SOCIAL CACHING AND VEHICULAR SOCIAL NETWORKS: TWO CONTRIBUTIONS TO THE THEORY AND PRACTICE OF ONLINE SOCIAL NETWORKS

BY LU HAN

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Department of Computer Science

Written under the direction of

Liviu Iftode

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2013

ABSTRACT OF THE DISSERTATION

Social Caching and Vehicular Social Networks: Two Contributions to The Theory and Practice of Online Social Networks

by Lu Han

Dissertation Director: Liviu Iftode

Recent years have witnessed an explosive growth of the Online Social Networks (OSNs), which serve as a fertile ground for research such as, examining individual and group behaviors, identifying information dissemination patterns, and building unconventional OSNs. This dissertation explores the design and implementation of two emerging OSNs, a distributed social network (DOSN) and a vehicular social network (VSN), both from theoretical and practical perspectives, with the goal of improving the performance and functionality of the systems.

Distributed OSNs have been proposed as an alternative to centralized OSNs. DOSNs neither have central repository of all user data, nor they impose control regarding how the data will be accessed, hence, enable users to keep control of their private data. One of the critical challenges in building DOSNs is the performance of the distributed social update dissemination. To address this, we propose Social Caches, nodes that cache updates of their friends in order to reduce the number of peer-to-peer network connections. We formulate social cache selection as an instance of the Neighbor-Dominating Set Problem, and propose a set of approximation, regression and heuristic social cache selection algorithms to solve it. Performance evaluation based on real social traffic data and five well-known social graphs shows that the proposed social caching mechanism can reduce P2P network connections by an order of magnitude.

In the second part of the dissertation, we introduce a new class of online social networks

called Vehicular Social Networks (VSNs), which connect drivers, commuters in particular, sharing the same road at the same time. As a first application over VSNs, we developed RoadSpeak, a scalable mobile Multiparty Voice Communication (MVC) system that allows drivers to communicate over automatically moderated chat groups based upon their interest. Each VSN chat group is regulated by a profile consisting of the triplet $\{$*time, location, interests*$\}$. Evaluation using benchmarks, field trials and simulations proves the feasibility of RoadSpeak, and demonstrates that it can support substantially larger groups of users compared with traditional MVC systems.

Acknowledgements

This dissertation was only possible through the contribution, encouragement, advice, and support from a large number of people. I humbly thank them all for their efforts. I wish to thank my advisor, Professor Liviu Iftode, and the members of my committee, Professor Badri Nath, Professor Vinod Ganapathy, Dr. Sanjeev Kumar (Facebook) and Professor Shan Muthukrishnan for their generosity in taking the time and energy to review and provide insightful suggestions to greatly improve this dissertation. I would also like to thank the many other Rutgers Computer Science faculty and staff members who encouraged me all the way throughout my Ph.D..

First and foremost, my deepest gratitude goes to my advisor, Professor Liviu Iftode, whose invaluable guidance, advice, efforts and suggestions in leading me through my Ph.D. procedure are difficult to overstate. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time, the guidance to recover when my steps faltered. I have received numerous treasures from Professor Iftode: his knowledge, vision, passion, consideration, inspiration, and sense of humor in both research and daily life. His enthusiasm for science and work have always been my first resource. It is simply impossible to enumerate all of the other valuable lessons I have learned while working with Professor Iftode. I cannot imagine having undertaken this journey without him, and feel immensely grateful and lucky to have the honor of working with him.

I have also had the good fortune to be able to work with Professor Badri Nath. He has been always available to discuss and explain problems. The discussions with him, have turned out to be very constructive, and have helped me sort out the technical details of my projects. Professor Nath's vision, direction, constant optimism, and advice have provided a wealth of benefits to me as I struggled to complete this work. His impact on my technical thinking and writing skills cannot be overstated.

I also would like to acknowledge Professor Shan Muthukrishnan. He is a listener as well

as an adviser, and I count myself very lucky to have worked with him. I am thankful for his insightful comments, constructive criticisms and thought-provoking encouragement at different stages of my research. He has been totally selfless in sharing with me, as he does with all his students, the fruits of his wisdom. The experience of working with him has a richness and depth far surpassing any ability of language to express, making it an honor, and a joy.

I would also like to thank Professor Vinod Ganapathy. It is regretful that I do not have a chance to work with him. However, I learned a lot from his vision, his insight into the research problems and his working style during group meetings. I also want to thank him for his feedback to my dissertation, my qualify exam, and all my practice talks.

I would like to thanks Professor Danfeng Yao (VirginiaTech University), from whom I learned how to be an female researcher. I also want to thank Magdalena Punceva, whose insights, hard work, and feedback helped me to build and evaluate the work. Furthermore, I want to thank Professor Tomasz Imielinski, Professor Apostolos Gerasoulis, Professor Abhishek Bhattacharjee, and Professor Louis Steinberg, It was a great pleasure to teach with them. Specially, Professor Tomasz Imielinski encouraged me a lot when I felt lost and hopeless.

I am grateful to the staff of the department for assisting me in many different ways, especially, Carol DiFrancesco, Maryann Holtsclaw, Regina Ribaudo, Komal Agarwal, Aneta Unrat, and Elkanah Rogers.

During my time at Rutgers, I have had the pleasure of working with many excellent people, especially the members of the DiscoLab with whom I have had the privilege of sharing my time, namely, Steve Smaldone, Pravin Shankar, Arati Baliga, Tzvika Chumash, Nishkam Ravi, Vancheswar Ananthanarayanan, Matthew Muscari, James Boyce, Mohan Dhawan, Shakeel Butt, Vivek Pathak, Gang Xu, Liu Yang, Rezwana Karim, Amruta Gokhale, Nader Boushehrinejadmoradi, Hongzhang Liu, and Wenjie Sha. Throughout the long journey all of these people have contributed their time, ideas, and opinions to the many projects, discussions, meetings, practice talks and papers that encompass the work and life of a graduate student. They made my Ph.D. life sparkling.

I want to thank my parents, Shaozhong Han and Xiurong Zhang, for their unfailing love, dedication, encouragement, and willingness to sacrifice. They give meaning to my life and

work. I can never repay them for the freedom that they have afforded me in allowing me to choose what I want to do. This dissertation would not exist but for their love.

Dedication

To my father, Shaozhong Han, and my mother, Xiurong Zhang.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Thesis

The thesis of the dissertation states that:

> Social caches can improve the performance of Distributed Online Social Networks. Vehicular Social Networks with Multiparty Voice Communication can help drivers to socialize while on the road.

This dissertation makes two contributions to the theory and practice of online social networks, Social Caching and Vehicular Social Networks.

Distributed Online Social Networks (DOSNs) have been proposed as an alternative to centralized Online Social Networks (OSN). Centralized OSN providers control individual user's data and associated policies on dissemination of data. Hence, user can hardly exert control of their personal data, resulting in potential privacy violations. In contrast to centralized OSNs, DOSNs do not have central repository of all user data, neither impose control regarding how user data will be accessed. Therefore, users can keep control of their private data and are not at the mercy of the social network providers. However, one of the main problems in DOSNs is how to efficiently disseminate social updates among peers. We introduce Social Caches as a way to alleviate the peer-to-peer connections in a distributed OSNs, and formulate social cache selection as the Neighbor-Dominating Set Problem. We propose a set of centralized and distributed algorithms to solve it, and evaluate their performance on five well known graphs. Performance evaluation based on real social traffic data shows that the algorithms can reduce P2P network connections by an order of magnitude.

We proposed the Vehicular Social Networks (VSNs) to extend the functionality of the current online social networks to reach the vehicular drivers. Vehicular Social Networks enables

the drivers who are physically driving on the same road segment, as the same time, to build virtual social communities for socialization purposed. VSN also facilitates the deployment of various applications on top of it. As the first application of VSN, we propose RoadSpeak, a scalable Multiparty Voice Communication (MVC) system that allows drivers to automatically join VSN groups along popular roadways. RoadSpeak achieves collision free Multiparty Voice Communication between users by utilizing automated moderation, buffering and flow control. We have implemented a RoadSpeak prototype, and built an MVC simulator to perform large-scale simulations that compare RoadSpeak with existing MVC systems. The results prove that RoadSpeak can support substantially larger groups of users compared with the traditional MCV.

We will first introduce the Online Social Networks, with a brief history in Section 1.2. We will present the current infrastructure of Online Social Networks in Section 1.3, with an emphasis on the *Core Service Infrastructure*, *Storage and Database Infrastructure*, *Application Service Infrastructure* and *Social Presence Infrastructure*. The challenges facing these types of infrastructure will be discussed in Section 1.4. We will summarize the contribution of the dissertation in Section 1.5, and the contributors in Section 1.6. Section 1.7 presents the organization of the dissertation.

1.2 Online Social Networks: the State of Art

Although human social networks have been studied for many years in the field of Sociology, Online Social Networks (OSNs) have only become popular as the topic of computer science research within recent years. In this section, we will briefly discuss the state of art of Online Social Networks, the range of user interactions they allow, and a brief history of the Online Social Networks.

1.2.1 Online Social Network Introduction

Although lacking a formal definition, an Online Social Network (OSN) [1] is considered to refer to any social network site that maps the relationships between individuals, indicating the ways

[1] Online Social Media, and Online Social Networking sites are interchangeable terms to represent an Online Social Network site. In this dissertation, we will user Online Social Networks.

in which they are connected ranging from casual acquaintance to close familial bonds.

For the purposes of this thesis, we use the term Online Social Network (OSN) to represent the system where:

(i) the basic entity is a user with a personal profile;

(ii) users are linked to other users or content items by means of *social relationships*;

(iii) users can navigate the social network by browsing the links and profiles of other users;

(iv) users can exchange information via *Social updates*.

Online Social Network sites provide opportunities for people to rendezvous, connect or collaborate. Although serving a number of different purposes, there are three primary roles that stand out as common features across all OSN sites. First, Online Social Network sites are used to *maintain, set up, strengthen, or weaken* the social ties between social network friends. They facilitate setting up connections between people based on shared interests, values, membership in particular groups (i.e., friends, professional colleagues). The sites allow users to "articulate and make visible their social networks, and communicating with people who are already a part of their extended social network" [18]. Second, Online Social Networks are used to *share* rich social contents in different formats. Third, Online Social Networks provide a platform for users to *explore* new, interesting content by searching, and recommending social content uploaded by other users/groups. These features of the OSN make it easier for users to find and communicate with any individual in the Online Social Networks; meet new people online; and connect with friends, family and acquaintances in real life.

1.2.2 A Brief History

The idea of social structure and social networks can be tracked back to the 1800's, and is much older than Computer Science itself. Systematic research exploring the patterns of human social networks and social ties linking individuals emerged in the 1930s. Social networking did not become an area of Computer Science until the 1990s, when the Internet became widely available.

Classmates.com [28] is believed to be the first OSN site, launched in 1995 as a site for users to reconnect with their previous high school classmates. It has a limited number of users

because it only allows those who attended the same school to be connected, but does not allow users to create links to users from outside their schools.

In 1997, the former SixDegrees.com was created, the first social networking site that allowed users to create links directly to other users without any restriction. It bore little resemblance to the SixDegrees.com in use today. The owner believes that the site failed because it was ahead of its time, since computers with Internet were not popular in 1997. In recent years, the Online Social Networks together with smart devices have became the most popular technique for connecting with other people. Most families have at least one computer or mobile device that connects people to the Internet all the time. In the early 2000s, a number of Online Social Networking sites were taking shape, such as Friendster [55], Match.com [98], LinkedIn [90], and MySpace [106]. Friendster developed from a friend-to-friend socialization site into the current online gaming site; Match.com is a popular online dating site; LinkedIn is used for professional connections; MySpace changed into a news feed web site. Many other OSN sites were also proposed at the same time, such as CyWorld [32], and Ryze [123]. A more complete history and analysis of the evolution of Online Social Networks can be found in papers by Boyd [19], [18].

Facebook, one of the most popular Online Social Networking sites, launched in February 2004. As of April 2012, Facebook was reported to have 900 million active users [49]. Recently, as a new format of Online Social Networks, Pinterest [119] provides an online pin-board that enables users who have the same interests to share pictures in the same categories.

People began to realize the profit that OSN can bring as the population of users increases. Today most websites are connected to one or more of the existing social, including such examples as multimedia content sharing sites (Flickr [53], YouTube [157], and Pinterest [119]), blogging sites (LiveJournal [95] and BlogSpot [12]), and professional networking sites (Branchout [20]). These sites have different goals but employ the common strategy of exploiting the social network to increase the number of users.

A complete and up-to-date list of the notable online social networking sites can be found at Wikipedia [92].

Jun 2011			
Country	**OSN 1**	**OSN 2**	**OSN 3**
Australia	Facebook	Twitter	LinkedIn
Canada	Facebook	Twitter	LinkedIn
France	Facebook	Twitter	Skyrock
Germany	Facebook	Twitter	Xing
Italy	Facebook	Badoo	Twitter
Russia	V Kontakte	Odnoklassniki	LiveJournal
Spain	Facebook	Tuenti	Badoo
United Kingdom	Facebook	Twitter	LinkedIn
United State	Facebook	Twitter	LinkedIn

Table 1.1: Online Social Network ranking for several countries.

1.2.3 Online Social Network Popularity

The most popular online social networking sites include Facebook [48], Twitter [143], and Google+ [61]. Besides these giant OSN sites that dominate the Internet in the U.S., a look around the world reveals more diverse social media sites in various languages, such as Kaixin [80], Weibo [150], and Renren [121] in China; Mixi [104] in Japan; Skyrock [128] has long dominated the French, Belgian, and Swiss social media market; while VKontakte [148] takes first place among Russian and Ukrainian social media sites. The most popular online social networking sites for several countries can be find in Table 1.1. For most of the countries, Facebook ranks first among all other online social networking sites.

Just as with other technology, Online Social Networking can be a very effective tool for connecting with people, exchanging information, and learning from each other. As in real life, connections between people do not just occur one-on-one, but form an entire network. These sites allow social network users to share photos, videos and information, organize events, chat, and even play games, and are very useful in sharing information and disseminating data. Online Social Networks are born to create efficient and convenient interaction between ONS friends.

1.2.4 Research in Online Social Networks

Research in the area of Online Social Networks did not become popular until 2005. Ever since than, OSN has attracted researchers from different computer science fields. Many new research directions focusing on Online Social Networks have been proposed, such as OSN analysis,

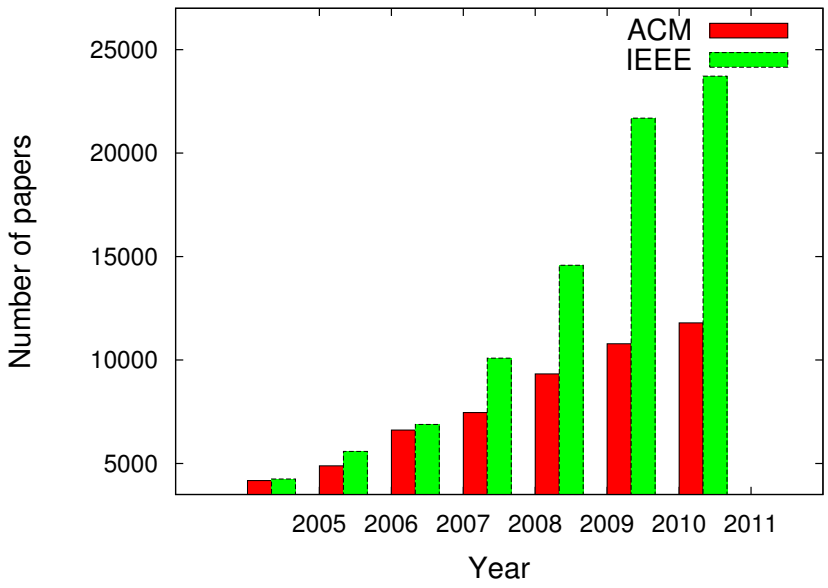


Figure 1.1: Number of papers published in the area of Online Social Networks.

OSN graph mining, OSN theory, OSN security, Mobile OSNs, etc. The research on OSNs quickly became an important and hot area in Computer Science. Figure 1.1 shows the number of papers published on "social networks" in ACM and IEEE each year from 2005 to 2011. The figure clearly illustrates that the number of papers in social networking area has increased dramatically every year. Note, the number of papers in the figure is based on a search using the key words "social" and "networks". Other papers, such as searches based on the keywords "social" and "computing", have not been included.

