column 7, the format of cell values are of the form Node (relevance value) (Adjacency List). The result of the execution of algorithm is shown from Row 2 to Row 7. At Iteration 0, each node assumes it is the topmost node for the topic t - this is shown in 2^{nd} row in Table 5.1. The column in the table shows the value for each node with each row representing a particular iteration. The format of each cell is $\{P_{th}(TN, val)\}$, where P_{th} represents the path (set of node(s)) by which the value is obtained, and TN represents the top node whose value is represented by val . Since at iteration 0, each cell assumes it is the top most node for a particular topic t , therefore the path value is the node itself. In the next iteration (iteration = 1), all nodes consult their neighbor(s) for the topic t , to know if any of them have a higher value and if that is the case they change their values accordingly. For example node c 's and f 's

relevance values are higher than their respective neighbors and so they don't change it. Other nodes, for example a and b , update topmost node as c for topic t at iteration 1. Thus each node performs these kind of iterations and eventually every node is aware of the topmost node.

Since the node f with the highest relevance value in the network is on the boundary of the network, it takes almost d iterations (where d is the diameter) for the algorithm to converge. This is the worst case. Although in this example we showed that node f is having the highest value of relevancy for topic t , generally people having higher relevance value act as hubs for the community [132] and have high degree (in terms of friends). Thus, we argue that our algorithm will take much lesser number of iterations than the diameter of the network. This is validated during experiments as well (section 4.2.3). Also in the past, research such as [110], [133], [134] has concluded that real networks do exhibit small world phenomena and generally the diameter is not large for such networks. Hence the number of iterations will not be very large.

Table 5.1: Algorithm's execution on a pseudo network

Iteration	a (0.34) (c)	b (0.45) (c)	c (0.78) (a,b,d)	d (0.67) (c,e)	e (0.32) (d,f)	f (0.84) (e)
0	{a(a,0.34)}	{b(b,0.45)}	{c(c,0.78)}	{d(d,0.67)}	{e(e,0.32)}	{f(f,0.84)}
1	{c,a(c,0.78)}	{c,b(c,0.78)}	{c(c,0.78)}	{c,d(c,0.78)}	{f,e(f,0.84)}	{f(f,0.84)}
2	{c,a(c,0.78)}	{c,b(c,0.78)}	{c(c,0.78)}	{f,e,d(f,0.84)}	{f,e(f,0.84)}	{f(f,0.84)}
3	{c,a(c,0.78)}	{c,b(c,0.78)}	{f,e,d,c(f,0.84)}	{f,e,d(f,0.84)}	{f,e(f,0.84)}	{f(f,0.84)}
4	{f,e,d,c,a(f,0.84)}	{f,e,d,c,b(f,0.84)}	{f,e,d,c(f,0.84)}	{f,e,d(f,0.84)}	{f,e(f,0.84)}	{f(f,0.84)}

Figure 5.1: Figure showing the sample graph.

5.2.3 Algorithm 2: Complex Cases

In this section, we present an algorithm by relaxing the assumptions stated in section 5.2.2.

- **Byzantine nodes:**

Although we assume in section 5.2.2 that nodes will calculate and share the values in a fair manner, there could be some malicious nodes which try to spread incorrect value of their relevance score with respect to the topic. It may be possible that two or more nodes collude to spread wrong information. For example, in Table 5.1 node e and f can collude to spread wrong information about f , so that f can become the top most node in the network for a particular topic t which might not be the case in reality.

To discourage collusion among malicious nodes, we introduce the concept of trusted authorities in the network. For every topic or interest, a trusting authority could be a third party or any node in the network with whom some minimum number of nodes are ready to trust. A node has to register with more than one trusting party for every topic it wants to participate. These trusting parties again register themselves with other trusting parties dealing with a totally different set of topics³. We assume no trusting authority will collude with malicious users who are trying to spread false values as it might affect their reputation. In case of any dispute, the path can be submitted along with the values to a trusted authority for verifying the value of the node. In case of a conflict, all trusting authorities with respect to a particular topic can share their information and can decide whether a particular node has provided correct information or not.

³Providing a full scale trust-mechanism is out of scope of our work.

- **Fragmented Communities:**

Not every node is interested in every topic. Thus, there is a high probability that a node will participate in the algorithm only if it is interested in the topic. An example is a coauthorship graph in which authors work only in limited areas. This limited interest of the nodes might lead to fragmented communities. That is to say, with respect to a particular interest, there might be islands of subgroups, connected to each other through nodes having totally different interest. Thus, when not every node is interested in every topic, it is difficult for the system to converge and find global top-k nodes w.r.t the whole network.

The algorithm will work well within the subgroups. Based on our past experience [23], we make use of random walk to find top-k nodes from different subgroups. The nodes of subgroups participating in the ranking process can start a random walk with some probability p_1 . The participating nodes of a particular subgroup can take the help of nodes having non-matching interests, to find distant subgroups of same community. There could be a scenario, in which the distant subgroup might have a user which has higher relevance value than the subgroup which has started a random walk. The message in the random walk consists of the top node of the subgroup and its relevance value.

To avoid infinite walk in the random-walk, each node starts a random walk message with energy E which decays with every hop by an amount δ . We also use a simple mechanism to avoid self-loops by letting each node remember the last random walk it participated in and the node to which it forwarded a message during the last random walk. If a node receives the same information which it forwarded to some node n_x during its last participation in the random walk, then, in the next attempt

it forwards the message to a node other than n_x ⁴. A node uses top node information in the message to detect a subgroup of community.

The pseudocode of the process of updating the score and starting a random walk is shown in Protocol 3 and 4.

Protocol 3 Each node n_i calculates its own score Sq_i as soon there is any update in a parameter(s) being used for score calculation. If the score of the node is higher than any of the top-k nodes $Sq(Tp_i)$ know to node n_i then it updates its own top-k node information and also pushes it to its neighbors $\aleph_i$.

```

 $Sq_i \leftarrow Score(i);$ 
if  $Sq_i > Sq(Tp_i)$  then
    Update  $Tp_i$ 
    Push  $Tp_i \forall n_j$  s.t.  $n_j \in \aleph_i$ 
end if

```

Protocol 4 Every time a node n_i receives a new update from a neighbor n_j , it updates its top-k information, if the information provided by n_i is more than the top-k entries of Tp_j . If the node n_i has one or more non-relevant nodes as its neighbors, node n_i randomly selects one of the non-relevant node n_k to start the random walk.

```

if  $Tp_j > Tp_i$  then
    Update  $Tp_i$ 
    Push  $Sq_i \forall n_j$  s.t.  $n_j \in \aleph_i$ 
    if  $|NonRelv| \geq 1$  then
         $DelvMsgTo(n_k)$  s.t.  $n_j \in \aleph_i$ ; {Start the random walk with the help of a non-relevant node. Node  $n_i$  selects a non-relevant node  $n_k$  from all the non-relevant nodes by uniformly at random.}
    end if
end if

```

5.3 Evaluation

In this section, we present the experimental results on synthetic and real datasets. In section 5.3.1 we first explain the datasets used. Then, in the next section 5.3.2 we explain the results.

⁴In our experiments, we kept the limit of two for a node to participate in self-loops to avoid infinite self loops.

5.3.1 Dataset

We first explain the datasets used in our experiments. We perform our experiments on both real and as well as on synthetic datasets to evaluate our algorithm. We perform our experiments on a static snapshot of the networks i.e., the relevance value of the nodes do not change during the entire run of the experiment.

5.3.1.1 Synthetic Dataset:

We generate two kinds of synthetic network using the NetworkX⁵ graph generator.

- **Scale free network:** Scale free networks [135] follow power-law degree distribution, at least asymptotically. In other words, most of the nodes have few edges and a very small percentage of nodes have a very high degree. Many networks in real world appear to be scale-free. The examples include the world wide web, citation networks, and also some social networks.
- **Small world networks:** In small world networks, most of the nodes not immediate neighbors. Although, from any given node, other nodes are reachable within a few hops. The collaboration network between Hollywood actors is a good example of this type of networks. Watts and Strogatz proposed a graph model in [133] to emulate such kind of networks. In our simulation we created the graph with $\beta=0.5$ and got a diameter of 15 (calculated using MCL [136]).

The model was proposed by Watts and Strogatz [133] to improve the short coming of random graph which have low clustering coefficient.

5.3.1.2 Real Dataset - DBLP Graph:

We created a coauthorship graph from the DBLP record of papers published between 2004 to 2008. Since the graph had many disconnected components, a big connected

⁵<http://networkx.lanl.gov/>

component was taken for our experiments, having 231704 nodes. We categorize 1169 unique conferences into different sets of interests (e.g., data mining, database, etc.). Each node is associated with different sets of interests based on the conference in which they published the papers.

5.3.2 Results

We first present the results for the simple case in section 5.3.2.1 and later for more complex cases in section 5.3.2.2.

In both the simple and in complex case, we carefully picked a single node and assigned it the highest relevance value. Also, the main focus of our experiments is to find out how fast the algorithm helps in convergence under different network characteristics.

5.3.2.1 Simple Case:

In the simple scenario, we consider all the nodes in the network having the same single interest in topic t . Thus, every node is interested in getting the top-k node information. This also implies that all nodes participate in the ranking algorithm. Nodes' relevance value was assigned uniformly at random in all the three networks.

- **Iterations:** Table 5.2 shows the result for the simple case for different networks. Column 1 represents the type of network and the information about the number of nodes in the network is provided in column 2. We define “iterations” as the maximum number of rounds after which every node in the network is aware of the top-k node. For example in Table 5.1, after the 4th iteration every node is aware that the node f is the topmost node with respect to topic t . We ran 20 experiments for each type of network. In each of these iterations, the nodes were assigned different relevance values (assigned uniformly at random). The value for “Iterations” column shows the average value of these 20 iterations. The different

network parameters were calculated using MCL algorithm [136]. In each case, the number of iterations have not crossed diameter. The number of iterations is almost half in case of synthetic graphs and one fourth of the diameter of DBLP. This is due to good cluster coefficient values in different networks. Although the diameter of the DBLP is biggest among all the networks, however it has a high cluster coefficient, which helps in faster spreading of the information in the network. The small standard error values attribute to tightly knit communities.