1.3 Online Social Network Infrastructure

In this section, we will discuss the research areas in Online Social Networks, utilizing the OSN Infrastructure as the chain. The general infrastructure of OSN is shown in Figure 1.2, and includes the *User Interface* layer, the *Application Service* layer, the *Social Presence* layer, the *Core Service* layer, and the *Storage and Database* layer.

The *User Interface* layer regulates how users interact with an Online Social Networking site, and how flexibly users can modify the appearance of their profile pages. The *Application Service* layer, collaborating with the *User Interface* layer, defines what applications can be supported by the OSN site. Application Service determines the *Application Domain(s)* of

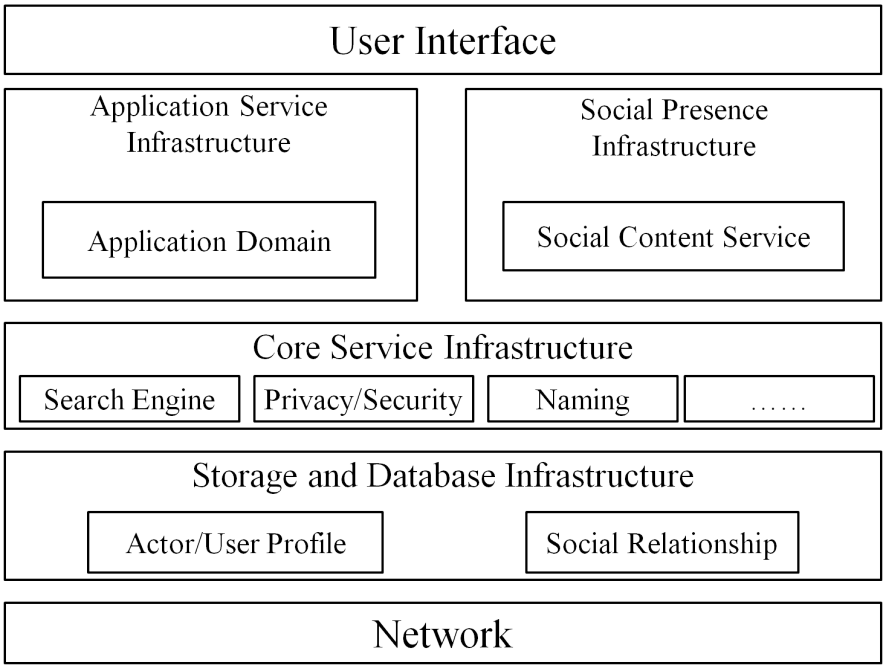


Figure 1.2: General infrastructure of Online Social Networks.

the OSN site. The Application Domains of the current Online Social Networks include but are not limited to: socialization applications, health applications, educational applications, online dating applications, etc. The *Social Presence* layer controls the quality of the social communication media between social network users. Different Online Social Networking sites support different social mediums, such as text, video, or audio, in a 2D or 3D presence. Video and 3D have a higher social presence and are more sociable and personal, while mediums such as text or 2D have low social presence. The *Core Service* layer includes the main services that an Online Social Network provides, such as *Searching*, *Privacy*, etc. The *Storage and Database* layer maintains the databases for user profiles, and the social relationships between users.

In the next sections, we will discuss in detail each of these layers and the current research topics within them.

1.3.1 Core Service

Privacy and Degree of Decentralization

The most important research question related to the Core Service Infrastructure layer is the privacy and security in Online Social Networks. Although highly successful, current OSNs

work by employing a model that sacrifices user privacy for availability and centralized control. Users upload their profiles and personal data to the OSN websites, thereby surrendering their personal relationships and profiles to the OSN providers. OSN providers and third-party applications operate on an individual user's private data, which should be only shared with a relatively small number of users. This centralized architecture of the OSNs runs counter to the nature of social networks and the privacy goals of OSN users, and has the following issues:

Disparate data sources and inconvenience. As we discussed in Section 1.2, sharing with friends on different OSNs involves separate registrations and providing friendships to multiple OSN services. Therefore, it is difficult for the users to keep track of their personal data, given that it is scattered across diverse providers and websites, depending on how the information is acquired and what kind of functions the users wish to perform. It is also very inconvenient for friends to try to learn all uploads of any given user. Using photo sharing as an example, when users want to share a picture with friends, they need to upload the picture to multiple OSN websites such as Path [116], Facebook, Twitter, Flickr [53] and more. The dispersal of their data is a negative incentive for users to take advantage of the OSN services. Furthermore, it is difficult, if not impossible, to move one's data if the user desires to migrate to other services.

Scalability. In a centralized OSN architecture, an application service provider runs central services to provide authentication and authorization, and to share users information with their friends. If the service becomes popular, it requires the service provider to scale out quickly.

Privacy. More important than the inconvenience, users lose the privacy of their relationships and data. OSNs that offer a broad range of services can acquire a lot of information about any individual who provide their detailed personal profiles. In most of the current OSNs, users only have very coarse-grain control over who can access their personal data. However, Gross et al. [62] show in their study that most users do not change the default privacy settings provided by the OSN services.

In the traditional Web, privacy is maintained by limiting data access and only granting access to authorized parties with user customized policies. In OSNs, by contast, data and identity are closely linked for sharing purpose, and are often visible to large groups of people. This makes it harder to preseve privacy and gives rise to privacy issues. Furthermore, OSN services

also experience most of the web security problems, such as phishing, spamming, identity theft, and malware. Phishing is much more effective in the "Social" environment. Things are even worse with Mobile Social Networks, which can detect a user's location by enabling location based services.

Anonymization, Decentralization, Privacy Setting Management, and Encryption technologies are the primary methods to solve the privacy problems in OSNs. Starting from basic anonymization by removing identifiable information from data, a set of *Anonymization* techniques [9], [7], [94], [69], [22] has been proposed so as to make users in the OSN dataset computationally indistinguishable from almost any other user. *Privacy Setting Management* enables user privacy by granting users more control over their privacy settings and making it easier to manage the settings. Baatarjav et al. [6] propose a system that automatically selects privacy settings according to user profile, which is used to predict expected user preferences. Maximilien et al. [99] manages privacy setting with a privacy risk score, which is calculated based on user's profile. *Encryption* is also used to protect a user's information from accessing by unwanted parties. FlyByNight [96] was proposed to encrypt critical parts of a user's profile using proxy cryptography to significantly mitigate the privacy risks. NOYB [63] preserves user privacy by encrypting user data to make the cipher text encoded to mimic legitimate data to the OSN providers who can operate on it, while only authorized users can decode and decrypt the data.

In this dissertation, we believe that the proposed unconventional OSN, *Distributed Online Social Network* (DOSN) architecture is a solution to address the privacy and security issues that were discussed earlier, and can significantly improve user privacy so that users can re-gain control of their data. Distributed online social networks enable users to host their personal data on the data storage that they choose, and free users from cross platform personal data maintenance. Distributed Online Social Networks, such as Diaspora [37], PeerSoN [21], PrPiI [126], and Safebook [31], were proposed since 2009; for a more exhaustive list of distributed OSN, we refer the reader to read the Wiki page [40].

The current deployments of DOSNs can be divided into three categories: *(i) Federation*, which requires that social network providers agree upon standards of operation in a collective

fashion. Federated social networks enable users to share their social contents in one OSN with friends from other OSNs. *(ii) OSNs over unstructured P2P underlay*, which have users' personal data distributed among multiple servers, and utilize a lookup server for bootstrapping functionalities [124], [113] and [126]. *(iii) OSNs over structured P2P underlay*, which utilize DHT underlays such as My3 [109] and PeerSoN [21].

1.3.2 Storage and Database

Online Social Network Analysis

The *Storage and Database Infrastructure* layer manages the database and storage that OSNs utilize to store actor profiles (any user, group, or organization is considered an actor), the social relationships between actors, and the social contents shared between actors. The rich social data available at the storage and database infrastructure layer enables the crawling of social networks, which gathers enough social information for one of the major research directions in Online Social Networks: the *Social Network Analysis*.

Social Network Analysis (SNA) is a visual and mathematical analysis of how people interact with each other, exchange information, learn, and influence one another. Social network analysis has a long history: the first SNA project began in 1987.

We first present the fundamental definitions in the social network analysis domain. An online social network is *a set of vertices* (points, actors, or agents) that connect with one another by *a set of edges* (relationships, or social ties). An Online Social Network can have few or many vertices, and one or more kinds of relationships between pairs of vertices. To build a useful understanding of a social network, a complete description of a pattern of social relationships is necessary. Social network analysis utilizes the SNA tools to manage these data, manipulating them to reveal patterns of social structure.

Given that Online Social Network analysis is a large area of research, we summarize major research topics as follows: *(i)* basic characteristics and properties of the graph; *(ii)* large scale Online Social Network data collection, crawling and anonymous; *(iii)* Online Social Network modeling and data mining; *(iv)* social community/cluster funding; *(v)* predictions in Online

Social Networking; and *(vi)* social issues including user data privacy, abuse of OSNs, reputation system and teenager problems. Publications that discuss different measurements and methodologies used in this area include: [83], [149], [125], and [130].

In this section, we will focus on the social network analysis metrics, measurements, and methodologies that relate to the topics in this dissertation, which primarily include: basic characteristics and properties of the graph; Online Social Network modeling and data mining; and social community funding. Furthermore, in this section, we will introduce some of the notations that we will use throughout this dissertation.

Basic Characteristics of an OSN graph

A social network is usually represented by $G = (V, E)$, where G represents the social network graph, V represents the vertex (node, actor) in the graph, and E represents the edges (social relationships). $|V|$ is used to represent the number of vertices, and $|E|$ the number of edges in the graph.

Online Social Network analysis typically begins with one node or a small group of nodes, then extends to the node's social relationships by using methods such as Breadth-First Search (BFS), random walk, or snowball. The following analysis levels are usually studied for any OSN site in the social network analysis, as well as in this dissertation.

Vertex level. The smallest unit in an OSN is a vertex in its social setting. An Egocentric network [97] is a vertex level network that contains only the vertex and its one-hop neighbors. Egocentric or ego network analysis often centers on network characteristics such as centrality, prestige and roles such as isolates and bridges, which we will discuss in Section 1.3.2.

Dyadic and Triadic level. A Dyad defines the social relationship between two vertices. Relationships in an OSN start with dyads. A triad in an OSN is formed by adding another vertex to a dyad, which also results in a set of three dyads.

A triad is transitive when there is an edge (social relationship) between any pair of dyads. The triad possibilities are shown in Figure 1.3: the left-most sub figure shows a potential transitive, the center part of the figure shows the intransitive, and the sub figure on the right shows a transitive triad. Research at this level usually concentrates on measurements such as balance

and transitivity, as well as social equality and tendencies toward reciprocity.

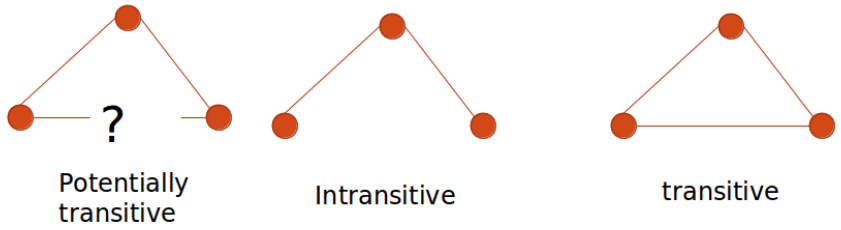


Figure 1.3: Level of transitive triad.

Sub-graph level. Sub-graph level research focuses on vertex distance and reachability, cliques, social community, cohesive subgroups, and group actions or behavior. A node, a dyad, a triad, and an egocentric graph are all special forms of sub-graph, but a sub-graph is not necessarily any one of those.

Clique. A clique in a graph consists of a subset of vertices such that every pair of vertices in the subset is connected by an edge, i.e., a vertex is directly tied to every other vertex. A k-clique of a graph is a clique of k vertices.

If a graph is a clique, it is also called a *complete graph*. A clique is a special graph structure in that: a dyad and a transitive triad are 2-clique and 3-clique, respectively. Furthermore, the egocentric network of any node in a complete graph is the graph itself.

Determing whether there is a k-clique in an OSN graph is NP-complete. Despite this hardness, many methods for finding cliques have been studied [57], [36], [8], [135]

Basic Metrics in Measuring OSN graph

The evaluation metrics used most in social network analysis includes:

Bridges. An edge is a bridge if deleting it would cause its endpoints to lie in different components of a graph.

Centrality. Centrality refers to a group of metrics that aim to quantify the ”importance” or ”influence” (in a variety of senses) of a particular vertex (or group) within a network [68], [93], [141]. Examples of common methods of measuring ”centrality” include betweenness centrality [149], closeness centrality, eigenvector centrality, alpha centrality and degree centrality. [114]

Clustering coefficient. The clustering coefficient is a very important measure of the likelihood that two neighbors of a vertex are connected themselves. A higher clustering coefficient indicates a greater 'cliquishness'. [23]

Cohesion. Cohesion is the degree to which nodes are connected directly to each other by cohesive bonds. Structural cohesion refers to the minimum number of members who, if removed from a group, would disconnect the group. [105]

OSN Data Mining

Date mining can be defined simply as "the extraction of valuable patterns that are hidden in large amounts of data" and its applications fall into two main classes: description and prediction. Description is the process of modeling the traits of existing objects; prediction is forecasting the object's behaviors from its known attributes. These two classes reflect the two research directions in SNA data mining: modeling and property predictions in OSNs, referring to the properties of three entities: the individual, the community (subgraph), and the entire graph. Individual properties include individual influence and social interaction patterns. Community properties include determing cliques and clusters. Entire graph mining includes graph growth, temporal boundaries of certain events.

Various mining techniques have been applied to different OSN sites for different purposes. Memon et. al. [102] presents the research in social networking analysis, and focuses on emerging trends in the discovery and analysis of communities and social activities, on network modeling, dynamic growth and evolution patterns, and on multi-agent based simulations, by utilizing machine learning approaches .

Meanwhile, business organizations, such as DataSift [34] and Gnip [60], aim to aggregate, filter, and manipulate OSN data into the analysis of social trends, social authorities, social influence, and social ties in real-time.

As an example, Klout [82] is a stand-alone service that identifies social authorities, who are the most influential persons in an OSN crowd. People are given a score in the range of 1-100, which "measures influence based on ability to drive action". They factor in how many people one reaches based on how many followers one has on multiple OSNs, and what kind of impact

one has on the egocentric network. People can use this service to find those that others would listen to in the community in the disaster evacuation or other severe situations.

OSN mining results can be utilized in many other directions. Companies do viral marketing by choosing the targets of their advertisements according to users' favorites and interests by mining their activities in OSNs [158]. Federal agencies are interested in mining Online Social Networks in order to identify geo-spatial breaking events, incidents or possible threats [51]. Although mining OSNs brings benefits for the users, crawling of personal data is considered a privacy threaten for normal users. In the next section, we will discuss the privacy and security issues facing the OSNs, and the methodologies proposed to solve these problems.

1.3.3 Application Service

Application Domains

In this section, we will discuss the Application Service Infrastructure layer, and focus on the state of the art of research on Application Domains, which are the key component of Application Service Infrastructure.

Although they were originally invented to serve the purpose of getting back and staying in touch with friends, Online Social Networks extend their applications to many other domains as they develops, such as career, health and medical care, education, and life style. Different researchers categorize Online Social Networks into various application domains based on different criteria. In this section, we will mainly focus on the following domains: *Career and business*, *Online Dating*, *Multimedia Sharing*, *and Consumer Advocacy*.

Career and Business

Applications for online social networking sites have extended toward career, businesses and brands, such as LinkedIn [90], MyWorkster [107], and Jobster [78]. LinkedIn, as one of the most popular career sites, provides a platform for setting up connections with professionals, efficient job searching, and hence the opportunity for a better career. It has over 150 million users in over 200 countries as of February, 2012 [91]. MyWorkster enables users to build up

social networks of classmates and alumni to connect for career opportunities. With career-oriented OSNs, employers can post job openings for candidates looking for available openings, so that users can target the correct person for obtaining job information.

The use of Online Social Networking services in an enterprise context also presents the potential to have a major impact on the world of business. Social networks connect people at low cost, which can be beneficial for small entrepreneurs and businesses looking to expand contacts and share knowledge. OSNs for business include Biznik [11],Cofoundr [30], Xing [152]. Biznik is a community of small businesses dedicated to helping each other. Cofoundr is an OSN for entrepreneurs, programmers, and investors looking to start new ventures. Xing is a European business OSN. Besides the OSNs designed specially for entrepreneurs, major OSNs such as Facebook and Twitter are also used for advertising, as well as connecting business professionals. These OSNs act as customer relationship management tools for companies selling products and services. Since businesses operate globally, OSNs make it easier to keep in touch with clients around the world.

Dating OSNs

Many online social networks, such as Match.com [98] and eHarmony.com [47], provide an environment for people to communicate and exchange personal information for dating purposes. They require users to give out some private personal information and allow users to search or be searched by selected criteria, but at the same time people can maintain a certain degree of anonymity. Online dating OSNs are similar to other OSNs in the sense that users create profiles to meet and socialize, but the purpose is to find a person of similar interests to date.

A major difference between dating OSNs and OSNs in other application domains is that online dating OSNs usually require a fee [35]. Many users opt to try general OSN services instead, leading to shrinking revenue for the online dating industry. As the number of users keeps increasing for general purpose OSNs (Facebook), online dating services are seeing a decrease in users. It is possible that the general purpose OSN sites will become the new way to find dates online, and replace the dating OSNs .

Education

A huge number of schools build up their own online social networking sites for education and e-learning purpose. Educational Online Social Networks focus on supporting relationships between teacher and student, student and student. Major social learning OSNs such as Ning [112], TermWiki [137], TeachStreet [136], [2] and other sites were built to foster such relationships. Social learning OSNs enable users to post educational blogs, form ad hoc communities, and communicate through chats, discussion threads, and synchronous forums. These sites allow both teachers and students to learn, discover, share content, and rate.

Consumer Advocacy

OSNs have also been used in the application domain of Consumer Advocacy. The OSNs used for consumer advocacy mainly fall into two categories: *(i)* Recommendation and ranking driven online social networking sites, such as tripadvisor [140] and yelp [153] are built for the purpose of having customers rank and recommend hotels, airlines, attractions, restaurants and so forth. *(ii)* Existing OSNs can be used for personalized and customized advertisement. By analyzing the personal profile and social relationships of a user in the OSNs, companies can provide OSN users personal, transparent, inclusive advertisements based on human communities instead of brands. Companies realize that OSNs can accelerate the spread of advertisements. Statistics from Deloitte show that one in three people come to a brand through a recommendation, and customers refereed by their friends have a 37% higher retention rate [159].

Multiple Application Domains

There is a new trend for multiple OSN sites to collaborate in order to enlarge the application domains of a single OSN. For example, Pinterest [119], Glassdoor [59] and other single domain OSNs are connecting to Facebook so that users can maintain connections with their social buddies from multiple OSNs and share information from different domains.

There are multiple reasons for this type of OSN development. First, we believe that none of the existing single functional OSN can satisfy users' requirements. Users expect OSNs to

[2] Teachstreet.com closed down on February 2012.

cover more than one application domains, and we will discuss this in Section 1.4.2. Second, users would prefer to use one OSN site where they can manage all of their public profiles and social contents. In other words, in order to share a picture, they do not need or want to upload to multiple OSN sites.

On the other hand, people also take the privacy of their personal data seriously. During the recent Facebook takeover of Instagram [75] event, a number of Instagram users decided not to discard the application due to the privacy issues facing Facebook. [56].

These opposite requirements for the current online social networking sites led us to believe that the fundamental problem lay in the infrastructure of the current OSN. The centralized infrastructure utilized by OSNs makes the extension of networks very hard, and harms the users' privacy since users give up the copyright and control of their personal data to the OSN providers. Privacy issues will be discussed in Section 1.4.1. We believe unconventional OSNs, especially the decentralization of the OSN, are the solution to this problem, which we will discuss in Section 1.4.1.

1.3.4 Social Presence

Social Content Format

Sharing social content on any Online Social Networking site is as easy as typing text, uploading images and videos, and then hitting the "publish" button. This functionality has been enabled by telecommunication technologies and social criteria in the *Social Presence Infrastructure* layer, which enables fast and efficient multimedia communication between any two OSN users. In this section, we will discuss the Social Presence Infrastructure and the current research on the topic of Social Presence.

Social presence has been defined by many researchers as the feeling of "being with others" [70], "a level of awareness of the co-presence of another human, being or intelligence" [10], "the degree of salience of the other person in the interaction" [127], and the "feeling that one has some level of access or insight into the others intentional, cognitive, or affective states" [10].

In this dissertation, we refer to Social Presence as *technologies that are primarily intended to increase real time social interaction by means of Social Contents*. The Social Presence

Infrastructure layer manages the technologies used for Social Contents communication between users. Therefore, it is a key research topic under Online Social Networks.

Social Presence and Social Content Categories

Based on the temporal characteristics, social contents can be categorized as *static, real-time or ephemeral*. A *static* social content means that the content, once published, will be forwarded to the publisher's friends unless the publisher explicitly removes or changes it, such as sharing a personal profile, a post, or a picture. A *real-time* or *ephemeral* social content is the one that has a time factor, such as Facebook online chatting.

Furthermore, the formats of social contents in the existing OSNs can be divided into four categories: *text, image, audio and video*; and most OSNs support at least one of them. Based on the quality of media, video has the highest social presence among all other mediums, while text has the least social presence. In this section, we will discuss these forms of social presence, as well as how and to what extent the current Online Social Network infrastructure support them.

Plain Text as a Social Presence

Plain text is the main communication method for human socialization in any existing OSN. Interactions between users on main stream Online Social Networking sites all involve plain text. People send tweets, update personal profiles, publish a new personal status, exchange emails, and send instant messages. Moreover, for OSN sites that do not support video/audio social presence, users need to upload their audio/video contents to websites that support such uploading and storage, and then share only the link to the audio/video clip in the format of plain text (URL).

Almost every OSN site supports *static* plain text, but only some of them support *real-time* plain text exchanges, such as the live chat in Facebook.

Images as a Social Presence

Sharing photos publicly or privately with friends is another major functionality supported by most OSNs. OSN sites usually provide users with the ability to select different sharing options

before posting.

The first photo sharing sites were launched during the mid 1990s primarily to provide online photo ordering of prints. Flickr [53], Picasa [118], and Pinterest [119] came into being during the early 2000s with the goal of providing permanent and centralized access to a user's photos. Sharing photos through such Online Social Networks has become a popular trend. OSN sites allow users to specify with whom to share their photo albums, whether it is all users or only those whom they choose.

Nowaday, almost all OSNs offer the capability of sharing photos directly from mobile phones to social networks. The most prominent of these is Instagram, which has quickly become one of the dominant mobile photo sharing social networks with over 15 million members [76].

Current OSNs only support *Static* image sharing.

Audio and Video as a Social Presence

Video cameras with WiFi, and smart phones with cameras allow everyone to be able to record audio or video clips at will. Users can share the recorded audio and video clips with their friends directly in OSNs, or upload the video or audio clips to websites that support multimedia and share the links with their friends.

Current OSNs that are designated for audio sharing include Spotify [134], imeem [74], SoundClick [132], SoundCloud [133], etc. For an exhaustive list of audio/music sharing OSNs, we refer the readers to read these two websites: [131], [5]. Youtube [157] was first created so that friends could share video clips of a dinner party that were too large for email, and now has become the most popular OSN for video sharing. There are also other video sharing OSN sites that can be found in [131].

Current Online Social Networking sites support audio and video sharing in a *Static* way. No real-time or ephemeral exchanges of audio and video clips are available.

1.4 Challenges in Online Social Networks

The big success of Online Social Networks relies on the current centralized infrastructure, which also has many issues that prevent OSNs from developing further, such as those discussed in Section 1.3. Very much like the development of the Internet, which was originally proposed for point to point communication between two computers in October 1969, the Online Social Network was originally proposed to enable two human beings to get in touch and socialize. The Internet has became one of the most important innovations ever, dramatically impacting people's everyday lives by supporting nearly-instant communications by email, instant messaging, VoIP, streaming, video/audio calls, and numerous other applications. The rapid development of the Internet would be impossible without the support of diverse technologies and protocols (DHCP, FTP, HTTP, RPC, IRP in application layer [3], TCP and UDP in the transport layer [139]; IP, BGP, ICMP from Internet layer, and so on), and the corresponding network infrastructures (CDN, NDS, routing, and OSI architecture). Note that the listed protocols and infrastructure represent only a small portion of the technologies used in the Internet, which continues to develop to provide faster and reliable communications to satisfy the demands from multiple applications.

As with the Internet, the purpose of an Online Social Networking service is not limited to the purpose of socialization, but continues growing to provide more functionalities and cover more application domains, as discussed in Section 1.3.3. The OSN infrastructure needs to meet a variety of requirements to support this rapid growth, which poses many challenges to OSN research. Novel, alternative, and unconventional Online Social Networks, as well as the corresponding protocols and architectures, need to be studied. In this dissertation, we specifically address, from both theoretical and practical aspects, three challenges that are critical for developing the next generation OSNs and will present an overview of these challenges in the rest of this section.

1.4.1 Challenges in Core Service and Storage

In this section, we will discuss the challenges applicable to the *Core Service, Storage and Database Infrastructure* layers as shown in Figure 1.2.

As discussed in Section 1.3.1, while Distributed Online Social Networks have been proposed for the users of OSNs to regain their data as well as the control over their personal data, the infrastructure of the DOSN is still an open question.

Multiple questions arise in building a DOSN, which include but are not limited to: should the DOSN be deployed by utilizing a structured P2P underlay or an unstructured one? How should data storage be chosen for duplications to provide reliable accessibility? What data storage should be used in a DOSN? How can messages be disseminated among the distributed online social networks without a centralized server?

Critical among these issues is the lack of an efficient data dissemination schema, since the effective distribution of information among friends is the main purpose of any OSN or DOSN. Data dissemination techniques have been developed in many other networking fields, including: sensor networks, mobile ad hoc networks, and distributed networks. The existing data dissemination methods include gossip protocol, epidemic routing, probabilistic routing based on prediction, distributed caching, and CDN (content delivery networks).

However, none of these existing techniques works for the Distributed Online Social Network infrastructure, for two principal reasons. *(i)* Social updates are different from any other information, since they are short and highly dynamic. They are usually sent from one information producer to many information consumers (friends) and form a one-to-Many mapping. *(ii)* Unlike in other systems, the publication of social updates is very unpredictable, although it may follow some cumulative daily patterns. Therefore, we believe a novel core service and corresponding storage and database infrastructure are needed to meet the requirements of new social information dissemination methods in a Distributed Online Social Network.

1.4.2 Challenges in Application Service

In this section, we will discuss the challenges lying in the *Application Domain Infrastructure* layer, as shown in Figure 1.2. As discussed in Section 1.3.3, current Online Social Networks cover a broad variety of application domains. However, there are still more domains to be covered, including vehicular drivers, property buyers, etc. Vehicular drivers would like to socialize with each other to exchange traffic information, and property buyers would like to discuss real

estate agents, and properties locations.

According to a recent U.S. census, ninety percent of Americans commute every day between home and office, and 80% of those are driving alone. Driving alone has two benifits: the drivers have much more freedom than when they carpool with other people, and they can stop anywhere on their way to the office and back home. However, carpooling with strangers is considered to be unsafe for some drivers. The principle drawback of driving alone is that drivers may feel tired or bored. Some remedies do exist, such as listening to a radio talk show or using a hands-free phone to talk with friends. The radio can also broadcast current traffic information to the listener; however, only the drivers who up ahead know the real-time situation about the road and traffic condistions.

Furthermore, during commuting hours, a huge number of other drivers are also driving on the same road facing the same situation. These drivers are on the same road at the same time but cannot take the advantage of the proximity. We believe that the only method that can enable the drivers to communicate with other drivers in the same proximity, as well as provide them some relief from boredom, is to create the novel *Vehicular Social Networks*. The Vehicular Social Network, which will be discussed in Chapter 4, is an Online Social Network specifically designed to cover the application domain of vehicular drivers, and to facilitate, for drivers in the same proximity, socialization and the exchange of traffic information.

1.4.3 Challenges in Social Presence

In this section, we will discuss challenges lying in the *Social Presence Infrastructure*, as shown in the Figure 1.2. As discussed in Section 1.3.4, the forms of social presence supported by the current Online Social Networks are *static* plain text, image, audio and video, and *real-time* plain text.

Current OSNs provide several real time services, such as *real time instant messaging*, *real time search*, and *real time marketing*. However, users cannot get real time updates unless they access the social network websites or use technologies like RSS. Vehicular Social Networks differ from existing OSNs in two main aspects. First, the Vehicular Social Network is proposed as a channel for motorists to exchange real-time traffic information as well as socialization.

Traffic information messages, as a special social content, have a temporal property: they must be real time and ephemeral messages in order to be meaningful and useful. Second, sending text message is not suitable for the Vehicular Social Network, since texting while driving is not only prohibited by law, but also very dangerous for drivers.

A real-time video message is not a reasonable social presence as well, since driving while watching video is also considered as one of the major distractions for drivers. Furthermore, real-time video messages have privacy issues in a Vehicular Social Network. It is not necessary to see who is speaking, and the user may not want to be seen by other users. Last, real-time video message can waste too much of user bandwidth.

Therefore, we believe the **Real-time Voice Message** is the correct social presence to be used for Vehicular Social Networks. This new social presence, that enables the transfer of real-time/ephemeral voice messages, required to be supported by the Social Presence Infrastructure.

1.5 Summary of Dissertation Contributions

This dissertation present three main contributions in exploring the challenges of improving the performance of Distributed Online Social Networks, and expanding the functionalities of the current Online Social Networks to include Vehicular Social Networks and to form unconventional Online Social Networks. These were published as follows:

- SocialCDN: Caching Techniques for Distributed Social Networks. *In the Proceedings of the 12th IEEE International Conference on Peer-to-Peer Computing (P2P'12),* September 2012 [66];

- Social Butterfly: Social Caches for Distributed Social Networks. *In the Proceedings of the 3rd IEEE International Conference on Social Computing (SocialCom'11),* October 2011 [65];

- Ad-hoc Voice-based Group Communication. *In the Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'10),* March 2010 [67];

- RoadSpeak: Enabling Voice Chat on Roadways using Vehicular Social Networks. *In the Proceedings of the First International Workshop on Social Network Systems (SocialNets'08),* April 2008 [129].

In this dissertation, we present the design, methodology, implementation and evaluation of the following systems: Social Butterfly [65], SocialCDN [66], Vehicular Social Network [129] and RoadSpeak [67]. Social Butterfly and SocialCDN enable efficient data dissemination by exploring both centralized and distributed Social Caching, respectively, to achieve efficient data dissemination in Distributed Online Social Networks. The Vehicular Social Networks (VSNs) expand the current Online Social Network application domains, and RoadSpeak, as an application of VSN, utilizes multiparty voice communication to support socialization between VSN users in the same group.

1.5.1 Social Caching

As discussed in Section 1.4.1, deploying Distributed Online Social Networks (DOSNs) offers many challenges, critical among which is the lack of data dissemination services. Users' personal contents are stored on heterogeneous types of machines varying from cloud storage to personal mobile devices such as smartphones dispersed geographically over a wide area. This implies that any update to personal data should be disseminated to all social contacts in a timely fashion at a low cost. Typical choices for distributing content include the push method, the pull method, and a hybrid of the two methods. Most existing distributed social networks utilize a "push immediately" scheme to handle social updates, sending updates to all contacts. Therefore, $O(N^2)$ connections need to be set up in order to send users' updates to all of their social contacts. Pull is even worse, as connections need to be set up periodically independent of whether there is any update or not. We argue in this dissertation that this data dissemination model should be re-examined in the context of distributed social networks. Rather than sending identical copies of a given social update to all contacts, an efficient strategy, Social Caching, should be developed to optimize the network traffic generated by social updates.