Table 5.2: Iteration values in the simple case

Graph Type	# Nodes	Diameter/Avg. Path Length	Cluster Coefficient	Iterations	STD. Err.
Watts and Strogatz	50000	15 / 9.0139	0.067	8.15	± 0.028258309
Scale Free Network	50000	6 / 4.1020	0.002	3.65	± 0.049500955
DBLP	231704	23 / 6.9185	0.542	7.9	± 0.279396694

Table 5.3: Iteration values in the complex case

Graph Type	# Nodes	Interest Assignment	Diameter/ Avg. Path Length	Cluster Coeff.	Itr.	STD. Err.	% Nodes Spammed
Watts and Strogatz	50000	Associativity	15/9.0139	0.067	6.25	± 0.0201	0.03%
Watts and Strogatz	50000	Random Cliques	15/9.0139	0.067	13.05	± 0.0564	2.25%
Watts and Strogatz	50000	Totally Random	15/9.0139	0.067	13.75	± 0.0893	2.98%
Scale Free Network	50000	Associativity	6/4.1020	0.002	3	± 0.0305	0.09%
Scale Free Network	50000	Random Cliques	6/4.1020	0.002	5.05	± 0.0501	2.23%
Scale Free Network	50000	Totally Random	6/4.1020	0.002	4.50	± 0.0532	2.90%
DBLP	231704	Conference Based	23/6.9185	0.542	12.95	± 0.4537	3%

- **Iterations vs. % nodes:** We also calculate what percentage of nodes were able to receive the top-k information after every round (Figure 5.2.a). We observe that about 90 % of nodes have got the top-k node information by the 4th iteration. In case of scale-free graph (represented by “SF”), almost 99% nodes receive top-k node information by the 2nd iteration itself. The small diameter and small average path length has helped SF network in achieving this. In the case of DBLP, the small number of iterations are attributed due to high cluster coefficient value (undermining the large diameter effect). In case of small-world (“SW”), high cluster coefficient (as compare to scale-free network) and also low average path length have helped in achieving the top-k information by most of the nodes within few iterations.

5.3.2.2 Complex Case:

We do observe in real world that not all nodes are interested in all the topics. People have limited interest and they are only interested in finding the top-k users in their field of interest. For example, a researcher working in the area of networks might not be interested in finding top-k nodes in the field of machine learning. Thus, we assign different interests to different users in our synthetic network. In case of DBLP, the conferences were used for determining the interest of the author (or nodes) such as data mining, database, etc. In case of synthetic networks, we created three different types of interest assignments as following⁶ :

- **Associativity:** Generally users working in a particular area know people working in that area, if not directly, at least indirectly. Thus, they all form a tightly knit community. However, there might be a few people working independently or newcomers, who might not be well connected to the group. We call these types of

⁶Please see examples of these different interest type assignment on page 3, Figure Y.

communities as associativity based communities. Coauthorship graph is an example of this type. To create such type of communities, we choose a node randomly and move breadth-first around the picked node to form one single clique and then a few nodes are selected which are not connected with this clique.

- **Random Cliques:** In another type of community scenario - there could be soccer fans who are well connected with other soccer fans in a particular area, but they might not know every soccer fan in the country. We call this type of interest assignment as random cliques. In our experiments, we choose three nodes that are well separated with respect to diameter as seeds of random cliques. We then start choosing neighboring nodes of these seed nodes and move breadth-first to form three subgroups of the same community. The size of each subgroup is smaller than 300 nodes.
- **Totally Random:** In a totally different type of community which we call as totally random, the users with same interest are scattered randomly in the network.

After assigning the interest to the nodes, we then assign the relevance values uniformly at random to the nodes.

- **Iterations:** As in simple case, we calculate “Max Iterations” (MI) in various complex cases as well. Table 5.3 shows the result for different types of networks and for different interest assignments. Columns 1 (Network type) and 2 (number of nodes) are same as in the Table 5.2. Column 3 provides the information about the type of interest assignment on top of synthetic network. The associativity type interest assignment is quite similar to the simple case (see section 5.3.2.1). The only difference is that the number of nodes and diameter are small in this kind of assignment. It is interesting to note that despite smaller diameter, because of the

few isolated nodes, the iterations have reached the value of seven in case of the Small World (SW) graph. In case of scale-free (SF) network, the isolated nodes are not affected much.

For random cliques we carefully selected three nodes such that each node has a distance of six or more, and three or more from other two nodes, for SW and SF graph, respectively. We then build small communities around these central nodes such that no two communities get overlapped with other. Interesting results are observed in case of random cliques (R.C) and totally random (T.R). In case of random cliques, as the cliques are not well connected, so each boundary node starts a random walk with probability p_1 to find another distant clique. Random walks are exploited in the case of totally random interest assignments as well. With probability p_1 , nodes ask for the help from neighbors with non-similar interests to initiate a random walk to find other nodes. In both the cases (random cliques and as well as in totally random interest assignment), we observe that non-continuity of the interest among the nodes leads to an increase in iterations.

We also measure the percentage of nodes with non-similar interest which participate in the random walk. As these nodes are not interested in the particular topic, thus the message about top-k node from a different interest is a spam. Column 8 shows the percentage of nodes being spammed during the execution of the algorithm. In all cases, the percentage of nodes being spammed have not cross more than 3% value which is a small number, thus showing the effectiveness of random walk exploration.

The value for number of iterations for totally random and random clique doesn't differ much in SW as well as in SF. In case of SF, the few nodes with a high value of degree increase the probability of reaching out to scattered random nodes in a

faster way. In case of SW, the presence of few short cut paths, is a prominent reason for speedy convergence.

In case of DBLP, despite having a large diameter in the network, the presence of a high cluster coefficient attribute to less spamming and faster communication⁷ between different subgroups of the same community. The aforementioned reasons also lead to smaller value of MI.

The small standard error (7 Column in Table 5.3) values are due to the fact that convergence has happened fast irrespective of the position of the top-k node in the network. In DBLP, the value of standard error is high compared to the synthetic networks as the DBLP network is quite large (almost 6 times) as compared to synthetic networks.

- **Iterations vs. % nodes:** Similar to the simple case, the percentage of nodes which were able to receive the actual top-k information after every round for various complex cases is shown in Figure 5.2.b (for SF), Figure 5.2.c (for SW) and Figure 5.2.d (for DBLP). In case of SF, for random cliques and totally random nodes, 90 % nodes got top-k information in three iterations itself. This is due to small diameter of the network. The low diameter undermines the low cluster-coefficient value. In case of SW – associativity (Ast) is much better than random clique (R.C) and totally random (T.R) because of tightly knit society. In case of random cliques, once the information has reached the boundary nodes, then random walks are attempted to find out distant subgroups of the same community. The small increase from 4th to 7th round of iteration is due to coverage of few disconnected nodes. Once the other subgroups of the community are reached, there is a shoot up in the percentage of nodes getting the top-k information (at 9th iteration). In

⁷Compared to synthetic network (in spite of having a large diameter and a bigger network). Please see Figure 5.2.d.

comparison to random cliques, in case of totally random assignment there is a slow and gradual increase in percentage of nodes, similar to DBLP, symbolizing a lot of scattered subgroups of the same community.

5.2.a: Simple case

5.2.b: Scale free network

5.2.c: Small world network

5.2.d: DBLP

Figure 5.2: Iterations vs. % of nodes

5.4 Conclusions

We have described and evaluated a decentralized ranking algorithm (DRank) for semantic social overlay based networks such as coauthorship graph. The proposed algorithm uses only local information and is purely decentralized in nature. That is to say nodes communicate with other nodes with which they are socially connected. The social constraint

makes it possible to exploit the algorithm in decentralized settings. The effectiveness of the DRank approach was verified by evaluating it in various kinds of synthetic and real networks settings.

Part III

Concluding remarks and future directions

Chapter 6

Conclusions and Future Work

This chapter summarizes the report and presents several future directions.

6.1 Summary

Recent years have seen several earnest initiatives from both academic researchers as well as open source communities to implement and deploy decentralized online social networks (DOSNs). The primary motivations for DOSNs are privacy and autonomy from big brotherly service providers. However decentralization introduces many challenges. In this thesis, we investigated three challenges in a DOSN, namely - (i) information dissemination, (ii) storage and data availability, and (iii) finding of top-k nodes. The problems were investigated under decentralized settings. The solutions proposed for various problems were driven by principles of social sciences and took into account the human behavioral model which makes them realistic and feasible in the real world.

Ease of communication and interactions is an important reason for the success of OSNs. To make DOSN also successful, we do need a basic mechanism by which users can disseminate a piece of information. Information dissemination in a decentralized network is a challenging issue as no single node can have knowledge of the whole network for reasons like security, access control, and limited resources. Most importantly - a centralized service provider is missing which can cater to user requests. We proposed and

investigated an algorithm for information dissemination in decentralized networks which only uses local information - that is information associated with a node and its neighbors for disseminating a piece of information with a goal of maximum coverage and minimum spamming. Along with exploring interests for focussed dissemination, the approach also explores multiple relations that exist among nodes while selecting potential forwarders in case members having same interests are not well connected. We presented a socially inspired, simple dissemination mechanism/approach which is generic enough to be used in various kinds of scenarios: for example in job market or network of emails, decentralized recommendation and contextual advertisement systems. We measured various metrics such as recall, precision and message duplication to analyze the quality of our protocol. The experimental results on real and synthetic datasets show the effectiveness of our approach.