As distributed social networks do not have a central server from which to update and read the social content, we propose a form of distributed caching scheme that intelligently places

data in the network so as to optimize the social traffic. Certain friends "cache" the updates for other friends, and these selected friends act as "*Social Caches.*" By utilizing Social Caches, we can optimize social traffic, reduce network connections and transmission latency, and reduce bandwidth usage in the network.

Social Butterfly [65] achieves the social update dissemination utilizing social caching techniques. Specifically, the cache selection and placement problem is formulated as the "Neighbor-Dominating Set" problem. We propose two centralized Social Cache selection algorithms, the "Approximate NDS" algorithm and "Social Score" algorithm. Both algorithms can select a sufficient number of caches so as to greatly reduce the P2P connections generated by the dissemination of the social contents to a great magnitude.

However, the two algorithms proposed in Social Butterfly require global parameters, such as graph topologies. Therefore, in SocialCDN [66], we propose a set of fully distributed social caching mechanisms. Four cache selection algorithms are proposed, the *Randomized* algorithm, the *Triad Elimination* algorithm, the *Span Elimination* algorithm, and the *Distributed Social Score* algorithm. The evaluation results show that among these four algorithms, *Span Elimination* outperforms the other three and can achieve a performance similar to the centralized method. These proposed social caching techniques can be applied on the distributed social networks of any kind.

1.5.2 Vehicular Social Networks

Vehicular Social Networks

First, we will present the Vehicular Social Networks(VSNs) [129], a novel online social network that expands the current OSN application domain to reach vehicular drivers, especially the daily commuters. Vehicular Social Networks perfectly fuse Online Social Networking with vehicular and transportation networks.

As we discussed in Section 1.4.2, the current Online Social Network application domains range from socialization, to a large variety of areas such as career, business, dating, education, etc. However, at the time of writing, none of the existing OSNs serves the vehicular drivers.

People partake in all forms of entertainment while driving to destinations. They may listen

to music, listen to and participate in radio talk shows, chat with friends over cell phones. In fact, in analogous commuting situations, e.g., on buses or trains, it is common for people to socialize and communicate with each other. We propose Vehicular Social Networks as a substantial opportunity for people to organize into social groups and expand their social connectedness while traveling on roadways, for the purposes of entertainment, socialization, passing the time during their commute, seeking for better routes during traffic jam, or even asking for help in case of an emergency.

There are three major purposes for Vehicular Social Networks: (i) for entertainment, (ii) for utility, and (iii) for dealing with an emergency. Entertainment-based Vehicular Social Networks are formed for people to share common interests, for example, to discuss recent social and political issues, sports, or any other interesting and entertaining topic. Entertainment-based Vehicular Social Networks primarily serve to occupy people during their long and boring commute, similar to the way radio programs provide entertainment. Utility-based Vehicular Social Networks are formed to facilitate non-essential yet helpful connections. For example, local towns or states might establish Vehicular Social Networks to suggest interesting local events or points of interest along popular highways, or for travelers to query for advice in selecting a local restaurant or hotel. There might also be Vehicular Social Networks established to coordinate carpooling, or to communicate roadway conditions (e.g., accidents, congestion, etc). Finally, Vehicular Social Networks can become particularly useful in the case of an emergency. Emergency-based Vehicular Social Networks might serve as a way for people to ask for and provide assistance to each other in critical situations such as a traffic accident, or disaster evacuation.

The key property of Vehicular Social Networks is that roadways, on which the Vehicular Social Networks are overlaid, provide a sufficient and regular concentration of people who wish to socialize.

RoadSpeak

As we discussed in Section 1.4.3, existing social presence lacks support for real-time audio/video information exchanges. However, audio exchange is extremely important for Vehicular Social Networks, where texting is a violation of the law. Reasonably enough, however, in most states, hands-free devices, such as smart phones, are permitted for information exchanges while driving.

We propose RoadSpeak [67], a Vehicular Social Network based system that allows large groups of motorists to socialize and communicate with each other by automatically joining Voice Communication Groups formed along popular roadways. A RoadSpeak Voice Communication Group is defined by a group owner when the group is originally created.

In a traditional Voice Communication Group system, such as CB radio, audio collisions effectively limit the number of active participants in a group, occurring when participants speak simultaneously or when one participant is interrupted by another. Such collisions increase the frustration level of participants, because they slow the rate of progress in the conversation and force participants to repeat their messages at the next opportunity.

To enable scaling to large voice chat groups, RoadSpeak utilizes automated moderation, three-way buffers and flow control to allow individual participants to experience an interruption-free conversation. A server-side buffer buffers the incoming messages and transmits them to the conversation participants, which decouples the sender from the receivers to ensure message delivery; a client-side sending buffer ensures the message can be sent to the server once completed; a client-side receiving buffer buffers the message before playing it back to enable clients to cancel or skip the message.

1.6 Contributors to the Dissertation

The following is a list of my colleagues who co-authored papers from which material was used in this dissertation. The Social Butterfly project presented in Chapter 2 and SocialCDN discussed in Chapter 3 are a result of collaboration between my advisor Professor Liviu Iftode, Professor Badri Nath, and Professor Shan Muthukrishnan. Magdalena Punceva contributed to

the design of cache selection algorithms for the SocialCDN system. Chapter 4 of this dissertation is the result of the Vehicular Social Networks and RoadSpeak projects that I worked with my advisor, Professor Liviu Iftode. Stephen Smaldone and Pravin Shankar contributed to the design and evaluation of the system. James Boyce contributed to the design and implementation of the front-end web interface for the RoadSpeak. Disco Lab members participated in the RoadSpeak field trial, and provided many insightful suggestions.

1.7 Dissertation Organization

The dissertation is organized as follows. Chapter 2 presents the Social Butterfly communication model, which utilizes the *Social Caching* technique for efficient data dissemination in Distributed Online Social Networks. Chapter 3 presents the SocialCDN system, which focuses on a set of fully *Distributed Social Caching* algorithms for data dissemination in DOSN of any kind. Chapter 4 describes Vehicular Social Networks (VSN), as well as an application based on VSNs, the RoadSpeak. The Vehicular Social Network extends the current application domain of Online Social Networks to include the motorist users. The RoadSpeak system utilizes multi-party voice communication system with automated moderation, and flow control techniques to enable large scale socialization between vehicular drivers in a Vehicular Social Network infrastructure. Finally, Chapter 5 concludes the dissertation and discusses the directions for future study.

Chapter 2

Social Caching for Distributed Online Social Networks

Recent years have seen an explosion in using online social networks (OSNs) such as of Facebook [48], Twitter [143], Google+, Weibo [150] and LinkedIn [90], among others. These online social networks have greatly revolutionized the way people interact and share information over the web. Online social networks enable users to join a network, publish their personal profiles and contents, create links to other users with whom they associate, and annotate on the published contents with "likes" and comments. The resulted social network provides a basis for maintaining and expanding social relationships, for finding users with similar interests to form social groups, and for searching content that has been contributed or endorsed by other users.

This revolution in human interaction through social media has brought to the forefront the issues of ownership and control of user generated data. In the case of centralized "Online Social Networking" sites, all the information that users generate or consume is stored on centralized servers. This information is mostly private and users voluntarily share it with the OSN site, in order to share it with their friends. Thus, users have less control over their personal profiles and data, which is often scattered over different OSN providers, which usually do not support data interoperability.

Recently, Distributed Online Social Networks (DOSNs), such as Diaspora [37], PeerSoN [21], and PrPI [126], have been proposed and developed as an antidote to centralized OSNs, with the goal to overcome some of the problems associated with the traditional OSNs. A DOSN is a P2P infrastructure that supports the features of OSNs in a distributed way. DOSNs enable users to host and organize their personal profiles and social connections at the place of their choice, such as cloud storage, enterprise servers, or personal devices, while retaining full control over their own data. With such flexible distributed storage, users can manage their data, control with whom the data are shared, and determine which third party applications can access their data.

An exhaustive list of existing DOSNs is maintained on the Wiki page [41].

The obvious advantages of DOSNs over centralized OSNs are counter-balanced by several challenges in deploying the DOSNs. Critical among them is the need for a scalable *social update dissemination* service. A Social Update is defined as any social content that users share with their friends, such as changes to profile information, wall postings, pictures, videos, status updates, links, messages, tweets, etc. Sending and browsing social updates represent a significant portion of the activities in OSNs, and contribute to the majority of the network traffic. According to a recent Facebook survey [64], on an average day, 15% of Facebook users update personal status; 22% comment on others' posts or status; 20% comment on others' photos; 26% "Like" other users' contents; and 10% send another user a private message. Social network sites generate more network traffic than most of the other websites. 28% of the Internet traffic comes from Social Media [110], ranking as the second source of Internet traffic after search engines

While a CDN [38] can effectively reduce the load on a centralized server, the feasibility of the DOSNs critically depends on the efficiency of social update delivery and on measures taken to support good data availability. The former can be achieved through caching in the social network, while the latter can be achieved through redundancy.

In this Chapter, we introduce *Social Caches* as a way to reduce the number of peer-to-peer network connections, as well as to alleviate the peer-to-peer network traffic that would ensue in a distributed OSNs. The proposed social caches act as local bridges among friends for efficient information delivery. The traffic pattern and the relationships among participants determine the placement of caches; hence the term social cache. We further propose a novel social communication model, *Social Butterfly*, for DOSNs that utilize social caches. We formulate the social cache selection problem as an instance of the *Neighbor-Dominating Set Problem*, and propose two algorithms to solve it: 1) Approximate NDS, and 2) Social Score algorithm. Performance evaluation based on real social traffic data shows that both algorithms reduce the number of peer-to-peer social connections by an order of magnitude.

We state the problems in Section 2.1, and discuss the related work in Section 2.2. Sections 2.3 and 2.4 introduce the Social Butterfly communication model and the social caches,

respectively. In Section 2.5, we formulate the social cache selection problem, and present the Approximate NDS and the Social Score algorithms. We discuss the dataset statistics that we use for evaluation and the social traffic pattern of the dataset in Section 2.6. Evaluation results are presented in Section 2.7. We conclude the chapter in Section 2.8.

2.1 Problem Statement

Online Social Network Limitations

The centralized architecture underlying the popular online social networks, while offering convenience, availability, and single point of control, suffers from several weaknesses:

(i) the data sources are disparate; users have to maintain multiple accounts and multiple data formats for different social networks;

(ii) user data is locked in, resulting in the so-called "big brother" effect; users lose control of the ownership of their personal data;

(iii) centralized servers provide a platform for information leakage [84]; recently, it was reported that Facebook applications leaked user names, phone numbers and addresses to a third party [52].

Privacy concerns and security threats are serious weaknesses of centralized social services provided by OSNs. Lack of privacy and control over personal content stored in servers are considered the major drawbacks with the centralized architecture of online social networks.

Distributed Online Social Networks

As an alternative to centralized architecture, various Distributed Online Social Networks (DOSNs), such as Diaspora [37], DSNP [42], Safebook [31], PrPI [126], Musubi [39] and AppleSeed [4], [1] have been recently proposed and developed to provide solutions to the privacy and scaling problems facing centralized online social networks. Distributed online social networks provide a platform for users to communicate securely and allow users to get back ownership of their personal data, thereby preventing to a large extent information leakage and privacy violations.

[1] Distributed Online Social Networks are also referred to as Peer-to-Peer Social Networks, or Decentralized Social Networks. We use Distributed Social Networks in this dissertation.

Users store and organize personal contents at the location of their choice such as cloud storage, enterprise servers, or personal devices. With such flexible distributed storage, users have control over their personal data, and can share this data with social contacts in their terms. Distributed Online Social Networks help overcome constraints imposed by the centralized social network providers in the form of social contents, such as number of pictures, size of pictures, and commercial ads.

Peer-to-peer systems and social networks share a common goal of resource and information sharing. However, Distributed Online Social Networks differ from the traditional P2P systems in the degree of usability. P2P networks have been mainly used in file/media sharing and instant messaging, such as BitTorrent, VoIP, Skype, etc. Peer-to-Peer file/media sharing systems assume the shared files do not change; in the BitTorrent system, for example, a new torrent needs to be generated and published when new files are added or changed. This property makes the traditional P2P networks unfit for handling the dynamically changing social relationships and social status updates.

DOSN Deployment

The current deployments of Distributed Online Social Networks can be divided into three categories:

(i) Federation, which manages social network providers collectively to agree upon standards of operations. Federated social networks enable users to share their profiles and connections in one OSN with friends from other OSNs. For example, Facebook users can enable their accounts connected with Pinterest [119] to share Pinterest picture with Facebook friends.

(ii) OSNs over unstructured P2P underlay, which utilizes a lookup server for bootstrapping functionalities, such as PeerSoN [124] and PrPl [126].

(iii) OSNs over structured P2P underlay, which utilize DHT underlays such as My3 [109] and PeerSoN [21]. My3 focuses on availability by replication, and models the replica placement problem as a Dominating Set (DS), while Peerson focuses on privacy.

Challenges in Building a DOSN

There are several challenges in building a truly distributed social network.

(i) Usability. The system should be easy to use, deploy, and manage by non-technical users.

(ii) Social Updates Dissemination. A *Social Update* is defined as any social content that users share with their friends, such as changes to profile information, wall postings, uploading pictures/video, updating status, sharing links, messaging, tweeting, etc. An efficient strategy is required for social updates delivery.

(iii) Social Topology and Storage. What is the architecture of the DOSN? Where should personal social content be hosted? How many replicas are needed to place inside the network? We need redundancy to provide availability, and this may be influenced by the geographic locations of peers and time zone differences.

(iv) Searching and Naming. A potential solution maybe Wayfinder [117], a peer-to-peer file system that targets the needs of content sharing communities. It provides a global namespace, content-based queries, and automatic availability management.

(v) Openness to applications. The systems should be extensible and easy for developers to write third-party applications. The infrastructures and APIs should be open, and modular.

(vi) Compatible with Mobile Devices. Almost all centralized online social networks support mobile access to a great extent. Smartphones/Tablets have become a substitute for traditional computers, and very important carry-on personal devices with 4-16GB storage. As natural social environment collectors and a type of hosts, Smartphones/Tablets can be used to collect personal social data (pictures, videos, locations, etc), social environmental information (temperature, compass, local sound, etc), and be a natural host for Distributed Online Social Networks. However, compare with servers, mobile devices usually have limited resources such as battery, data plan, etc. Continuously hosting a DOSN service will drain the resources very quickly. Therefore, DOSN design must take the smart device hosts into consideration.

Critical among these challenges generated by Distributed Online Social Networks deployment is the lack of data dissemination services. Users' personal contents are stored on heterogeneous machines varying from cloud storage to personal mobile devices such as smart phones dispersed geographically over a wide area. This implies that any update to personal data should

be disseminated to all social contacts in a timely fashion at a low cost. Typical choices for distributing content include the push method, pull method, and a hybrid of the two methods. Most existing distributed social networks utilize a "push immediately" scheme to handle social updates by sending updates to all contacts. Therefore, N connections need to be set up for sending a given user's updates to N social contacts. Using the naive "push-to-all-friends-immediately" method requires a TCP setup and tear-down for sending every social update with each friend. Generally, in a network with $|V|$ users, and $|E|$ edges, $|2 * E|$ TCP connections need to be set up to send and receive ONE social update for each user, which is a big waste of resources and bandwidth. For those who use smart devices to host their social data, this method of updating will be a significant drain on bandwidth and battery. Pull is even worse, as $|2 * E|$ connections need to be set up periodically independent of whether there is any update or not. The data dissemination model should be re-examined in the context of DOSN. Rather than sending identical copies of a given social update to all contacts, an efficient strategy should be developed to optimize the network traffic/connections generated by social updates.

2.2 Related Work

Before being used to represent real peer-to-peer social networks, the term "Distributed Social Networks" had been used to represent umbrella protocols that aim to integrate existing centralized social networks. Umbrella projects such as Distributed Social Networking Protocol [43] and Distributed Social Networking Technologies [44], allow users to choose on which social network providers to host personal profiles, maintain control over personal information, and interact with their friends in a secure manner. However, umbrella projects only provide mechanisms for covering existing social network sites, users still need to rely on centralized services.

Recently, Distributed Social Networks (DSNs) such as Diaspora [37] and Safebook [31] have been proposed. Diaspora allows users to host their data on their personal machines, and to communicate with friends in a P2P manner. However, in order to host personal data, a user must have a domain name and must know how to set up a web server. Otherwise, users can host personal data on machines provided by Diaspora volunteers. This not only makes the users who do not host their own data lose data ownership again, but also makes the volunteers'

machines vulnerable. Cutillo, et al. proposed Safebook [31], a distributed social network that leverages trust relationships to cope with the lack of trust and cooperation in P2P systems. The system utilizes "Matryoshkas", concentric rings of nodes built around each member, to provide data storage to replicate friends' profiles. Dodson, et al. proposed Musubi [39], which is a social sharing platform that enables users to share and interact with friends on the phone in a decentralized way without having to give up privacy to any third-party service providers. There are some other DOSNs that are being developed such as PeerSoN [21] and PrPI [126]. They have yet to exploit social update patterns and do not provide efficient information delivery mechanism for social updates.

2.3 Social Caching

As Distributed Online Social Networks do not have a central server from which to update and read the content, we propose a novel distributed social network data dissemination model called *Social Caching*. Social caching is a caching scheme that intelligently places data in the social network so as to optimize the message traffic that originates from members of the DOSNs. Certain friends "cache" the updates for other friends, and these selected friends act as "*Social Caches.*" Utilizing social caches, we can optimize network connections generated by distributing social updates, reduce the transmission latency of setting up multiple connections, and reduce bandwidth usage in the network.

In this section, we describe the design of Social Caching, which selects certain nodes as "local servers" for bridging social updates delivery between the social update producers and consumers, thereby reducing the cost of operating a distributed online social network. These selected "local servers" nodes are the *Social Caches*, while the remaining nodes are assigned as members of one or more social caches to form *Social Clusters*. A member node only contacts the social caches it is associated with, while a social cache can be a member of multiple social clusters. Every node in the social network can be both a consumer and producer. Producers push social updates to the social caches they are associated with, while consumers fetch data from the corresponding social caches when they want to learn the updates from their friends.

Let's consider the example shown in Figure. 2.1, where upper diagram shows social relationships, and the lower diagram shows where users store personal data. As a social network user, Alice connects to her colleagues in graduate school as well as to classmates in college. Without social cache, Alice needs to set up connection and send any update to everyone in her network. Bob is one of Alice's college classmates who connects to everyone of Alice's college classmates and hosts his personal data on a highly available cloud storage service. Charlie is Alice's colleague in graduate school connected to all of Alice's graduate school connections. Therefore, Bob and Charlie can be selected as social caches for the two corresponding groups for Alice. With these social caches, when Alice updates her status, besides changing her personal profile on her host, she caches the new status on Bob's and Charlie's storage as well. When her college classmates want to see her update, they fetch it from Bob (same situation for Charlie and her graduate colleagues), which greatly reduces the total number of P2P network connections. Moreover, if Alice only wants to share some news with her college classmates, she can choose to cache the news to Bob only. In this scenario, besides reducing the number of network connections, users have more control over their data and can select with which group of people they would like to share their personal data.

Typically, social update consumers and producers in social networks are friends and are at a one-hop distance. This property requires the social caches that bridge social update consumers and producers to be friends with both the consumers and the producers if none of them is a social cache. Thus, social caches have information that only belongs to friends. Furthermore, for the purpose of minimizing social traffic and network latency, both sending and receiving social update actions should be, wherever possible, within one-hop of each other. This requirement ensures that producers push social updates to a social cache one-hop away and consumers fetch them from a social cache, which is also one-hop away. If the number of selected caches K is significantly less than the number of friends N, then the use of social caches will significantly reduce the network connections when compared to the "push-to-all-friends" or "pull-from-all-friends" approach.

Figure 2.2 gives an example of the distributed social network architecture with six nodes. The edges in Figure 2.2-a show the friend relationships and the edges in Figure 2.2-b show the

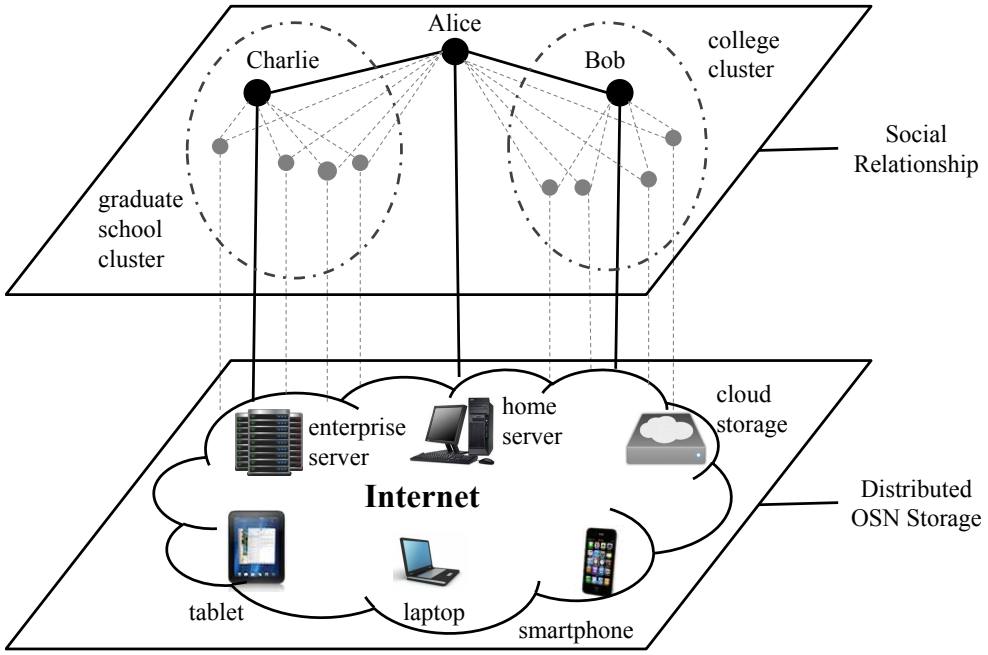
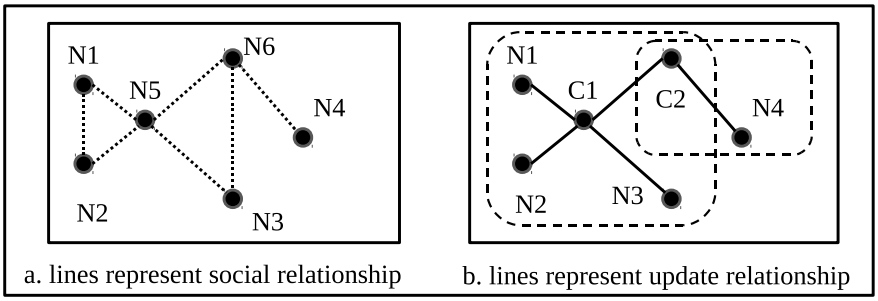


Figure 2.1: An example scenario to illustrate the use of social caches.



The edges in Figure-a show the friends relationship. Push schema incurs an update cost of $\sum_{i=1}^{6} degree(N_i)$ = 14. The edges in Figure-b show the update relationship with social caches C1 and C2. Updates are from N1, N2, N3 and C2 to C1, and N4 to C5, which incurs a total cost of 5.

Figure 2.2: Social updates cost without and with social caches in DOSN.

update relationships, where C1 and C2 act as social caches. In Figure 2.2-a, the push scheme will incur an update cost (in the term of network connections) equal to the sum of the out degree of all the nodes, which is significantly higher than the update cost incurred in the network with social caches ($5 \ll 14$) as shown in Figure 2.2-b.

The design of social caching is also based on the observation made by Viswanath et al. [147]

that, people actively interact with only 30% of their friends on Facebook, and rarely update the rest of their friends. This indicates that it is not necessary to push every social update to all friends immediately. With the social caching method, we can not only reduce the number of social connections within distributed online social networks, but also save the effort of sending the information to the 70% of users with whom we do not actively interact.

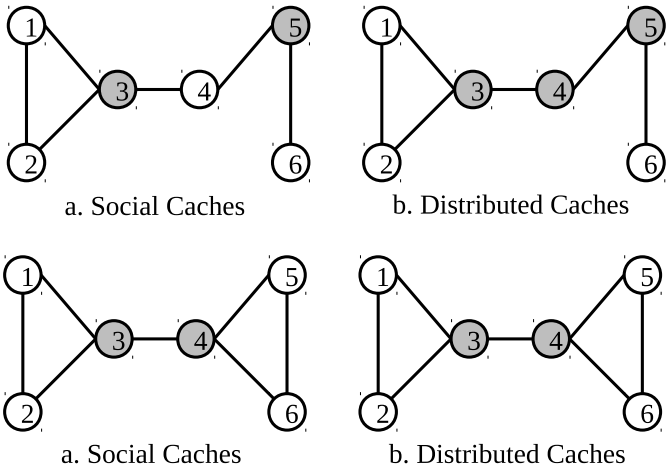
Designing a good social cache selection algorithm, and forming the social clusters based on the selected social caches are not trivial since among all of the potential candidates for a cache, there is a choice to be made based on the underlying social graph. We will discuss the two social cache selection algorithms in Section 2.5.

2.4 Social Caching v.s. Distributed Caching

Distributed caching [154], [79], and Dynamic Data Replication [2] are well studied topics. Traditional distributed caching mechanisms are mainly used for achieving lower latency and for minimizing the overall access cost for information providers and consumers. The cache size and distance to the consumers are the main factors that determine the selection of distributed caches. Content distribution networks also provide caches to reduce the overall cost of access by intelligently placing caches close to consumers. However, they differ fundamentally from social caching in several aspects.

First, social network topology is different from common distributed networks, exhibiting non-trivial clustering (network transitivity) and positive degree correlations [111]. Distributed cache placement and dynamic data replication algorithms are designed for arbitrary or tree topology networks. Second, the basis for selecting caches differs. Selection of distributed caches and data replications is demand-driven, passive, and dependent on routing protocols. Caches and replications change dynamically to ensure that the latest, most popular contents are stored. On the other hand, selection of social caches is social-relationship-driven; information providers proactively send updates to the selected social caches. Third, selection of distributed caches or data replications is purely syntactic; in another words, any node that can reduce latency for the distributed cache or reduce message traffic for the data replica can be selected. Social caches are selected to cache social updates only for friends to ensure security compliance

and allow only one-hop communication (i.e., communication between friends). Finally, in distributed networks, the number of information providers is small and sparse compared to information consumers in the network. However, in distributed social networks, every node can be considered as an information provider. Therefore, neither distributed cache selection algorithms nor dynamic data replication placement methods can be used for selecting social caches.



Given the same topology, the top two graphs show that social cache and distributed cache selection yield different caches. The bottom two graphs show that social cache and distributed cache selection yield the same caches. The shaded vertices are the selected caches.

Figure 2.3: Social cache v.s. Distributed cache.

Figure 2.3 illustrates that, given the same network topologies, social cache selection and distributed cache selection methods can yield both different and identical caches. Both left-hand figures (a) show social cache selection scenarios in distributed social networks, where vertices represent social nodes and edges represent friendship relations; The right-hand figures (b) show distributed cache selection in distributed networks, where edges represent connectivity/distance. The shaded vertices are the corresponding selected caches. The top two graphs show that given the same graph topology, distributed cache selection and social cache selection may yield different caches. In the social cache scenario, node 3 is the social cache for nodes 1, 2 and 4. Node 5 is the social cache for nodes 4 and 6. In the distributed cache scenario, on the other hand, 3, 4, and 5 could be selected as caches depending on where the providers and consumers are relative to each other, as there is no one-hop constraint. In the bottom two

graphs, given the same graph topology, distributed cache selection and social cache selection methods yield identical caches. In the social cache scenario, node 3 is the cache for nodes 1,2, and 4, while node 4 is the cache for nodes 3, 5 and 6. The distributed cache selection schema produces the same caches, in which node 3 caches content for nodes 1 and 2, node 4 caches content for nodes 5 and 6.

2.5 Social Cache Selection Algorithms

The selected social caches should satisfy the following requirements: given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the social caches form a *Neighbor-Dominating Set (NDS)* such that:

- Every vertex should either be a social cache or connect to at least one social cache;
- A pair of friends should be connected by at least one social cache if none of them is a social cache.

In this section, we define the NDS problem formally, and discuss two social cache selection algorithms: *(i)* Approximate NDS algorithm; and *(ii)* Social Score algorithm. The approximate NDS algorithm selects social caches by approximating the problem to the Set Cover problem. The Social Score algorithm selects social caches by taking into account the nodes connectivity in a social graph. The social score algorithm selects the nodes with higher social scores, where a social score is the representation of a vertex's social properties.

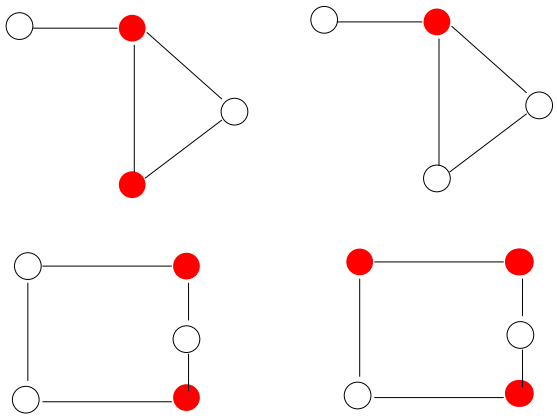
2.5.1 Problem Definition

Define $N(u) \subseteq V$ to be the set of neighbors of node u, that is, $v \in N(u)$ iff $(u, v) \in E$.

Definition 1. *The* Neighbor-Dominating Set *of graph $G = (V, E)$ is the set $S \subseteq V$ of vertices such that for each edge $(u, v) \in E$, there exists a $w \in S$ satisfying $w \in (N(u) \cap N(v)) \cup \{u, v\}$.*

Intuitively, for each edge (u, v), either one of its endpoints should be in the neighbor-dominating set, or one of the common neighbors of u and v.

We should distinguish this from a number of related, well-known concepts. For example, a *vertex cover* is a set $S \subseteq V$ such that for each edge $(u, v) \in E$, either $u \in S$ or $v \in S$. Also, a



The top two graphs show that the minimum vertex cover (on the left) is not the same as the minimum neighbor-dominating set (on the right). The bottom two graphs show that the minimum dominating set (on the left) is not the same as the minimum neighbor-dominating set (on the right).

Figure 2.4: NDS v.s. Vertex Cover v.s. Dominating Set

dominating set is a set $S \subseteq V$ of vertices such that for all vertices $u \in V - S$, there exists an edge $(u, w) \in E$, where $w \in S$. See Figure. 2.4 for examples where these concepts differ from one another.

Definition 2. *(*`NDS` *problem) Given an undirected graph $G = (V, E)$, find a neighbor-dominating set of the smallest size.*

2.5.2 Approximate NDS Algorithm

Our algorithm is as follows: from an instance of the `NDS` problem, we will generate an instance of the set cover problem. Given a collection $\mathcal{S}$ of sets over universe U, a *set cover* is a collection of sets from $\mathcal{S}$ whose union is U. [151]

The universe is the set E of edges. For each vertex $u \in V$, generate a set S_u such that $e = (\alpha, \beta) \in E$ belongs to S_u iff $u \in (N(\alpha) \cap N(\beta)) \cup \{\alpha, \beta\}$. Let $\mathcal{S}$ be the collection of all such S_u's.

Lemma 3. *Each set cover $\mathcal{T}$ is a neighbor-dominating set of size $|T|$, and each neighbor-dominating set S gives a set cover of size $|S|$.*

Proof. Consider the set cover $\mathcal{T}$ of $\mathcal{S}$. Each set S_u in $\mathcal{T}$ corresponds to some vertex u.