One of the principal problems is to guarantee availability of data even when the data owner is not online, so that others can access the said data even when a node is offline or down. Intuitively this can be solved by replicating the data on other users machines. Whereas the solution for information dissemination stresses more on interest exploration of social relations, the problem of storage and data availability cannot just rely on social relations. The solution must take into consideration the social-temporal behavior of the nodes to be effective. Peer-to-peer networks where data is exchanged and stored solely by nodes owned by trusted users, can guarantee dependability, privacy and uncensorability by exploiting social trust. However, the limitation of storing data only with friends can be detrimental for data availability: if no friends are online, then data stored in the system will not be accessible. In the first work on data availability, we explore the trade-offs between redundancy (*i.e.*, how many copies of data are stored on friends), data placement (the choice of which friend nodes to store data on) and data availability (the probability of finding data online). We show that the problem of obtaining maximal availability while

minimizing redundancy is NP-hard; in addition, we perform an exploratory study on data placement strategies, and we investigate their performance in terms of redundancy needed and availability obtained. By performing a trace-based evaluation, we show that nodes with as few as 10 friends can obtain good availability levels.

Existing DOSN proposals try to solve this problem using heuristics which are agnostic to the various kinds of heterogeneity both in terms of end user resources as well as end user behaviors in such a system. In the second work on data availability, we argue that a pragmatic design needs to explicitly allow for and leverage on system heterogeneity, and provide incentives for the resource rich participants in the system to contribute such resources. To that end we introduce SuperNova - a super-peer based DOSN architecture. Super-peers can help (i) bootstrap new peers who are yet to have/find any friends by either providing them storage space, (ii) maintaining a directory of users, so that users can find friends in the network by name or interests, (iii) help peers find other peers to store their content in case they don't have adequate friends to do so, or if their friends are already overloaded. Users may want to become super-peers out of altruism (they want DOSNs to succeed), for the sake of the reputation (e.g., being an influential member for an interest based community) as well as potentially to monetize their special roles (e.g., run advertisements). While proposing the SuperNova architecture, we envision a dynamic system driven by incentives and reputation. However, investigation of such incentives and reputation, and its effect on determining peer behaviors is a subject of our future study. In this work we instead investigate the efficacy of a super-peer based system at any time point (a snap-shot of the envisioned dynamic system), that is to say, we try to quantify the performance of SuperNova system given any (fixed) mix of peer population and strategies.

The third problem of finding top-k nodes in a network is closely related to above two problems. Users can use top-k node information to select neighbors for disseminating

information or for requesting nodes to replicate their data. The decentralized settings make the problem of finding top-k node as non-trivial. Like the solution for information dissemination and availability, the algorithm to find top-k nodes uses local information only. The algorithm explores the social links and is completely distributed and decentralized in its approach for solving the problem. These attributes of the approach makes it scalable.

In this thesis, we have made an attempt to solve three challenges, out of several challenges of a DOSN. Although seen as three different premises of a DOSN, the problems are closely related to each other. We have various future plans (enumerated in next section) with respect to our present work to make our present solutions more robust.

6.2 Future Directions

Our future work is briefly described as follows.

6.2.1 Information Propagation

Given that GoDisco++ gossips are carried out exclusively using social links, the task is relatively easier than other decentralized settings, but nevertheless, there are significant outstanding research challenges to be dealt with. We would like to improve this work in various ways.

- (i) We plan to apply GoDisco++ in various kinds of applications, such as information dissemination in peer-to-peer online social networks [4] - for example, for probabilistic publish/subscribe systems, contextual advertisement of products to sell to other users of a social network (C2C commerce/garage sale on online social network) or as a decentralized recommendation system - emulating similar recent functionalities in centralized online social networks.

- (ii) Integrate human behavioral models [47] with the network evolution models, to come up with theoretical framework of how various information spreads in online social networking communities. Such a work will be qualitatively analogous to say the preferential attachment model for small-world network formation.
- (iii) Improve the existing schemes in various manner.
 - (a) Devising self-tuning mechanisms to choose the various system and algorithm parameters like X , p_1 , α , β and γ , as well as refining the interest categories I themselves to better describe user interests, and thus improve quality of dissemination based on such descriptions.
 - (b) Currently, there is no feedback mechanism in the system through which a node can establish about the false positive and false negative classification of the nodes. Introduction of such a feedback system from neighbors can improve the performance.
 - (c) Use more sophisticated techniques to identify information agents or carry out damping, or explore the network to find isolated communities, e.g., using random walkers in sparse networks - which is expected to improve dissemination.
 - (d) Incorporate various security mechanisms and disincentives for antisocial behavior. The current work assumes well behaved nodes. For example, message sources choose correct description and prudent parameters, intermediate nodes carry out the gossip task and do not tamper the content, etc. Nodes may choose to misbehave - act selfishly or proactively launch denial of service attacks. We need mechanisms to discourage as well as mitigate the effects of such misbehaviors. Given that GoDisco gossips are carried out exclusively using social links, the task will be relatively easier than other decentralized settings, but nevertheless, there are significant outstanding research challenges to be dealt with.

6.2.2 Data availability in DOSN

6.2.2.1 Empirical Study

We plan to continue working on this topic in following future directions :

- (i) One direction would be to explore the impact of limited storage resources on nodes.
- (ii) Combining both global optimization techniques and interaction of selfish data placement strategies is one of the strategies we would like to explore.
- (iii) We understand that the traces are sparse and doesn't reflect the partial real world traces. We understand exploring more extensive traces of an online social network would definitely give us rich insights and more confidence about the results.

6.2.2.2 SuperNova

Studying the dynamic role of reputation in changing population mixes, as well as designing and incorporating incentive mechanisms in SuperNova, in order to achieve desirable mix of user populations; refinement of the security mechanisms in the design; and incorporation of the SuperNova mechanisms in the PeerSoN initiative comprise our planned future work. Some prospective work includes:

- (i) Continuation of the preliminary architecture of Supernova. Introducing enhancements to cover various scenarios such as impolite departure of the nodes, network and power failure as well.
- (ii) Understanding the dynamics of the network by modeling the storage problem using game theory approach in a larger context (but limited), such as [63].
- (iii) Redesigning the algorithm for SuperNova by leveraging on dynamics of the network.

- (iv) As part of immediate future work, we plan to propose a trust and incentive based mechanism so that nodes can decide with high confidence about the neighbors for selecting storekeeping purpose.
- (v) In a DOSN, data has to be stored securely at storekeepers so that no malicious user can corrupt the data. We also need a mechanism so that nodes can collude to get information about the data being stored. To overcome all these issues we need a secure and robust mechanism. This is also part of our future work which we want to integrate with the DOSN architecture.
- (vi) Incorporating existing availability models like [137] for the nodes.

Along with analyzing the real traces and proposing a decentralized architecture for DOSN, recently, we have started working on a framework (called PriSM) for decentralized deployment of a federation of autonomous social networks (ASN). The individual ASNs are centrally managed by organizations according to their institutional needs, while cross-ASN interactions are facilitated subject to security and confidentiality requirements specified by administrators and users of the ASNs. Such decentralized deployment, possibly either on private or public clouds, provides control and ownership of information/flow to individual organizations. Lack of such complete control (if third party online social networking services were to be used) has so far been a great barrier in taking full advantage of the novel communication mechanisms at workplace that have however become commonplace for personal usage with the advent of Web 2.0 platforms and online social networks. PriSM provides a practical solution for organizations to harness the advantages of online social networking both in intra/inter-organizational settings without sacrificing autonomy, security and confidentiality needs.

6.2.3 Ranking

We have a multidirectional plan for future extensions of this work.

- We would like to apply DRank in various applications including collaborative browsing and search (COBS) [111].
- Integrate human behavioral models [47] with the network evolution models.
- Incorporate various security mechanisms and disincentives for antisocial behavior. Trust and incentives are two important factors which need to be considered. The current work assumes well behaved nodes. However, this may not be a fair assumption in some cases.

6.3 List of Publications

- Rajesh Sharma, Anwitaman Datta, *GoDisco++: A Gossip algorithm for information dissemination in multi-dimensional community networks*, Journal of Pervasive and Mobile Computing (PMC), Oct, 2012.
- Rajesh Sharma, Anwitaman Datta, *A tunable graph model for incorporating geographic spread in social graph models*, IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) 2012, Istanbul, Turkey.
- Rajesh Sharma, Anwitaman Datta, *SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks*, International Conference on Communication Systems and Networks (COMSNETS) 2012, Bangalore, India.
- Rajesh Sharma, Anwitaman Datta, *DRank: Decentralized ranking mechanism for semantic community overlays*, International Conference on Communication Systems and Networks (COMSNETS) 2012, Bangalore, India.

- Rajesh Sharma, Anwitaman Datta, *Decentralized information dissemination in multidimensional semantic social*, International Conference on Distributed Computing and Networking 2012, Hongkong.
- Rajesh Sharma, Anwitaman Datta, *An empirical study of availability in friend-to-friend storage systems*, International Conference on Peer to Peer Computing, IEEE P2P 2011, Kyoto, Japan.
- Anwitaman Datta and Rajesh Sharma, *GoDisco: Selective Gossip based Dissemination of Information in Social Community based Overlays*, In proceedings of International Conference on Distributed Computing and Networking (ICDCN'11), Bangalore, India, Jan 2011. [Received **Best Paper award** in Networking Track]
- Rajesh Sharma and Anwitaman Datta, *Distributed Social Computing Primitives*, presented in Doctoral Symposium at International Conference on Distributed Computing and Networking (ICDCN'11), Bangalore, India, Jan 2011.

Part IV

At the periphery of main research

Chapter 7

Appendix

In section 7.1 we describe our work which lies at the periphery of our main research. Later, in section 7.2 we provide all the acronyms being used in various chapters.