Consider the set S of all such $u's$. Since $\mathcal{T}$ is a set cover, for each edge $e = (\alpha, \beta)$, there is some set $S_u \in \mathcal{T}$ that contains it and in turn, $u \in S$. Thus, S is a neighbor-dominating set. The proof that edge neighbor-dominating set S gives a set cover of size is $|S|$ in a similar way. ■

Theorem 4. *There is an $O(\log m)$ approximation algorithm for the neighbor-dominating set problem that takes time $O(md)$, where m is the number of edges and d is the maximum degree of any node in G.*

Proof. This follows from the best known approximation for the set cover problem. [151]. ■

We implement a greedy algorithm for the set cover problem. For each vertex $v_i \in V$, generate a set S_i that contains a set of edges $e = (\alpha, \beta)$, where either $v_i \in \{\alpha, \beta\}$, or v_i is a common neighbor of (α, β). We have a finite set of edges E and a family $\mathcal{S}$ of subset of E, such that every element of E belongs to at least one subset of S. At each stage, the algorithm picks the set $S_i \subseteq S$ that covers the greatest numbers of elements not yet covered. The v_i selected together with S_i is the set of social caches. The vertices in each set S_i form the corresponding social cluster. Algorithm 1 describes the approximate NDS algorithm.

Algorithm 1: Approximate NDS Algorithm

Input: undirected graph *G* = (*V, E*)
Outputs: Social caches *C* and social clusters *SC*
Initialization:
Universal Set = *E*
Covered Set = ϕ
for each vertex $v_i \in V$ **do**
  generate S_i contains edges $e = (\alpha, \beta)$, where $v_i \in (N(\alpha) \cap N(\beta)) \cup \{\alpha, \beta\}$.
end for
Main algorithm:
while Universal Set != ϕ **do**
  select S_i that maximizes |Universal Set $\cap$ S_i|
  Universal Set = Universal Set - S_i
  Covered Set = Covered Set $\cup$ S_i
end while

Let's look at an example graph shown in Figure 2.2-a. For each vertex $v_i \in V$, we generate a set of edges in v_i's egocentric network, which is shown in Table 2.1. According to Algorithm 1, vertex 5 will be selected first as it covers 6 edges out of 7. Vertex 6 (or 4) will be selected next, which covers edge (4, 6). Therefore, $\{5, 6\}$ are the selected social caches.

Vertex	Edges in subset
1	{(1,2), (1,5), (2,5)}
2	{(1,2), (1,5), (2,5)}
3	{(3,5), (3,6), (5,6)}
4	{(4,6)}
5	{(1,2), (1,5), (2,5), (3,5), (5,6), (3,6)}
6	{(4,6), (5,6), (3,5), (3,6)}

Table 2.1: Subset of edges for each vertex in Figure 2.2-a

The approximate NDS algorithm works well on social graphs, and is valid for any graph. However, the algorithm does not exploit any properties exhibited by the social graph. Since we are designing a novel selection scheme in the context of social networks, we can also integrate social properties into the cache selection algorithm. We will now discuss an algorithm that exploits social properties called the *Social Score Algorithm*.

2.5.3 Social Score Algorithm

Before describing the algorithm, we will discuss two important social network properties that are used in this algorithm.

Clustering Coefficient

The clustering coefficient is a measure of degree to which vertices in a graph tend to cluster together [71]. Social networks show a property of community structure where groups of nodes have a high density of links within them, with a lower density of links between groups. A vertex with a lower clustering coefficient indicates that it may connect to many unconnected nodes that belong to different communities.

Egocentric Betweenness Centrality

An Egocentric network [97] is a "local" network for a vertex that contains only the vertex and its one-hop neighbors. Betweenness centrality measures to the extent to which a vertex can facilitate communication with others in the network. Vertices that occur on many shortest paths

between other vertices have higher betweenness centrality. Egocentric Betweenness Centrality [97] is the corresponding betweenness for an egocentric network. It not only shows how many times a vertex is on the shortest path of all its neighbors in the egocentric network, but also indicates how many additional pairs of nodes among the neighbors could be connected if a vertex were selected as a social cache.

Social Score Algorithm

In addition to clustering coefficient and egocentric betweenness centrality, vertex degree is also an important factor. Higher degree vertices with more neighbors should have a higher possibility of being selected as a social cache. Therefore, vertices with lower clustering coefficient (CC), higher egocentric betweenness centrality (EBC), and higher degree connect to a larger number of non-interconnected nodes, and should be the potential candidates for social caches.

The above discussion suggests that a social score can be defined as a combination of cluster coefficient with egocentric betweenness centrality scaling by vertex degree in the form of equation 2.1.

$$SS = ((1 - CC) + EBC) * D \tag{2.1}$$

A similar metric has been used in ad-hoc networks to select the backbone nodes for energy-efficient forwarding[24].

In Figure 2.2-a, node 5 connects two groups of nodes and is on the shortest paths of all its friends. Therefore, it has the highest score in that network. If selected as a social cache, it can reduce the cost of information delivery for both clusters. Nodes 1, 2, 3 and 6 have lower social scores. Among them, nodes 1, 2 and 3 have the same egocentric networks, they are inside an inter-connected clique, thus their clustering coefficient is 1. They have a low degree, and are not on their neighbor's shortest path. Node 4 is an endpoint with only one neighbor. For any endpoint vertex, the clustering coefficient is 0 and egocentric betweenness centrality is 0; therefore, it have a constant social score of 1.

The social score algorithm works as follows: for each vertex v_i in the graph, we maintain a table with the following six fields: clustering coefficient, egocentric betweenness centrality,

social score, IsCache, Cache_List, Member_List. The IsCache field is a boolean variable indicating whether the vertex is a social cache or not, and the Cache_List field stores the social caches vertex v_i affiliate to. Member_List holds all the members within a social cluster if the vertex is a cache. Algorithm 2 presents the Social Score algorithm. During each stage, we pick the vertex with the highest score as a social cache, and treat its neighbors who have not yet been covered by other social clusters as its social cluster members.

Algorithm 2: Social Score Algorithm

Input: undirected graph *G* = (*V, E*)
Outputs: Social caches *C* and Social clusters *SC*
Initialization:
for each vertex $v_i \in V$ **do**
  calculate CC_i, EBC_i, and SS_i;
  set $IsCache_i$, $Cache\_List_i$ and $Member\_List_i$ to ϕ;
end for
Main algorithm:
for each vertex in *V* **do**
  pick vertex v_j with highest social score
  if $Cache\_List_j == \phi$ **then**
    $IsCache_j$ = True
    $Member\_List_j$ = v_j's friend F_j
  else
    get $Cache\_List_j$ and friends of v_j: F_j
    for each cache $C_k \in Cache\_List_j$ **do**
      get friends of C_k: F_k
      $Member\_List_j = F_j - F_k$
    end for
    if $Member\_List_j$!= ϕ **then**
      $IsCache_j$ = True
      update $Member\_List_j$
    end if
  end if
end for

We use Figure 2.2-a as an example to illustrate the algorithm. The clustering coefficient, egocentric betweenness centrality and social score are calculated in Table 2.2. Node 5 and node 6 are the selected social caches with sets $\{1,2,3,6\}$ and $\{4\}$ as social cluster members.

Node	Degree	CC	EBC	SS	IsCache	Cache_list
1	2	1.0	0.0	0.0	N	{5}
2	2	1.0	0.0	0.0	N	{5}
3	2	1.0	0.0	0.0	N	{5}
4	1	0.0	0.0	1.0	N	{6}
5	4	0.333	0.667	16/3	Y	{}
6	3	0.333	0.667	12/3	Y	{5}

Table 2.2: Table for Social Score algorithm.

2.6 Dataset and Social Traffic Pattern

In order to evaluate social cache selection algorithms, we study two datasets and the *Social Traffic Pattern* for these two datasets in this section. We define *Social Traffic* to be a piece of information passing from one user to another in the social networks. Social Traffic usually comes from three sources: *(i)* original composed, such as status updates and picture posting; *(ii)* non-original copy and paste, such as a re-tweet; *(iii)* advertisements and spam [77].

2.6.1 Datasets

Facebook and Twitter are the two most popular social networks today. In order to understand the social traffic pattern, we analyze social updates for both Facebook and Twitter datasets.

Facebook is an undirected graph as it requires a pair of friends to be connected. Twitter connects people by means of "following" and forms a directed graph. Followers are people who follow a user, and friends are anybody whom a user follows. A user's followers do not need to be his friends, and vice versa. Facebook allows users to communicate via various social update methods, such as posting on the wall, like, etc; while Twitter uses tweets as the main communication medium.

The Facebook dataset we study was crawled from Facebook New Orleans Network by [147]. The dataset contains the public profiles of 60,290 users who are connected by 1,545,686 links with an average node degree of 25.6418. The dataset also crawled the wall posts data spans from September 26th, 2006 until January 22nd, 2009. Specifically, we use posts from Jan. 1st, 2007 to Dec. 31st, 2008, which includes 838,092 wall posts.

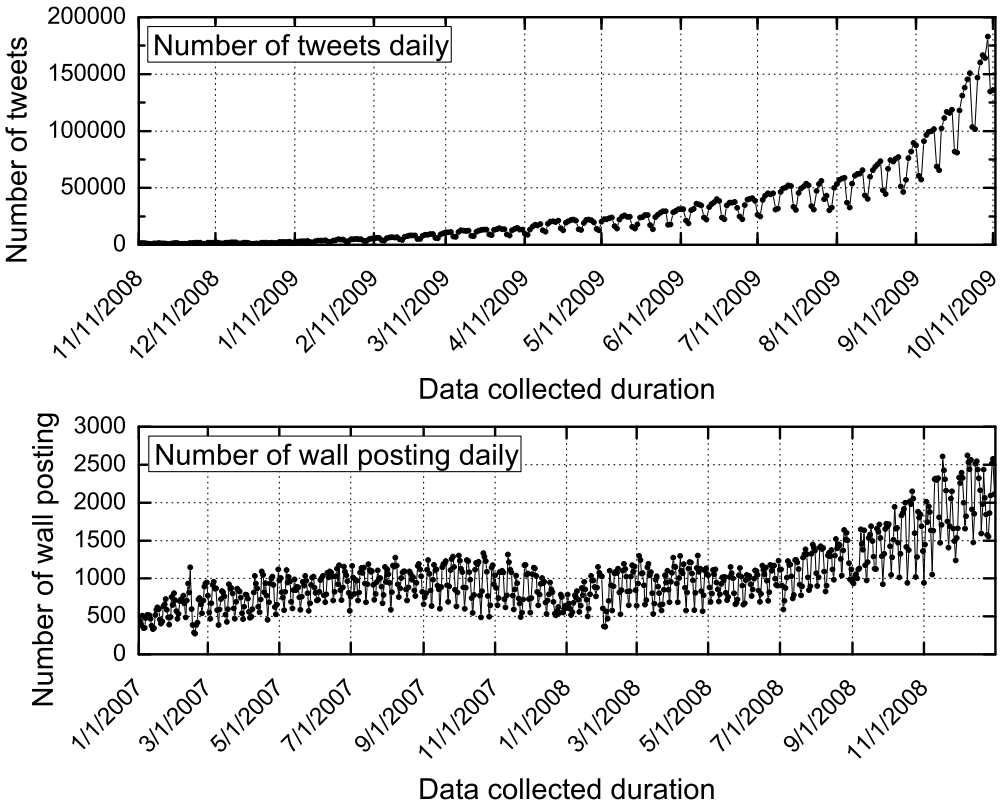
The Twitter datasets we study are the Twitter social graph [85] and Tweets [144]. The Twitter social graph dataset [85] includes profiles of 41.7 million users who are connected by 1,468,365,182 following-followed links with an average number of followers of 36.6146. The tweets dataset [144] includes 10 million real tweets that were collected from Nov.11th, 2008 to Oct.11th, 2009.

2.6.2 Social Traffic Patterns

There are two social traffic aspects we are interested in: *(i)* the average daily social traffic increase ratio; *(ii)* the distribution of number of social updates per user. These aspects describe the global social traffic pattern within the social networks, as well as and individual user's communication pattern. We can utilize them to design the social cache selection algorithm, as well as for generating synthetic social traffic.

The number of daily tweets and wall posts for Twitter and Facebook is shown in Figure 2.5. It is clear that the number of tweets and wall posts increases over time. Specifically, the number of daily wall postings on Facebook increases 464% in two years, and the number of daily tweets increases 1397% in less than one year. We believe that number of tweets increases faster than that of wall posings because tweet is the major communication method for Twitter, while wall posting is only one of the communication methods in Facebook. As the number of daily social traffic continously increases, the naive "push-to-all-friends-immediately" method will not scale, but makes the decentralized social network a "distributed deny of service" network.

Furthermore, we study the distribution of the number of social updates a user produces and the number of users who sent that number of social updates, and plot it in Figure 2.6. This figure shows a typical scale-free power law distribution, meaning the majority of social updates were sent by a minority of social network users. As our goal is to reduce the social connections within the social networks, the social cache selection will be affected by the power law distribution of social updates per person. If the distribution is uniform, we can place the social caches anywhere, However, power law distribution gives us a chance: social connections will be greatly reduced if one can carefully and intelligently place the social caches. One can also use this power law distribution as a model to simulate synthetic social traffic.



The x axes represent the duration of data collect, and the y axes show the number of social updates by means of tweets for Twitter and wall posts for Facebook. The upper and lower figures show daily tweets and daily wall postings, respectively, during the data collection period. Both show clearly increasing trends.

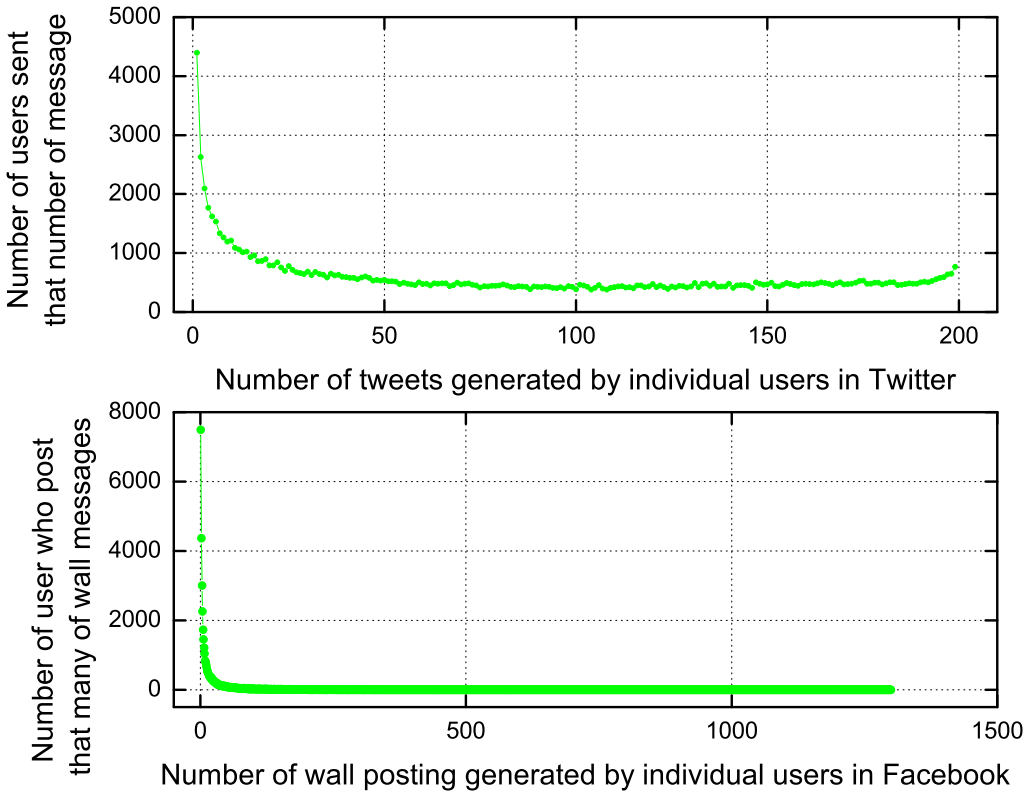
Figure 2.5: Daily social traffic trend for Twitter and Facebook.

2.7 Evaluation

We first compare the performance of the two caching algorithms against various metrics such as number of caches selected and cluster size by using data available from Facebook [147]. We then evaluate how social traffic could be reduced using caches, when compared with other data dissemination methods.