7.1 A tunable graph model for incorporating geographic spread in social graph models

Modeling and understanding social network structure has interested researchers from many backgrounds including social science, computer science, theoretical physics and graph theory. Notable models include [138] and [133] achieving graphs with power-law degree distribution using preferential attachment and small-world characteristics using randomized rewiring of a regular ring lattice respectively. In contrast to a body of follow-up research which refine upon these seminal works to better capture the graph structure and characteristics (such as improving clustering coefficient by considering social triads along with preferential attachment [139]), this work aims additionally to model the geographic spread in social networks. With increased mobility in our society as well as enhanced communication opportunities social networks are increasingly spread all over the globe. Synthetic graphs imitating real-world social network characteristics are often used for driving simulations for planning and decision support. Incorporating geographic spread can facilitate better infrastructure provisioning in distributed systems supporting

social and collaborative applications or model information of malware diffusion, word-of-mouth marketing, etc. The proposed model is tunable and modular. The model can be tuned to produce graphs with different geographic spread. The model is modular in the sense that existing geographic spread agnostic social network models can be plugged into our model to achieve desirable geographic spread in addition to other characteristics (such as degree distribution, clustering coefficient) that such a model would natively support.

7.1.1 Introduction

Graph models for social networks have been heavily researched for over a decade since the seminal works on *preferential attachment* induced network growth by Barabassi-Albert [138] and Watts-Strogatz's [133] randomized rewiring of regular ring lattice to build small-world graphs. It has also long been recognized that geographic processes have strong influence on the structure of the social graph. In his famous experiment Milgram [140] demonstrated that geographic co-location has a strong influence on connectivity among nodes.

Such information can be used when provisioning a globally distributed infrastructure for applications such as online social networking. For instance, depending on which users have what sort of spread of friends, a service provider may plan resources needed for data caching or design pre-fetching algorithms for better content distribution. It can play an even more critical role in the design of decentralized & peer-to-peer online social networks (DOSN)¹ and as well as in transportation networks to find optimal networks [141].

While the spatial aspect was initially considered as too complex to handle in the graph models [142], it was later incorporated in various graph models such as [143], [134] and

¹While there is no wide-scale DOSN deployment at present, there is a lot of interest in such infrastructure from the open-source and free-speech communities in addition to from academia.

[144]. Yook et al. studied the role of linear distance in networks [143]. Using theoretical models Kleinberg [134] and Watts [144] explain the presence of short discoverable paths by exploring geographic proximity.

The quest to find the relationship between individuals and their geographic distance also motivated the researchers to analyze the real datasets. Based on a large LiveJournal dataset, Liben-Nowell et al. observed that 70% of contacts were formed due to ‘geographic processes’², and many of one’s contacts in fact reside quite far away from one another [145]. The same views were expressed by Huberman and Adamic [146]. In another study using Nexus’ dataset from Stanford University [147], it was shown that - the probability of Nexus friends between two individuals living in the same dorm was about 5%. The chances of being a Nexus buddy between two individuals who were not staying in the same dorm was only 0.3% irrespective of the distance between the two individuals.

Using a mobile phone company’s dataset Lambiotte et al. found that the probability of acquaintance between two individuals decays inversely to the square of distance [148]. More formally, let $P(d)$ be the probability of acquaintance between two individuals, and d be the distance between two individuals. Then, $P(d) \sim d^{-\alpha}$, where $\alpha = 2$ (α is the decaying factor). Bernardo and Lada investigated the real dataset based on email interactions between HP employees [146]. They showed that the probability of acquaintance is well-fitted by a $1/d$ curve (that is $\alpha = -1$).

In a recent study [149] on location-based social networks, authors discover strong heterogeneity across users. They found out that probability of having a social connection between two individuals is $\sim d^{-\alpha}$, ($0.5 \leq \alpha \leq 1.0$). One reason of this heterogeneity could be due to different cultural and social aspects [150]. The difference in reporting of parameter α in various previous works [149], [145], [147], [148] may be due to the fact that many a times, access to real data containing all the relevant (for a specific purpose)

²Such as migration to a new region, befriending friends of friends from different regions and aging.

information may not be feasible or induce prohibitive costs. Furthermore, the size or characteristics of any accessible dataset may be limited, thereby limiting their usage. For instance, when designing a distributed system or algorithm, it may be desirable to test the same under different workloads. The ability to create synthetic datasets with realistic characteristics is thus important.

The different studies [149], [145], [147], [148] which have been done to understand the relationship between individuals and their geographic distance (or location) have not reached on a consensus for the value of α . In any case, they all establish one fact that geographic distance (or location) is an important aspect. Thus, it should not be ignored specially while proposing social graph models but has unfortunately been missed by most prominent previous works [139], [138], [133].

The motivation of this work is to provide a simple and modular mechanism to incorporate geographic spread characteristic to generate social graphs while also preserving other characteristics such as clustering coefficient, degree distribution, network diameter, etc. To that end we propose a graph generation model (Figure 7.1 gives an overview) which is inspired by some real life geographic processes. We call our approach modular because it is used along with existing social graph models. We will refer to whichever existing model we extend as the ‘native’ graph generation model.

The main contributions of this work are as follows:

- (i) The model is ‘tunable’ in the sense that by varying the different parameter values, different geographic distributions can be achieved.
- (ii) Social and geographic aspects are treated modularly/orthogonally w.r.t each other, thus, any native social graph model may be used to achieve the other social network properties. The social distribution phase (who is whose friend) is totally ‘modular’, as we can plug in any social model (for example Scale-free [138], small world [133], etc).

Figure 7.1: Node migration as well as creation of remote edges are leveraged in our social graph model. After migration, nodes may also discard some edges from the previous region. Such an aging process is also incorporated in our model, though it has not been shown in this figure.

- (iii) The model 'preserves' native network model properties.
- (iv) We define a new graph clustering metric namely 'Geographic-clustering compactness' which can be used to quantify the geographic distribution of out-region neighbors (that is neighbors which are from different locations/regions to that of the node) of the social networks.

We next summarize some background and related works. Then we introduce our graph model in Section 7.1.3 followed by its evaluation in Section 7.1.4. We draw our conclusions and outline some ongoing and future works in Section 7.1.5.

7.1.2 Related Work

Two seminal works in the area of social graph modeling are from Watts and Strogatz[133] and Barabasi and Albert [138]. Small world property with high clustering coefficient is

achieved in [133]. However this random rewiring based model has its own shortcomings. Some of the nodes can become isolated during rewiring and the degree distribution of the nodes is not scale-free, which is however typical in real social networks [151]. The preferential attachment based model in [138] allows incremental population growth and does not require the knowledge of total number of nodes in advance (unlike [133]) and produces graph which are scale-free. On the flip side, the clustering coefficient of such graphs is very low. Later works such as [139] propose approaches which can achieve high cluster coefficient and scale-free degree distribution. In [139] whenever a new node creates edges with the existing nodes in the system, edges are also introduced between these existing nodes with some probability. This approach probabilistically establishes social triad and thus boost the network's clustering coefficient.

Table 7.1: Geographical distribution phase parameters

Parameter	Definition
f_g	Determines the relative frequency of Geographical Distribution Phase.
f_m	Determines the relative frequency to call either (i) movement of nodes from one region to new or (ii) creating new edges with nodes from a different region.
f_f	The fraction of nodes a node forgets when it moves to new region.

The model presented by [133] also considers the spatial aspect by putting the nodes (in an order) on the ring lattice. Each node is connected with its x ‘adjacent’ neighbors and also some random shortcuts are introduced (by rewiring). However such a notion of distance does not capture the more nuanced nature of geographic regions.

Various theoretical work have been done to understand the relationship between the nodes and their geographic distance. Also, analyses of real datasets have been performed with same intensions, such as - internet [141], [143], [152], infrastructure networks [141],

Stanford university’s dormitories [147], email network of corporate employees [146] and recently on location based social networks [149].

Works such as [134] and [146] investigated the relationship between two individuals and their geographic distance. However, these two studies were done to understand the idea of information dynamics in large networks. In a recent work, using real dataset from three different location-based social networks, authors concluded that a ‘strong heterogeneity’ exists among users [149]. In comparison to all the above mentioned works, we have proposed a tunable and a modular graph generator model which considers the spatial aspect of the nodes.

Using LiveJournal’s dataset Liben-Nowell had observed that approximately 30% of friendships are independent of geographic location [145]. They also proposed a rank-based friendship model to describe the distribution of social relations in different geographic locations [153]. However they overlooked the typical social network graph characteristics, such as, power-law, cluster-coefficient etc. In this work while proposing the tunable socio-spatial graph generator we also preserved the social properties of a social network.

The work closest to ours is that of Goldenberg and Zheng [154] where each person in the network has a distribution over social interaction spheres termed as ‘contexts’. For example, a user will make friends from different places like gym, social gatherings (contexts) etc. where he/she visits. The more frequently a context is visited the more friends a user is going to make from that context. The links fade away with time for the contexts which are less visited. The explicit geographic spread of the nodes across regions and measurement of these cross region friends were not considered there.

7.1.3 The Social Graph Generation Model

Social processes have strong bearing on the structure of the resulting network. That is why many network generation models have imitated real life behaviors such as preferential attachment [138] or social triads [139]. Since social processes are naturally localized,

social networks bear similar geographic characteristics. Nevertheless, a significant portion of an individual's social network may be geographically distributed. This is increasingly the case with enhanced mobility with globalization as well as online interaction mechanisms which aid in establishing and maintaining long distance relations.

We try to capture the geographic distribution in social networks in the proposed model. The model itself is inspired by real life phenomena which have in addition geographic implications and interpretations (see Figure 7.1 for an overview). This includes the processes of *migration*, establishment of new links with socially proximate neighbors, e.g., *friends of friends (FoF)* even if geographically separated, and cessation of some links with time, i.e. *aging*.

There are many nuances for each of these processes that are hard to capture while keeping the network model simple. For instance people from a particular region may prefer to migrate to only some specific other regions. For example, people from India are more likely to migrate to Singapore than to China. Even though we ignore such details, preference in our model to migrate to regions which already have some friends or friends of friends leads to similar behaviors (to certain extent).

The model requires the number of regions (R) and an initial distribution p_d of population across these regions as input parameters. The total number of nodes for which the network is to be created is another input parameter. Depending on what native graph model we use determines whether this can be changed dynamically. For instance, if we use the model proposed in [133] then the population has to be predetermined, while if the model in [138] is used then the network can be grown incrementally and indefinitely.

7.1.3.1 Phase 1 - Social Distribution Phase (creation of social links):

In what we call a bootstrap phase, a very small constant number of nodes³ are assigned per region to form a network using the native graph model such as [133, 138, 139]. The

³We used 30 nodes per region in all the reported experiments. Variation of this number did not have any notable influence in the results.

rest of the nodes are assigned an initial region based on a random process where the probabilities are determined according to p_d . The nodes continue to join the respective local network following the same native model.

This phase is pluggable in the sense that any model can be used for creation the social graph⁴.

7.1.3.2 Phase 2 - Geographical Distribution Phase:

A logically separate phase induces the geographic distributions. We call this the geographic distribution phase (GDP). It may run simultaneously even while more new nodes are being added in the previous phase provided that the native model supports incremental network formation (e.g., [138, 139]) while it is run after the previous phase when that is not the case (e.g., [133]). A simulation parameter f_g determines the relative frequency of this phase with respect to the overall number of new node joins. Summary of the meaning of f_g as well as the next two variables that we introduced is provided in Table 7.1.

We imitate three real life phenomenons in this phase. An existing node from the system is chosen uniformly at random. This node then decides to either (i) move to another region and establish new links with nodes in the newly chosen region imitating migration or (ii) create new edges with nodes from a different region. The relative frequency of these two actions can be seen as another parameter f_m of our model. We fixed the number of new links that the node creates to be equal to half the number of links it previously had. When a node migrates, it also ‘forgets’ a fraction f_f of its older links with probability imitating (iii) an aging process.

The choice of new region to migrate to or establish new links itself is determined as follows. If the chosen node has any existing friend or friend-of-friend (FoF) in a different

⁴We used Barabassi model [138] and scale free plus social triad based model [139] in our experiments.

Figure 7.2: Out-location distribution in various scenarios

region, then such a region is chosen. If there are several such regions, then one is picked uniformly at random. More formally, if R_f are the total distinct out-location regions (that is regions other than the region of node itself) then probability of forming the new connection with any of these out-location region is $1/R_f$. While, if no such region exists, then a new region is determined randomly out of all the other regions. We note that it is conceivable to introduce many other model parameters to make more flexible decisions, however in the interest of simplicity we have opted not to do so.

7.1.4 Evaluation

We first describe the experimental setup, choice of parameters to perform the experiments with and the baseline scenarios against which we compare the results of our graph generation model.

7.1.4.1 Experimental setup

Native graph models: Our approach adds geographic distribution to existing graph generators. In the following experiments, we used the following ‘native’ models:

- (i) **Preferential Attachment (PA):** The Barabassi and Albert graph generator produces a scale-free network [138]. In this model every time a new node joins the network, the newcomer makes connection with m existing nodes chosen with probabilities proportional to the degree of the nodes (at that time).
- (ii) **Preferential Attachment with Social triad (PAST):** Holme and Kim [139] extended the [138] approach by introducing the notion of social triads in the graph (as explained in Section 7.1.2). The higher the probability of a social triad, the higher the clustering coefficient (CC) while the graph still retains a scale-free degree distribution. In our experiments the probability of social triad was chosen as 1. This means that whenever a new node A created edges with existing nodes B and C, an edge was introduced between B and C if no such edge existed.

Population and Regions: We perform our experiments with two kinds of population distributions (p_d) across different regions, viz. uniform and Zipf⁵.

Bootstrap phase: We experimented with the number of regions varied as 6, 12 (default), 18, 24, 30, 36 or 42. We first populated each region with 30 nodes and local graphs confined within individual regions were created using the native algorithm. After this, the rest of the nodes were introduced into the system by assigning them a region randomly based on the population distribution. A newcomer would then create new edges with the existing nodes of the region. The geographic distribution phase (GDP) was probabilistically invoked with relative frequencies f_g which corresponded to 10, 20, 30,

⁵<http://www.geohive.com>

40, 50 and 60% of the total number of nodes (joining) in the system. Within the GDP both (i) the migration of nodes or (ii) introduction of edges from across the regions were done with equal likelihood, i.e., $f_m = 1$.

Recall also that the number of new edges created in the new region to which a node has migrated is half of its existing number of edges. Likewise half of the existing number of edges are added when a node decides to establish cross-region edges. After migration we delete $f_f = 30\%$ of the existing edges from the older region of the migrated node.

7.1.4.2 Evaluation Metrics

We evaluate the geographic spread of the resulting networks based on the following metrics:

- (i) Out-location: It measures the percentage of neighbors from across regions for a particular node. Average out-location of the graph is the average of individual nodes' out-location.
- (ii) Geographical-clustering compactness: We also measured how scattered a particular node's neighbors are (across regions). For each node, we first measure the total distinct regions in which it has a neighbor while excluding its own (lets call it l_d). Out of the total distinct out-locations (l_d), we select l_p most populated regions such that $l_p = \lambda l_d$ (where $\lambda \in (0,1)$, in experiments $\lambda = 1/3$). We then measure the percentage of out-location nodes in the most populated l_p regions. Average GCC of the graph is the average of individual nodes' GCC. The higher is the percentage of the out-location nodes in the l_p regions the better is the Geographical-clustering compactness of the graph. As this signify that most of the out-locations friends are from few regions. This has already been established in various studies [145], [154], [147]

7.1.4.3 Results

- (i) **A Macroscopic View** We first provide a summary of the properties of the graphs generated by our approach under different scenarios (see Table i). The first column identifies the different settings that were used. The nomenclature in the first column is a combination of the native graph model (preferential attachment (PA) or preferential attachment with social triad (PAST)), number of regions (RX, where X could be 1, 6, 12, 18, 24, 30, 36 and 42), population distribution ('U' for uniform, 'Z' for Zipf and NA to represent not-applicable when there is a single region), relative percentage of using GDP (GNA when not applicable/applied or GY, where Y could be 10, 20, 30, 40, 50 and 60) and the total number of nodes (in thousands, ranging between 10–90K) for which the graph was generated. These are combined with hyphens '-'.

For example, 'PA-R1-NA-GNA-10K' means the native graph model used is preferential attachment (PA) and only single region is considered, thus population distribution or geographic distribution phase are not applicable (NA and GNA respectively) in this scenario. 10K means the graph was generated for 10 thousand nodes.

Table i lists the macroscopic properties such as average degree (Avg. Degree), clustering coefficient (CC), diameter, gradient, average out-location (Avg. OL) and average geographical-clustering compactness (Avg. GCC) values to characterize the graphs resulting from our model.

- (a) **Impact of GDP:** Intuitively, with the increase in the probability of using GDP (Rows: 3-5, 7-9, 12-14, 16-18, 39-42), the number of nodes participating in creating new links (across regions) also increases. Thus the average degree of the graph increases. Increase of cross region links also helps in the shrinking

of graph's diameter from 7 (Row 3, where GDP is called with the frequency of 10 %) to 5 (Row 41, where GDP is called with the frequency of 60 %). However, with increase in the frequency of GDP (note Row 3-5, 39-41) the average GCC decreases as the percentage of nodes movement increases (as they make new friends from more out-location regions) and thus the GCC decreases (compare Row 38 with Row 42 and Row 3 with Row 41).

When nodes create links across regions it has a positive impact on the clustering coefficient (CC) value (the percentage improvement is better in case of PAST as compared to PA). To see the effect of GDP on CC values, compare Rows: 3-5, 7-9, 12-14, 16-42 (when GDP is called) to Rows: 2, 6, 11, 15 (GDP not applicable).

- (b) **Impact of Variation of Nodes:** We also studied the impact of variation of nodes for both uniform and Zipf population distributions. We vary the number of nodes (Row: 5, 9, 19-24, 31-35) while keeping the number of regions 12 and invoking GDP 30% as frequently as new node joins. We observe that with an increase in the number of nodes, the average out-location (Avg. OL) value increases (although the impact is very low). However it has a negative impact on clustering coefficient. When the number of node increases the number of edges also increases, however, the number of social triads decreases resulting in decreased clustering coefficient. With increase in nodes value while keeping the regions and GDP frequency constant with a value of 12 and 30% respectively we notice slightly better value for average GCC (from 80 to 81 - Row 5,19-21, 31-35). As the nodes increase the probability of having neighbors in same out-location also increases. The improvement is negligible however positive.
- (c) **Impact of Variation of Regions:** We also varied number of regions for both uniform and Zipf population distributions, while keeping the nodes as

well as GDP constant (10K and 30% respectively). We observe that when we vary the number of regions (Rows: 5, 9, 25-30, 36-38) it has almost no effect on the out-location values (Avg. OL). There is slight improvement in geographic-clustering compactness (Avg. GCC) due to the fact that as the number of region increases the nodes per region decreases which increases the probability of social triads.

(d) **Impact of Zipf and Uniform Population Distribution:** The population distribution across different regions do have mixed impact on different macroscopic characteristics. However the average GCC metric values for zipf distribution are always lower than the corresponding uniform (compare Row 3-5 with 7-9, 12-14 with 16-18, 19-21 with 22-24, 25-27 with 28-30). The lower values in case of zipf could be because of the scale free distribution of population across different locations. When the GDP is called for a node, the probability of friends not being in same location is higher as there are large number of locations which are scarcely populated (and very few locations are densely populated). Thus out-locations neighbors are more scattered as compare to uniform population distribution. Thus it affects the average GCC value of the graph.

(e) **Native Graph Model's Characteristics are Preserved:** As most social networks follow power-law degree distribution [151], we also capture the gradient of the graphs to understand the effect of various strategies on the degree distribution. With the increased invocation of GDP, the graphs have shifted from ideal power law gradient values (which is between 2-3). However, the distribution has not completely lost the scale-free distribution. Overall we observe that while use of GDP does change the graph properties, these characteristics are broadly preserved. The values have either improved or deviated

Figure 7.3: Geographic-clustering compactness distribution in various scenarios

a little bit than what the corresponding native algorithm would produce. This is both acceptable and expected.