2.7.1 Social Score Algorithm Properties

Before comparing the two algorithms, we first present some results about social score algorithm. As discussed in Section 2.5.3, social score is calculated based on clustering coefficient,



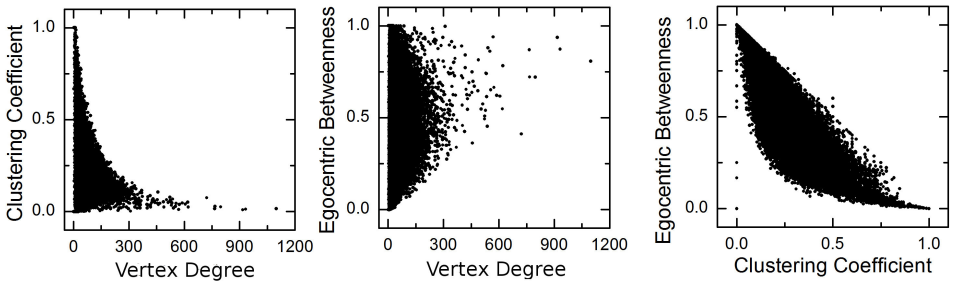
The x axes represent the number of tweets and wall postings, respectively, sent per user during the period of data collection. The y axes indicate how many users sent that many updates for Twitter and Facebook. The plots follow a power law distribution for both datasets.

Figure 2.6: Social updates frequency for Twitter and Facebook.

egocentric betweenness centrality, and degree. The correlations between the three factors for the Facebook dataset have been plotted in Figure. 2.7. The figure on the left shows that as vertex degree increases, the range of clustering coefficient shrinks exponentially from [0,1] towards 0. The figure in the middle shows that as vertex degree increases, the range of egocentric betweenness centrality shrinks from [0,1] towards 1. This means that a vertex with a higher degree has a lower value of clustering coefficient and a higher egocentric betweenness centrality. The figure on the right shows that most vertices are clustered around the line of

$$EBC = 1 - CC. \tag{2.2}$$

Figure 2.8 shows social scores for the Facebook dataset in the space composed of degree,



Correlations between vertex degree, clustering coefficient and egocentric betweenness centrality for Facebook data. Range of clustering coefficient shrinks exponentially towards 0 as degree increases (on the left). Egocentric betweenness centrality shrinks towards 1 as degree increase (in the middle). Most vertices are clustered around the line of $EBC = 1 - CC$ (on the right).

Figure 2.7: Correlations between vertex degree, CC and EBC.

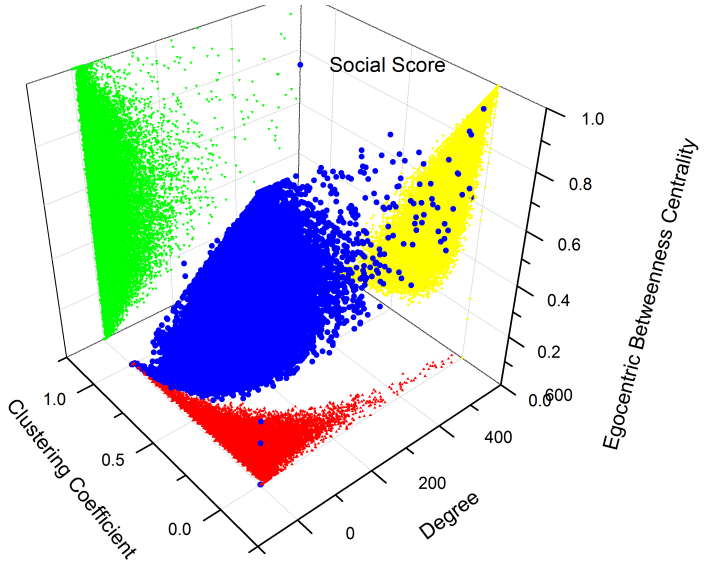
clustering coefficient, and egocentric betweenness centrality as x, y, z axes. The blue dots are the social scores in this coordinate system. The red dots are the trajectory on the x-y surface, the yellow dots are the trajectory on the x-z surface, and the green dots are the trajectory on the y-z surface.

Nodes with higher social scores have higher egocentric betweenness centrality, and smaller clustering coefficient but a larger degree. They are connected to a large number of non-interconnected vertices, and are candidates for social caches.

2.7.2 Comparison of the Two Algorithms

We evaluated both algorithms by using the dataset obtained from Facebook [147], and compared the selection of social caches, as well as the corresponding social clusters.

The statistics on the total number of social caches selected and the number of social caches each vertex connects by applying the two algorithms are listed in Table 2.3. The social score algorithm selects about 1.5 times the total number of caches that the approximate NDS algorithm does. The number of caches a vertex connects determines the cost of sending or requesting social updates. Both algorithms yield relatively equal average numbers of social caches, which means that they should generate relatively equal social traffic. The cumulative distribution of the number of social caches a vertex connected is shown in Figure 2.9. The CDF for both



Social score distribution in the coordinate systems of vertex degree, clustering coefficient, egocentric betweenness centrality as x, y, z axes respectively. The blue dots are the social scores, red dots are the trajectory on the x-y surface, yellow dots are the trajectory on the x-z surface, and the green dots are the trajectory on the y-z surface.

Figure 2.8: Social score distribution in 3D coordinate system.

indicate that they converge fast, with each vertex connecting to less than 15 social caches; the caches selected by the algorithms can cover 90% of the network, and with each vertex connecting to fewer than 25 social caches, 99% of the entire network can be covered. The approximate NDS algorithm converges faster with the given dataset.

	Algorithm	
No. of social caches	**Social Score**	**Approximate NDS**
Total no.	40782	26288
Avg no. a vertex connects	7.058(5.991)	5.867(5.434)
Max no. a vertex connects	61	48
Min no. a vertex connects	0	1

Table 2.3: Statistics of social caches selected for the two algorithms.

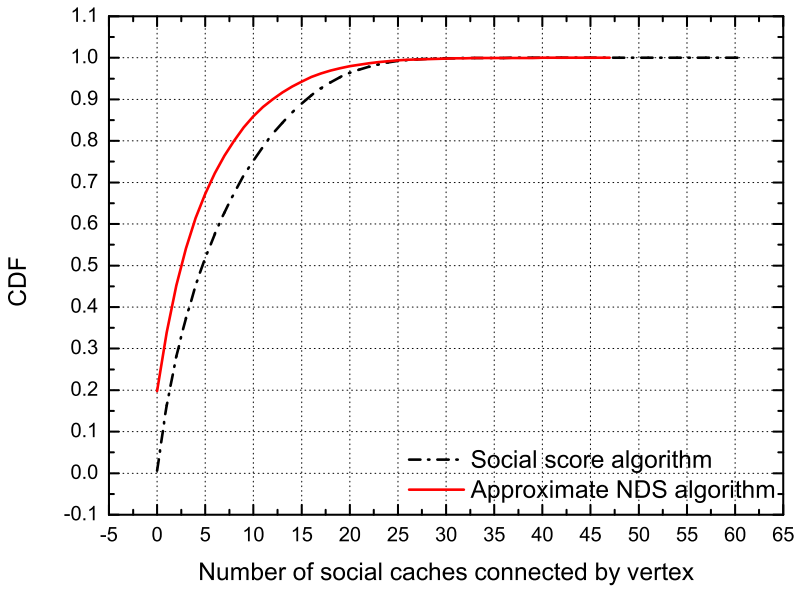


Figure 2.9: CDF for number of social caches that each vertex connects.

Table 2.4 presents the statistics about social cluster size for the algorithms. Cluster size measures the number of nodes needed to contact the same social cache for social updates, and therefore the social traffic a cache needs to handle. The average social cluster size is similar for the two algorithms. Figure 2.10 shows the CDF of social cluster size for both algorithms: both have a few large social clusters ($>$ 1000 members) exhibiting a long-tail distribution. Social caches associated with large social clusters are required to handle significant social traffic from members, which should be avoided.

	Algorithm	
Social cluster size	**Social Score**	**Approximate NDS**
Average size	11.424 (24.491)	13.224 (34.418)
Median size	4	4
Maximum size	1098	1098
Minimum size	1	1

Table 2.4: Statistics of social cluster size for the two algorithms.

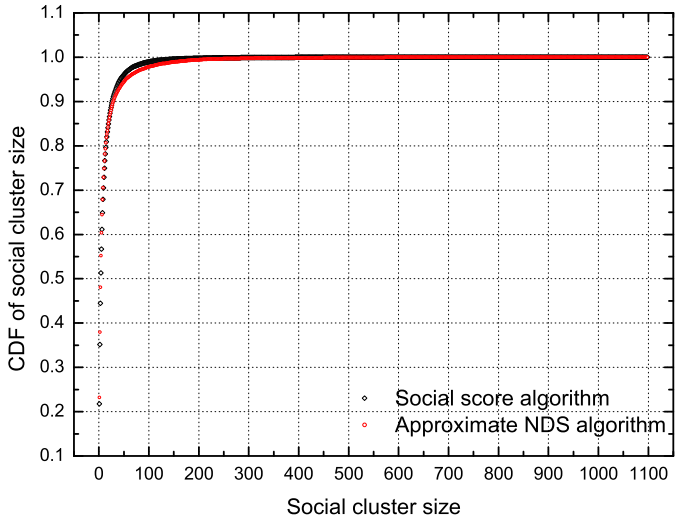


Figure 2.10: CDF of social cluster size for the two algorithms.

2.7.3 Social Traffic Comparison

We evaluate the effectiveness of the two proposed algorithms by determining the social traffic generated in comparison to other social network data dissemination methods, such as "push-to-all-friends" in a distributed social network. In particular, we compare the following six methods:

a. Push social updates immediately to all friends.

b. Push periodically (every m minutes) to all friends. In this scenario, the amount of social traffic does not depend on how many social updates are generated by vertices, but on how many friends each vertex has and the time interval m. In another words, it depends on how many edges

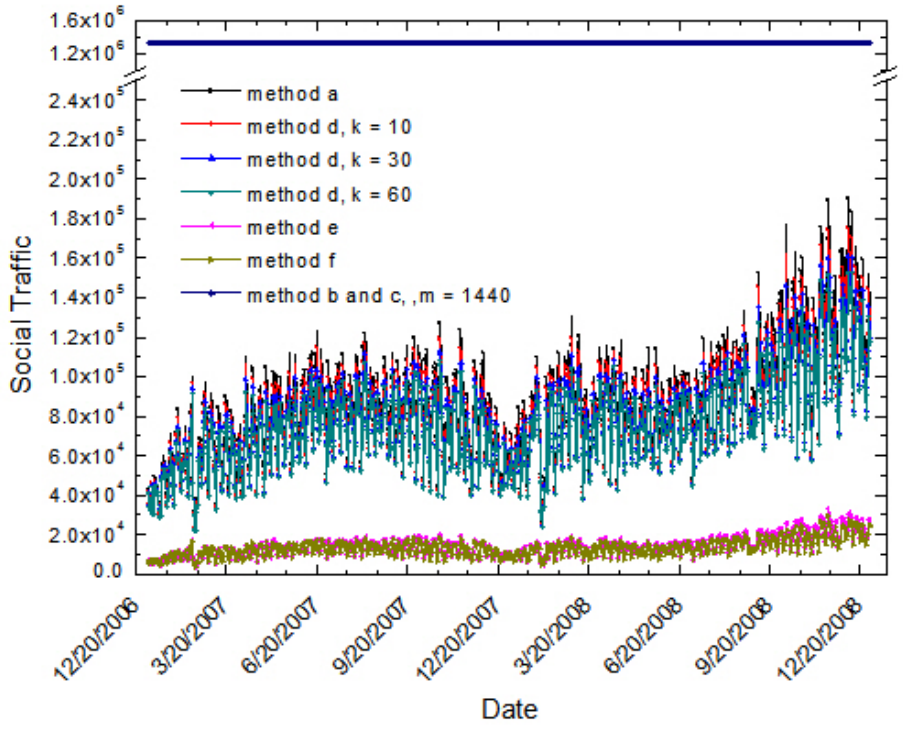
are there in the graph. Therefore, social traffic is calculated as $ST = 24 * 60/m * 2 * |E|$

c. Pull periodically from friends. Same as above.

d. Push periodically (every k minutes) to all friends if the vertex has updates. If a node updates two or more times during the k minutes, only one connection needs to be set up. This will reduce the total network traffic in the way of local caching.

e. Social Score algorithm.

f. Approximate NDS algorithm.



The overall social traffic generated daily utilizing data dissemination method a, methods b and c with m = 1440, method d with k = 10, method d with k = 30, method d with k = 60, and methods e and f.

Figure 2.11: The overall social traffic generated by different data dissemination methods.

We evaluate the above six algorithms by utilizing the Facebook dataset. For each wall posting, we simulate generating social traffic by pushing or pulling defined by the methods. The total daily social traffic is plotted in Figure 2.11. The first straight line is the plot for methods b and c. We choose m = 1440 minutes, which is the total number of minutes in a day.

The result shows that even with push or pull once per day between each pair of friends, the total network traffic is still extremely high compared with other methods. The next set of lines in the graph are for methods a and d, with k = 10, 30, and 60 minutes. The local caching mechanism used in method d reduces the social traffic by 4.68%, 8.79%, and 11.92%. The bottom set of lines shows the results by utilizing social caching methods, e and f, which can effectively reduce the overall social traffic by 83.90% and 84.92%, respectively, compared with method a. The performance of the two algorithms is similar given the current Facebook dataset. We anticipate that, with different social graphs and social traffic patterns, the performance will be different.

2.8 Summary

In this chapter, we introduced Social Butterfly, a novel peer-to-peer information dissemination model for distributed social networks based on social caches. We propose the Neighbor-Dominating Set problem as a formal definition of the social cache selection problem and present two algorithms for it. Evaluation with social traffic data made available from Facebook shows that the use of social caches reduces total social traffic by up to 80%.

The social cache selection algorithms we discussed in this chapter are centralized algorithms based on knowledge of the entire underlying topology of the social graph, and assume the selected social caches are available all the time. Issues such as social traffic patterns, social tie-strength [58], and availability models need to be considered. In the next chapter, we will introduce the distributed algorithms for social cache selection.

Chapter 3

Distributed Algorithms For Social Cache Selection

In Chapter 2, we proposed *Social Caches*, which are nodes selected to act as local bridges for their friends in order to reduce the number of connections necessary to collect social updates in DOSNs. We also propose *Social Butterfly* communication model for efficient social update dissemination by utilizing *Social Caches*. However, the solution was not fully distributed, as it required knowledge about the entire social graph topology.

In this chapter, we present **SocialCDN**, an efficient social content distribution system based on a set of novel, fully distributed social cache selection mechanisms that does not require global knowledge of the underlying social graph. Within the context of SocialCDN, we propose and analyze four distributed cache selection algorithms: *Randomized* algorithm, *Triad Elimination* algorithm, *Span Elimination* algorithm, and *Social Score* algorithm. The Randomized algorithm, used as a baseline for comparison, elects caches based on a uniform probability. In the two elimination algorithms (Triad Elimination and Span Elimination algorithms), each pair of nodes selects a cache in the local neighborhood in a greedy manner and afterwards minimizes the total number of caches. These algorithms are applicable to any arbitrary graph. The *Social Score* algorithm is designed specifically for social graphs and takes into account measures such as the centrality of a node in its local network. A node decides whether it will become a social cache or not based on its local properties.

We evaluate the performance of these four cache selection algorithms on five different graphs that can be grouped into three categories: *Social, Semi-Social, and Non-Social Network*, and discover that the proposed algorithms perform almost as well as the centralized best approximating algorithm, *Approximate NDS algorithm* [65] (discussed in Section 2.5.2) would do. Besides the quantitative differences that we analyze extensively through simulation, we also discuss the qualitative differences of the proposed algorithms.

Existing distributed data dissemination approaches for DOSNs include [113] and My3 [109]. In [113], the authors proposed a P2P OSN that assumes that a user's updates will reach another user if there is a path in the overlay network between the two users. Sending content through several links makes the system more vulnerable to failures and requires stronger incentives for intermediary peers to store and transfer the content. In our Social Caching approach, content can reach the other party through a common friend, which means within at most two-hops. My3 [109] is a P2P based OSN that relies on users' geographical locations and online time statistics to provide availability. It uses the original Dominating Set (DS) problem to minimize the number of replicas. Our method does not consider availability. Instead, we use the Neighbor Dominating Set (NDS) problem to minimize the number of selected cache nodes in order to improve social update dissemination efficiency. NDS is sufficiently different from the DS problem to render the known DS approximation algorithms unusable in the NDS case.