Table 7.2: Macroscopic characteristics for different scenarios

Row	Scenario	Avg. Degree	CC	Diameter	Gradient	Avg. OL	Avg. GCC
1	PA-R1-NA-GNA-10K	7.9968	0.004395	6	-2.14	N/A	N/A
2	PA-R12-U-GNA-10K	8.1816	0.0319	N/A	-2.61	0	N/A
3	PA-R12-U-G10-10K	8.6526	0.0359	7	-2.27	6.24	87
4	PA-R12-U-G20-10K	9.3284	0.054	7	-1.95	12.48	84
5	PA-R12-U-G30-10K	10.2396	0.0707	7	-1.82	18.93	80
6	PA-R12-Z-GNA-10K	8.1816	0.0305	N/A	-2.65	0	N/A
7	PA-R12-Z-G10-10K	8.6858	0.03865	8	-2.19	6.78	81
8	PA-R12-Z-G20-10K	9.3928	0.0608	7	-1.86	13.41	78
9	PA-R12-Z-G30-10K	10.3174	0.0677	7	-1.73	18.77	76
10	PAST-R1-NA-GNA-10K	0.2215	0.43	5	-1.65	N/A	N/A
11	PAST-R12-U-GNA-10K	9.94	0.44028	N/A	-1.40	0	N/A
12	PAST-R12-U-G10-10K	11.7672	0.4308	11	-1.41	8.88	91
13	PAST-R12-U-G20-10K	14.6082	0.4359	9	-1.47	15.16	88
14	PAST-R12-U-G30-10K	111.3084	0.5034	9	-0.55	18.96	87
15	PAST-R12-Z-GNA-10K	9.94	0.44671	N/A	-1.52	0	N/A
16	PAST-R12-Z-G10-10K	11.6578	0.4394	11	-1.41	8.78	86
17	PAST-R12-Z-G20-10K	36.7838	0.4719	9	-1.06	14.03	86
18	PAST-R12-Z-G30-10K	157.766	0.5129	8	-0.55	17.40	86
19	PA-R12-U-G30-20K	10.3471	0.05458	7	-1.71	18.95	80
20	PA-R12-U-G30-30K	10.2199	0.04873	7	-1.88	18.99	80
21	PA-R12-U-G30-40K	10.2224	0.03766	7	-1.86	19.00	80
22	PA-R12-Z-G30-20K	10.1561	0.05929	7	-1.76	18.84	76
23	PA-R12-Z-G30-30K	10.2556	0.05705	7	-1.81	18.92	76
24	PA-R12-Z-G30-40K	10.2569	0.03763	7	-1.84	19.22	74
25	PA-R6-U-G30-10K	10.089	0.0735	6	-1.78	18.95	74
26	PA-R18-U-G30-10K	10.222	0.07789	7	-1.80	18.91	82

Continued on next page

Table 7.2 – Continued from previous page

Row	Scenario	Avg. Degree	CC	Diameter	Gradient	Avg. OL	Avg. GCC
27	PA-R24-U-G30-10K	10.183	0.07968	7	-1.93	18.73	83
28	PA-R6-Z-G30-10K	10.2048	0.0553	6	-1.72	18.77	71
29	PA-R18-Z-G30-10K	10.235	0.06706	7	-1.72	18.77	78
30	PA-R24-Z-G30-10K	10.111	0.07607	7	-1.92	18.08	81
31	PA-R12-U-G30-50K	10.213	0.04105	7	-1.94	19.0017	80
32	PA-R12-U-G30-60K	10.163	0.0399413	7	-1.87	19.1856	80
33	PA-R12-U-G30-70K	10.1774	0.034938	7	-1.9484	19.484	81
34	PA-R12-U-G30-80K	10.2155	0.03774	7	-1.9362	19.519	81
35	PA-R12-U-G30-90K	10.2508	0.0309195	7	-1.9890	19.817	81
36	PA-R30-U-G30-10K	10.3624	0.090606	7	-1.9301	18.1107	83
37	PA-R36-U-G30-10K	10.2632	0.101644	7	-1.4846	18.505	83
38	PA-R42-U-G30-10K	10.3348	0.099008	6	-2.0561	18.94	83
39	PA-R12-U-G40-10K	11.5908	0.0887	7	-1.6317	24.276	77
40	PA-R12-U-G50-10K	13.8478	0.13527	6	-1.5472	30.1314	73
41	PA-R12-U-G60-10K	19.2246	0.27098	5	-1.3308	36.2863	68
42	PA-R42-U-G60-10K	17.2890	0.20107	6	-1.5811	34.8462	70

(ii) Microscopic View Figure 7.2 and 7.3 gives a microscopic view of the resulting graphs.

- (a) **Out-location :** To capture the microscopic information we divide the out-location values into three ranges: $[0-20)$, $[20-40)$ and $[40-100]$ corresponding to the percentage of nodes whose out-location neighbors fall on these ranges. We consider the following two naive graph models as baselines. We first create a graph assuming a single region using the native graph model (with $n=10000$). After the creation of the whole graph, we assign the nodes to different regions by two different approaches. In the first baseline, nodes are assigned different regions uniformly at random (U). In the second baseline we assign the locations based on friends-of-friends (FoF). For a particular node, if its friends have already been assigned a location, then a FoF mechanism is used probabilistically⁶. Also for this node its new location is picked uniformly at random from the different locations in which its friends are residing. Otherwise the location is assigned uniformly at random.

As shown in an earlier study [145] the out-location friends constitute on an average 30% of value. However, in case of the baseline approaches (first 10 bars in Figure 7.2) the out-location friends are almost 90% or above (which is not realistic) except in the case of PA-FOF-100. In ‘PA-FOF-100’ approach, FOF method is called with probability 1 (i.e., 100% of the time) to assign locations for nodes belonging to a graph generated by PA model. Although the value in PA-FOF-100 is better (compare to other baseline scenarios) we note that only 4 out of the 12 locations got populated at all. In other approaches all the 12 regions were populated, but resulted in extremely high and unrealistic out-location values.

⁶The probabilities of calling the FoF mechanism we experimented were 1, 0.9, 0.8 and 0.7.

In comparison to the baselines, in our approach (after the first 10 bars in Figure 7.2) the out region values (around 70% of nodes are having out-region friends in the range of [0-40%]), which is more realistic and somehow closer to what has already been observed in analysis of LiveJournal dataset [145].

(b) **Geographical-clustering compactness (GCC):** The average GCC gives a macroscopic view of the graph (Table i). However, this mechanism can also be used for a microscopic analysis of the distribution of the out-location neighbors (for each node as well as for the graph as whole). For a microscopic analysis, for each node, we categorize the regions having out-location neighbors into following four types.

- i. The top 10% most populated regions.
- ii. The next 15% of the most populated regions.
- iii. The next 25% of the most populated regions.
- iv. The least 50% of the most populated regions.

We then measure the percentage of the out-location nodes in each of the above four categories for all the variations (see Table 1, except where GDP is not applicable that is Row 1-2, 6, 10-11, 15). Figure 7.3 gives a detail macroscopic view for the different graph settings. With the increase in frequency of call to GDP the percentage of nodes in top 10% decreases from 87% (PA-R12-U-G10-10K) to 77% (PA-R12-U-G60-10K). This is due to the fact that as the frequency of GDP increases, the out-location friends are more scattered. Thus the percentage of out-location friends in top 10% also decreases. The most interesting and noticeable point is that in each of the graph settings the top 10% populated regions are having around 70% of the out-location nodes which is what was observed in [145]. The results in Figure 7.3 establish the fact that graph model can produce real-world scenarios.

By varying the percentage of times GDP is invoked, we can achieve a wide range of out-location values for different numbers of locations and node populations. This shows that our model (where we deliberately chose to use few parameters, and fixed many other possible ones in order to keep things simple) is nevertheless ‘tunable’ (while preserving graph characteristics) and can produce different geographic spread characteristics.

7.1.5 Conclusion

This work proposes a two-phased social graph generation model where the first phase uses any existing approach for nodes to join a local network, while the second one incorporates geographic spread by imitating some social processes that occur in real life to interconnect the localized networks. While there are several possible design choices in the proposed model which may be parameterized, for the sake of simplicity as well as space constraint we have only explored one aspect, namely the relative frequency of the second phase with respect to the first one, while keeping other design choices fixed. Even then, a rich gamut of geographic spread could be achieved by our model, while preserving other network characteristics as achieved by the ‘native’ model used in the first phase.

The proposed approach did not explicitly model richer geographic phenomenon such as preferential linkages with some specific regions, co-location of distant neighbors, etc. Likewise modeling temporal characteristics (such as who is online when) may provide better way to study information or malware spread online, resource planning, etc. Our plan for future is to carry out more comprehensive experiments with the current model and to extend it to better embrace a richer set of spatiotemporal characteristics in social graphs.

7.2 Acronym Table

In Table 7.3 we provide all the acronyms used in various chapters.

Table 7.3: Acronyms

Acronym	Full Name
DHT	Distributed Hash Table
DOSN	Decentralized Online Social Network
DTN	Delay Tolerant Network
F2F	Friend to Friend
IP	Initial Phase
MMOG	Massively Multiplayer Online Game
OSN	Online Social Network
P2P	Peer-to-peer
TCP	Take Care Phase
Vis	Virtual Individual Server

References

- [1] Facebook Statistics. <http://www.facebook.com/press/info.php?statistics>.
- [2] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM, 2005.
- [3] Danah M. Boyd. Facebooks privacy trainwreck. 14(1), 2008.
- [4] Sonja Buchegger, Doris Schiöberg, Le Hung Vu, and Anwitaman Datta. Peer-son: P2p social networking - early experiences and insights. In *Proceedings of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys*, 2009.
- [5] Amre Shakimov, H. Lim, Landon P. Cox, and Ramon Caceres. Vis-à-vis:online social networking via virtual individual servers. Technical report, Duke University, May 2008.
- [6] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. In *Communications Magazine, IEEE*, 2009.
- [7] linkedInstats. <http://press.linkedin.com/about>.
- [8] Twitter Statistics. <http://teknophilia.wordpress.com/2011/01/27/twitter-statistics>.

REFERENCES

- [9] Beacon. <http://en.wikipedia.org/wiki/facebookbeacon>.
- [10] MarketingLand. <http://marketingland.com/google-now-censoring-borderline-offensive-google-images-2287>.
- [11] Forbes. <http://www.forbes.com/sites/matthewherper/2011/07/13/did-reddit-discover-silent-censorship-on-google-plus/>.
- [12] Techcrunch. <http://techcrunch.com/2012/05/05/facebooks-positive-comment-policy-irrelevant-inappropriate-censorship/>.
- [13] Sonja Buchegger and Anwitaman Datta. A case for P2P infrastructure for social networks - opportunities and challenges. In *Proceedings of WONS 2009, The Sixth International Conference on Wireless On-demand Network Systems and Services*, Snowbird, Utah, USA, February 2-4, 2009.
- [14] Rammohan Narendula, Thanasis G. Papaioannou, and Karl Aberer. Privacy aware and highly available osn profile. In *19th IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE*, 2010.
- [15] Scholar. <http://scholar.google.com.sg/>.
- [16] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Ker-marrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2), 2003.
- [17] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2), 1999.
- [18] Abdelmajid Khelil, Christian Becker, Jing Tian, and Kurt Rothermel. An epidemic model for information diffusion in manets. In *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, MSWiM '02, pages 54–60. ACM, 2002.

REFERENCES

- [19] Rajesh Sharma and Anwitaman Datta. GoDisco++: A gossip algorithm for information dissemination in multi-dimensional community networks. *Pervasive and Mobile Computing*, October 2012.
- [20] Rajesh Sharma, Anwitaman Datta, Matteo Dell’Amico, and Pietro Michiardi. An empirical study of availability in friend-to-friend storage systems. In *Peer-to-Peer Computing*, pages 348–351, 2011.
- [21] Rajesh Sharma and Anwitaman Datta. Supernova: Super-peers based architecture for decentralized online social networks. *COMSNETS*, 2012.
- [22] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.
- [23] Anwitaman Datta and Rajesh Sharma. Godisco: Selective gossip based dissemination of information in social community based overlays. In *ICDCN*, 2011.
- [24] Rajesh Sharma and Anwitaman Datta. Decentralized information dissemination in multidimensional semantic social overlays. In *ICDCN’12*, pages 473–487, 2012.
- [25] Anwitaman Datta, Sonja Buchegger, Le Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Springer’s Handbook of Social Networks*, 2010.
- [26] Wikipedia. [http : //en.wikipedia.org/wiki/distributed_computing/citrefdolev2000](http://en.wikipedia.org/wiki/distributed_computing/citrefdolev2000).
- [27] Napster. www.napster.com.
- [28] Gnutella. <http://www.gnutellaforums.com>.
- [29] BitTorrent. www.bittorrent.com.

REFERENCES

- [30] Nicolas Liebau, Konstantin Pussep, Kalman Graffi, Sebastian Kaune, Eric Jahn, André Beyer, and Ralf Steinmetz. The impact of the p2p paradigm. In *In Proceedings of Americas Conference on Information Systems 2007*, Aug 2007.
- [31] SetiAtHome. <http://setiathome.ssl.berkeley.edu>.
- [32] S. Androutsellis-theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. In *ACM Computing Surveys*, pages 335–371, 2004.
- [33] Freenet. <http://freenetproject.org>.
- [34] Kazaa. <http://www.kazaa.com>.
- [35] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM 01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA: ACM, 2001*, pages 149–160, 2001.
- [36] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg. London, UK:Springer-Verlag*, pages 329–350, 2001.
- [37] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM 01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA: ACM*, pages 161–172, 2001.
- [38] K. Aberer, P. Cudre-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Puceva, and R. Schmidt. P-grid: a self-organizing structured p2p system,. 32(3):29–33, 2003.

- [39] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas. A survey of application-layer multicast protocols. 9(3), 2007.
- [40] Anwitaman Datta, Silvia Quarteroni, and Karl Aberer. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks. *Semantics of a Networked World*, 2004.
- [41] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 26(5), June 2008.
- [42] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, 2008.
- [43] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. *World Wide Web*, 8(2), 2005.
- [44] Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkinsi. Information diffusion through blogspace. In *WWW*, 2004.
- [45] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *WWW*, 2009.
- [46] Andrea Apolloni, Karthik Channakeshava, Lisa Durbeck, Maleq Khan, Chris Kuhlman, Bryan Lewis, and Samarth Swarup. A study of information diffusion over a realistic social network model. In *International Conference on Computational Science and Engineering*, 2009.
- [47] Xiaodan Song, Ching-Yung Lin, Belle L. Tseng, and Ming-Ting Sun. Modeling and predicting personal information dissemination behavior. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005.

- [48] Devavrat Shah. Gossip algorithms. *Found. Trends Netw.*, 3(1), 2009.
- [49] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2000.
- [50] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*. ACM, 1987.
- [51] J. A. Patel, I. Gupta, and N. Contractor. Jetstream: Achieving predictable gossip dissemination by leveraging social network principles. In *Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications*, 2006.
- [52] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. T-man: Gossip-based fast overlay topology construction. *Comput. Netw.*, 53(13), 2009.
- [53] Ymir Vigfusson, Ken Birman, Qi Huang, and Deepak P. Nataraj. Go: Platform support for gossip applications. *P2P*, 2009.
- [54] Lia C. Rodrigues, Clodoaldo A. M. Lima, Pedro P. B. de Oliveira, and Pollyana N. Mustaro. Clusterization of an online game community through self-organizing maps and an evolved fuzzy system. In *Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 02*, pages 330–334. IEEE Computer Society, 2008.
- [55] Pang Sheng and Chen Changjia. Community Analysis of Social Network in MMOG. *Int'l J. of Communications, Network and System Sciences*, 2010.
- [56] *An Online Community as a New Tribalism: The World of Warcraft*, January 2007.

REFERENCES

- [57] John Kubiawicz, Davic Bindel, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *ACM SIGARCH Computer Architecture News*, volume 28, pages 190–201, 2000.
- [58] Wuala. www.wuala.com.
- [59] Dinh Nguyen Tran, Frank Chiang, and Jinyang Li. Friendstore: cooperative online backup using trusted nodes. In *SocialNets '08: Proceedings of the 1st Workshop on Social Network Systems*, 2008.
- [60] B. Knutsson, Honghui Lu, Wei Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.
- [61] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker¹, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M. van Steen¹, and H.J. Sips. Tribler: A social-based peer-to-peer system. In *5th International workshop on P2P Systems*, 2006.
- [62] Anwitaman Datta. SoJa: Collaborative Reference Management Using A Decentralized Social Information System. In *6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2010.
- [63] Krzysztof Rzadca, Anwitaman Datta, and Sonja Buchegger. Replica Placement in P2P Storage: Complexity and Game Theoretic Analyses. In *ICDCS*, 2010.
- [64] Frédéric Giroire, Julian Monteiro, and Stéphane Pérennes. P2p storage systems: How much locality can they tolerate? In *LCN*, pages 320–323, 2009.
- [65] Michal Kryczka, Ruben Cuevas, Carmen Guerrero, Eiko Yoneki, and Arturo Azcorra. A first step towards user assisted online social networks. In *Social Network Systems*, 2010.

- [66] Rammohan Narendula, Thanasis G. Papaioannou, and Karl Aberer. Towards the realization of decentralized online social networks: An empirical study. In *ICDCS Workshops*, pages 155–162, 2012.
- [67] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.
- [68] Peixiang Zhao, Jiawei Han, and Yizhou Sun. P-rank: a comprehensive structural similarity measure over information networks. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, 2009.
- [69] Michael Brinkmeier. Pagerank revisited. *ACM Trans. Internet Technol.*, 6:282–301, August 2006.
- [70] Shuming Shi, Jin Yu, Guangwen Yang, and Dingxing Wang. Distributed page ranking in structured p2p networks. In *In ICPP*, pages 179–186, 2003.
- [71] Uwe Thaden, Wolf Siberski, Wolf tilo Balke, and Wolfgang Nejdl. Top-k query evaluation for schema-based peer-to-peer networks. In *In Proceedings of 3rd International Semantic Web Conference (ISWC)*, pages 137–151, 2004.
- [72] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In *In PODC*, pages 206–215, 2004.
- [73] Mark Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst*, 23:219–252, 2005.
- [74] Xiao Bai, Marin Bertier, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. Gossiping personalized queries. In *Proceedings of the 13th International Conference on Extending Database Technology, EDBT '10*, pages 87–98. ACM, 2010.

- [75] J. Douceur. The sybil attack. 2002.
- [76] R. Dingledine. The free haven project: Design and deployment of an anonymous secure data haven. In *Master's thesis*. MIT., 2000.
- [77] D. Hughes, G. Coulson, and J. Walkerdine. Freeriding on gnutella revisited: the bell tolls? In *In IEEE Distributed Systems Online*. IEEE, 2005.
- [78] B. Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. In *In Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002.
- [79] Eytan Adar and Bernardo A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.
- [80] Daniel Hughes, Geoff Coulson, and James Walkerdine. Free riding on gnutella revisited: The bell tolls? *IEEE Distributed Systems Online*, 6(6), June 2005.
- [81] T. Grandison, T. Gr, and M. Sloman. A survey of trust in internet applications. 2000.
- [82] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, page 6007, Washington, DC, USA, 2000. IEEE Computer Society.
- [83] J. Shneidman, D. Parkes, and L. Massoulie. Faithfulness in internet algorithms. In *Proc. SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS'04), Portland, OR, USA*, September 2004.
- [84] Karthik Tamilmani, Vinay Pai, and Alexander Mohr. Swift: A system with incentives for trading. In *Proceedings of the 2nd Workshop on the Economics of Peer-to-peer Systems.*, June 2004.

- [85] Rameez Rahman, Tamás Vinkó, David Hales, Johan Pouwelse, and Henk Sips. Design space analysis for modeling incentives in distributed systems. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pages 182–193. ACM, 2011.
- [86] Tsuen-Wan Johnny Ngan, Dan S. Wallach, and Peter Druschel. Incentives-compatible peer-to-peer multicast. In *Proceedings of the 2nd Workshop on Economics of Peer-to-Peer Systems.*, June 2004.
- [87] Ahsan Habib and John Chuang. Incentive mechanism for peer-to-peer media streaming. In *Proceedings of International Workshop on Quality of Service 2004(IWQOS04)*, pages 171–180, June 2004.
- [88] Guan-Yu Huang, Shun-Yun Hu, and Jehn-Ruey Jiang. Scalable reputation management for p2p mmogs. In *Proceedings First International Workshop on Massively Multiuser Virtual Environments*, 2008.
- [89] Chiranjeev Buragohain, Divyakant Agrawal, and Subhash Suri. A game theoretic framework for incentives in p2p systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 48–56, 2003.
- [90] Alberto Blanc, Yi-Kai, and Amin Vahadat. Designing incentives for peer-to-peer routing. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.
- [91] Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *In Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 294–301. ACM Press, 2002.
- [92] Tomasz Kaszuba, Krzysztof Rzadca, and Adam Wierzbicki. Discovering the most trusted agents without central control. In *Proceedings of the 2008 IEEE/IFIP*

REFERENCES

- International Conference on Embedded and Ubiquitous Computing - Volume 02*, EUC '08, pages 616–621, Washington, DC, USA, 2008. IEEE Computer Society.
- [93] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Proceedings of the 2nd Workshop on Economics of Peer-to-Peer Systems.*, June 2004.
- [94] José De S.P. Guedes, Viviane Torres Da Silva, and Carlos J. P. De Lucena. A reputation model based on testimonies. In *Proceedings of the 8th international Bi conference on Agent-oriented information systems IV*, AOIS'06, pages 37–52, Berlin, Heidelberg, 2008. Springer-Verlag.
- [95] Karame Ghassan and Ioannis T. Christou. A secure hybrid reputation management system for super-peer networks. In *5th IEEE Consumer Communications and Networking Conference, 2008. CCNC 2008.*, 2008.
- [96] Yao Wang, Jie Zhang, and Julita Vassileva. Super-agent based reputation management with a practical reward mechanism in decentralized systems. In *In Proceedings of the IFIP WG 11.11 International Conference on Trust Management*, 2010.
- [97] Black Hat. <http://www.blackhat.com/html/bh-usa-08/bhusa-08-archive.html>.
- [98] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, October 2007.
- [99] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 551–560, New York, NY, USA, 2009. ACM.
- [100] Matthew M. Lucas and Nikita Borisov. Flybynight: mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, WPES '08, pages 1–8, New York, NY, USA, 2008. ACM.

REFERENCES

- [101] Saikat Guha, Kevin Tang, and Paul Francis. Noyb: privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 49–54, New York, NY, USA, 2008. ACM.
- [102] Benjamin Greschbach and Sonja Buchegger. Friendly surveillance – a new adversary model for privacy in decentralized online social networks. In *Current Issues in IT Security 2012, 5th interdisciplinary Conference, Freiburg, Germany, May 08-10, 2012. Proceedings*.
- [103] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: An online social network with user defined privacy. In *ACM SIGCOMM*, 2009.
- [104] Amin Tootoonchian, Kiran Kumar Gollu, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: social access control for web 2.0. In *Proceedings of the first workshop on Online social networks*, WOSN '08, pages 43–48, New York, NY, USA, 2008. ACM.
- [105] Luca Maria Aiello and Giancarlo Ruffo. Secure and flexible framework for decentralized social network services. In *Proceedings of SESOC '10: Security and Social Networking Workshop*, pages 594–599. IEEE Computer Society, 2010.
- [106] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, UK, 2002. Springer-Verlag.
- [107] Roxana Geambasu, Magdalena Balazinska, Steven D. Gribble, and Henry M. Levy. Homeviews: peer-to-peer middleware for personal data sharing applications. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 235–246, New York, NY, USA, 2007. ACM.

REFERENCES

- [108] Seok-Won Seong, Jiwon Seo, Matthew Nasielski, Debangsu Sengupta, Sudheendra Hangal, Seng Keat Teh, Ruven Chu, Ben Dodson, and Monica S. Lam. Prpl: a decentralized social networking infrastructure. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, MCS '10, pages 8:1–8:8, New York, NY, USA, 2010. ACM.
- [109] Vicent Cholvi, Pascal Felber, and Ernst W. Biersack. Efficient search in unstructured peer-to-peer networks. In *SPAA*, 2004.
- [110] Adriana Iamnitchi, Matei Ripeanu, Elizeu Santos-Neto, and Ian Foster. The small world of file sharing. In *The 23rd Conference of the IEEE Communications Society (InfoCom 2004)*, 2004.
- [111] Christian von der Weth, Anwitaman Datta, and Sally Ang. COBS: A tool for collaborative browsing and search on the web. In *ICME*, 2010.
- [112] Anwitaman Datta, Manfred Hauswirth, and Karl Aberer. Updates in highly unreliable, replicated peer-to-peer systems. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. IEEE Computer Society, 2003.
- [113] Santosh Kulkarni. Badumna network suite: A decentralized network engine for massively multiplayer online applications. In *Peer-to-Peer Computing*, 2009.
- [114] Giuliano Mega, Alberto Montresor, and Gian Pietro Picco. Efficient dissemination in decentralized social networks. In *Peer-to-Peer Computing*, pages 338–347, 2011.
- [115] Antoine Boutet, Davide Frey, Rachid Guerraoui, and Anne-Marie Kermarrec. Whatsup: News, from, for, through, everyone. In *Peer-to-Peer Computing*, pages 1–2, 2010.

REFERENCES

- [116] Anatol Rapoport. Spread of information through a population with sociostructural bias: I. assumption of transitivity. *Bulletin of Mathematical Biophysics*, 15, 1953.
- [117] M. Newman. Scientific collaboration networks: I. Network construction and fundamental results. *Phys. Rev. E*, vol. 64, 2001.
- [118] P. Gibbons C. Palmer and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *KDD*, 2002.
- [119] A. Datta. Peer-to-peer storage systems: Crowdsourcing the storage cloud. In *ICDCN tutorial*, 2010.
- [120] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel. Private communication through a network of trusted connections: The dark freenet. 2010. <http://freenetproject.org/papers.html>.
- [121] R.M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, page 85, 1972.
- [122] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *IPTPS*, 2006.
- [123] Diaspora. <http://www.joindiaspora.com/>.
- [124] OpenSocial. <http://www.opensocial.org/>.
- [125] NoseRub. <http://noserub.com/>.
- [126] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.
- [127] Yao Wang, Jie Zhang, and Julita Vassileva. Effective Web Service Selection via Communities Formed by Super-Agents. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2010.

REFERENCES

- [128] Twitter. <http://www.twitter.com>.
- [129] Yahoo Q & A. <http://answers.yahoo.com/>.
- [130] Marin Bertier, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec, and Vincent Leroy. The gossip anonymous social network. In *ACM/IFIP/USENIX 11th International Middleware Conference*, volume 6452 of *Lecture Notes in Computer Science*, pages 191–211. Springer, 2010.
- [131] Antoine Boutet, Davide Frey, Rachid Guerraoui, and Anne-Marie Kermarrec. Whatsup: News, from, for, through, everyone. In *In P2P*, 2010.
- [132] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46:668–677, 1999.
- [133] D. Watts and S. Strogatz. Collective dynamics of small-world networks. In *Nature*, vol. 393, 1998.
- [134] Jon Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.
- [135] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286:509–512, October 1999.
- [136] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, April 2002.
- [137] Zhongmei Yao, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Modeling heterogeneous user churn and local resilience of unstructured p2p networks. In *In ICNP*, pages 32–41, 2006.

REFERENCES

- [138] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286:509–512, October 1999.
- [139] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65, 2002.
- [140] S. Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.
- [141] M. T. Gastner and M. E. J. Newman. The spatial structure of networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 49(2):247–252, January 2006.
- [142] Batty M. Cities as small worlds. *Environment and Planning B: Planning and Design*, 28(5):637 – 638, 2001.
- [143] Soon-Hyung Yook, Hawoong Jeong, and Albert-László Barabási. Modeling the Internet’s large-scale topology. *Proceedings of the National Academy of Sciences*, 99(21):13382–13386, October 2002.
- [144] Duncan J. Watts, Peter Sheridan Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296:1302–1305, 2002.
- [145] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11623–11628, 2005.
- [146] Bernardo A. Huberman and Lada A. Adamic. Information dynamics in a networked world. In *Complex Networks, Lecture Notes in Physics*, pages 371–398. Springer, 2003.
- [147] Lada A. Adamic and Eytan Adar. How to search a social network. *Social Networks*, 27:2005, 2005.

REFERENCES

- [148] Renaud Lambiotte, Vincent D. Blondel, Cristobald de Kerchove, Etienne Huens, Christophe Prieur, Zbigniew Smoreda, and Paul Van Dooren. Geographical dispersal of mobile communication networks. *PHYSICA A*, 387:5317, 2008.
- [149] Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. Socio-spatial Properties of Online Location-based Social Networks. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.
- [150] Seong Eun Cho. Cross-cultural comparison of korean and american social network sites: exploring cultural differences in social relationships and self-presentation. *Unpublished doctoral dissertation, Rutgers University, New Brunswick, New Jersey.*, 2010.
- [151] A Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51:661–703, November 2009.
- [152] Gorman S P and Kulkarni R. Spatial small worlds: new geographic patterns for an information economy. *Environment and Planning B: Planning and Design*, 31(2):273 – 296, 2004.
- [153] Ravi Kumar, David Liben-Nowell, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. Theoretical Analysis of Geographic Routing in Social Networks. Technical report, June 2005.
- [154] A. Goldenberg and A. Zheng. Exploratory study of a new model of evolving networks. In *In ICML workshop on statistical network analysis*. SpringerVerlag, 2006.