There exist other data dissemination schemas such as gossip protocol [101], epidemic routing [145] and probabilistic routing based on prediction [89]. Giuliano Mega, et al. [101] propose a modified gossip protocol for data dissemination in DOSN. They utilize vertex centrality and clustering to select the recipient of the gossip message. However, gossip protocol and epidemic routing schema rely on a flooding technique, which does not help to reduce social traffic. Prediction-based methods do not work for social update distribution, since publishing and browsing in OSNs, although they usually follow certain daily patterns, are very hard to predict.

We discuss the SocialCDN system model and approach in Section 3.1. Section 3.2 clarifies the notations used in the chapter. Section 3.3 presents the four distributed social cache selection algorithms. In Section 3.4 we discuss the properties and measurements of the five social graphs we used for evaluation. The evaluation of the four methods using the social graphs is presented in Section 3.5. Section 3.6 presents the mechanism to maintain cache consistency. We discuss open questions for socialCDN in Section 3.7, and Section 3.8 concludes the chapter.

3.1 SocialCDN

In this section, we first present the underlying models on which the SocialCDN relies. The second half of this section presents the SocialCDN approach for social update dissemination in distributed OSNs.

3.1.1 Model

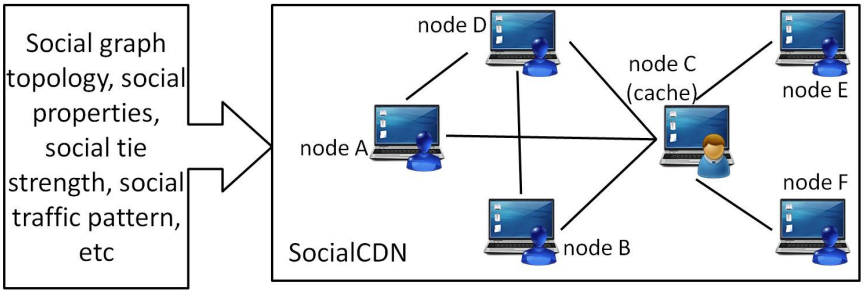
SocialCDN works by making the following assumptions about the underlying communication network, social graph, and trust:

- **Network and Communication.** For simplicity's sake, we assume a static network and a standard synchronous round-based distributed communication model. In particular, we assume that, once selected, the cache nodes are always available.
- **Social Graph.** SocialCDN assumes that each node knows its immediate friends and its two-hop friends. Knowing two-hop neighbors is a common feature in today's OSNs that allows users to see friends-of-friends' comments, and to expand their networks of friends. We also assume that social links in the social graph are equal; and that there are no edge weights to represent social tie strength, distance, delays or communication loads.
- **Trust and Altruism.** SocialCDN assumes friends are trust worthy and altruistic. In other words, they do not misbehave and are willing to cache content for their friends without compensation. Furthermore, we assume that the social cache altruism applies to friends only, for whom personal bandwidth and storage can be sacrificed. Reputation and economic models are outside the scope of this dissertation.

With all these simplifying assumptions, our problem is still an NP-complete optimization problem.

3.1.2 Approach

Within a pure DOSN, dissemination of social updates among friends necessitates that $O(n^2)$ (where n is the number of friends) network connections be established, which can significantly

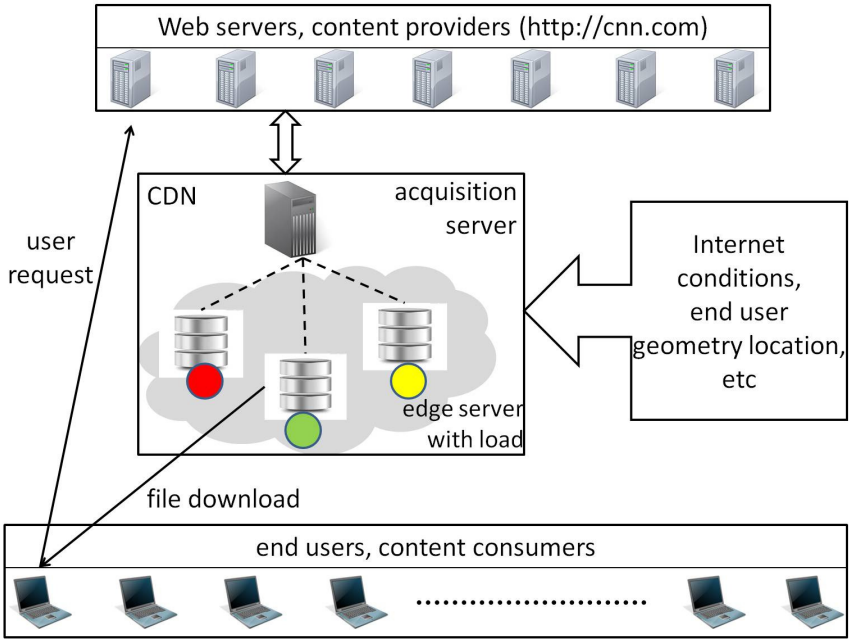


An example of SocialCDN network with six nodes, where node C is selected as the social cache in this example. Nodes A, B, D, E, and F send their social updates to node C, and fetch their friends' updates from node C as well. The system takes social graph topology, social properties, social tie strength, social traffic pattern, etc. (in the box to the left) as input parameters for deciding the cache.

Figure 3.1: An example of a SocialCDN.

degrade the performance compared to a centralized DSN. SocialCDN utilizes social caches to reduce the total number of P2P connections necessary for social update dissemination within a DOSN. Nodes push their social updates to, and fetch those of their friends from the social caches they are connected to. If the number of selected caches is significantly less than the number of friends, then the use of social caches will significantly reduce the total connections when compared to a fully P2P pull/push approach. This is why, in this dissertation, the goal is to minimize the number social caches using fully distributed algorithms. Therefore, the following constraint that we discussed in Chapter 2 still holds when selecting social caches distributedly: *social caches are a subset of vertices in the graph, where a vertex is either a social cache or connected to a social cache, and where any pair of friends must have at least one common friend who is a social cache, if neither of them is a social cache.*

Although both SocialCDN and CDN rely on caching schemas for content delivery, the selection methods they employ are completely different. CDN technology selects the edge server depending on the geographic location of the users, edge server traffic loads, and network conditions such as bandwidth, as shown in Figure 3.2. SocialCDN decides the placement of social caches based on social graph topology, social properties, social tie strength, social traffic pattern, etc., as shown in Figure 3.1.



An example architecture of a Content Delivery Network (CDN). The graph illustrates how the CDN system selected an edge server to serve the end user request. The edge server selection is decided by the current Internet conditions, the end users geometry location, etc., as shown in the box to the right.

Figure 3.2: An example architecture of Content Delivery Network(CDN).

3.2 Notation

In general, we use $G = (V, E)$ to represent an undirected graph, where V is the set of vertices, and E is the set of edges. The terms *social network user*, *vertex* and *node* are used interchangeably to represent a vertex in the graph.

We also define the following notations:

- $deg(v)$ to be the degree of node v.
- $N(v)$ to be node v's one-hop neighbors.
- The set of *edges covered by node* v, S_v, is the subset of E and is composed of any $e = (\alpha, \beta) \in E$ iff $v \in (N(\alpha) \cap N(\beta)) \cup \{\alpha, \beta\}$.
- $size(S_v)$ denotes the number of edges in S_v. An edge e could be covered by multiple nodes depending on the topology of the graph.

- $T(v)$ is the number of *Transitive Triads* a node v is part of. One of the basic units of social network theory is the *Dyad*, which is a pair of parties who may or may not share a social relation. A *Triad* is a set of three parties and consists of three dyads. A triad is *transitive* if when there is a tie between party A and party B and between B and a third party C, then there is also a tie between A and C.
- $CN(u, v)$ or $CN(e)$ denotes the set of common neighbors of edge $e = (u, v)$.

During the execution of a cache selection procedure:

- A node v belongs to one of the following categories:

  i) *Black*: v is selected as a social cache;

  ii) *Grey*: every edge in S_v is *covered* by social caches, but v is not selected as a social cache;

  iii) *White*: v is not a social cache and there is at least one edge in S_v that has not been covered.

- The edges covered by a social cache are *green* edges, others (uncovered edges) are *red* edges.
- Define $span(v)$ to be the number of *red* edges in S_v. At the beginning of a selection procedure, $span(v) = size(S_v)$, but it decreases as the algorithm executes, and will be 0 when the algorithm terminates.

3.3 Distributed Social Cache Algorithms

In this section, we present four distributed social cache selection algorithms that will be used to solve the following cache selection problem: *find the smallest set of cache nodes such that each edge is connected by at least one social cache if none of its endpoints is a social cache.* The problem is also formulated as a *Neighbor-Dominating Set* problem [65] in Section 2.5.1 and is defined as:

> " The *Neighbor-Dominating Set* of graph $G = (V, E)$ is the set $S \subseteq V$ of vertices such that for each edge $(u, v) \in E$, there exists a $w \in S$ satisfying $w \in (N(u) \cap$

$N(v)) \cup \{u, v\}$. Given a graph $G = (V, E)$, find a *Neighbor-Dominating Set* of smallest size."

In the *Randomized* algorithm, each node elects itself as a social cache with a probability of θ, which is predefined. We use this method mainly as a baseline for evaluation and comparison with the other three. *Triad Elimination* and *Span Elimination* algorithms are elimination methods that work in two phases. During the first phase, a social cache for every edge is selected based on how many *transitivity triads* a node is a part of, or *span* of a node, respectively. During the elimination phase, redundant caches are reduced as much as possible. The *Social Score* algorithm works by utilizing the *social score probability*, which is derived from the social score [65] that has been discussed in Section 2.5.3: a measure that captures the centrality of a node in its local network. [1]

3.3.1 Randomized Algorithm

We use the Randomized algorithm as a baseline for evaluation and comparison with the other three algorithms. The algorithm works by letting node v elect itself as a social cache with a threshold probability of θ. More precisely, each node applies the distributed algorithm in the following steps until all edges in the graph are colored *green*:

a. calculate $span(v)$,

b. if $span(v)$ == 0, (meaning that the edges in S_v are all marked as *green*), node v makes itself as *grey* and quits the loop.

c. if $span(v) > 0$, generate a number $p(v) \in [0, 1]$ uniformly at random. If $p(v) > \theta$, node v elects itself as a social cache, marks itself as *black* and all edges in S_v as *green*, informs its neighbors about its election and quits the loop.

The pseudo code for the algorithm is given in 3.

We use the graph in Figure 3.3 as an example to explain all the distributed social cache selection methods. Given the social graph in Figure 3.3, we assume that node i generates a random number $p(i)$ in each iteration. Consider the following scenario: $p(3) > \theta$, and

[1] The Social Score algorithm in this Chapter is a fully distributed algorithm, with is different from the one discussed in Section 2.5.3, although they share the same name.

Algorithm 3: Randomized Algorithm

```
while color(v) == white do
  count span(v);
  if span(v) == 0 then
    mark v as grey
  end if
  if span(v) > 0 then
    p(v) = generate a random prob
    if p(v) > θ then
      mark v as black
      mark edges in S_v as green
    end if
  end if
end while
```

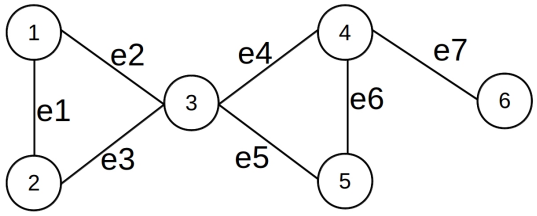


Figure 3.3: Graph to illustrate distributed cache selection algorithms.

$p(i) < \theta$ for $i = 1, 2, 4, 5, 6$. In this case, during the first iteration, node 3 elects itself as a social cache, and marks edges $\{e1, e2, e3, e4, e5, e6\}$ as *green*. Therefore, randomly electing a node from node $\{4, 6\}$ will cover the whole graph. It is clear that the performance of this method is determined by the predefined θ. The performance evaluation will be discussed in Section 3.5.1.

3.3.2 Triad Elimination Algorithm

Both the Triad Elimination algorithm and the Span Elimination algorithm (Section 3.3.3) have two phases: the *selection* phase, and the *elimination* phase. During the *selection* phase, a social cache is selected for every edge based on how many *transitivity triads* the node is a part of, or on the *span* of a node, respectively. During the *elimination* phase, the redundant caches are reduced as much as possible. Both algorithms terminate within a **constant** number of rounds: two rounds. We will discuss the Transitive Triad elimination algorithm in this section, and the Span elimination method in the next Section.

In the *cache selection* phase, each node v calculates $T(v)$ as the number of transitive triads it is part of. Note that it is relatively easy to figure out how many triads a node is part of once the node knows its two-hop neighbors. For each edge $e = (u, v)$, nodes u and v exchange their $T(v)$ and $T(u)$, and select the one with higher T to be the *Temporary Social Cache* for the edge, as $TSC(e)$. In the case $T(u) = T(v)$, the choice is made randomly. Each node v will keep track of the status of every edge $e \in S_v$, and the $TSC(e)$ associated with the edge.

Node	1	2	3	4	5	6
Transitive Triads, $T(v)$	1	1	2	1	1	0
Size of span, $size(S_v)$	3	3	6	4	3	1

Table 3.1: $T(v)$ and $size(S_v)$ of each node for the graph in Figure 3.3.

The selection phase enables all edges to be covered, i.e., *green*, however, the number of caches is not optimal. For example, in a graph with three nodes A, B and C forming a transitive triad, it is possible to select all three nodes as caches in the worst case, as $T(v) = 1$ for all of them. In fact, choosing one node as the cache is the optimal solution for this graph. This observation gives us insight into the elimination: if two nodes of an edge have common neighbors, we can compare among all common neighbors to select the one that has a higher span to be the social cache.

The *elimination* phase reduces redundancy by utilizing the fact that *every common neighbor of an edge can also be a cache for that edge, besides the two endpoint nodes*. The temporary cache for each edge contacts all the common neighbors of $e = (u, v)$, checks how many times each of them has been selected during the *selection* phase, and chooses the one that has been selected the most number of times as the final social cache for that edge. More precisely, the temporary cache $TSC(e)$ for each edge $e = (u, v)$ compares the number of times it has been selected $freq(TSC(e))$ with $freq(w)$ for every $w \in CN(u, v)$. Node $n \in \{u, v\} \cup (N(u) \cap N(v))$ with the highest $freq(n)$ will be selected as a cache for that edge. Node n marks itself as *black*, marks edges in S_n as *green*, informs its friends about its selection, and terminates the algorithm.

Given the graph shown in Figure 3.3, the number of transitive triads for each node v is listed in Table 3.1. The selected $TSC(e)$ and the common neighbors $CN(e)$ are listed in Table 3.2. Since $T(1) = T(2)$, we select node 1 (uniformly at random) as TSC for edge $e1$. A similar

Edge	e1	e2	e3	e4	e5	e6	e7
$CN(e)$	{3}	{2}	{1}	{5}	{4}	{3}	{}
Triad Elimination Method							
$TSC(e)$	1	3	3	3	3	4	4
Cache selected	3	3	3	3	3	3	4
Span Elimination Method							
$TSC(e)$	3	3	3	3	3	3	4
Cache selected	3	3	3	3	3	3	4

Temporary social cache and final social cache selected utilizing the two elimination algorithms. $TSC(e)$ selected are different for the two algorithms, but the final cache selection is the same.

Table 3.2: Measurements for the elimination algorithms.

situation happens for edge $e6$, where node 4 is selected as a temporary social cache. Further, node 3 is selected as a TSC for edges $e2, e3, e4$ and $e5$, since $T(3) > T(1), T(2), T(4), T(5)$. Finally, $T(4) = 1 > T(6) = 0$, and node 4 will be selected for edge $e7$. During the elimination phase, node 3 is selected for edges $\{e1, e2, e3, e4, e5, e6\}$, and node 4 is selected as for edge $e7$ based on frequency. The results are listed in Table 3.2.

3.3.3 Span Elimination Algorithm

As in the Triad Elimination algorithm, each node v initially calculates S_v, the set of edges that it covers. During the *selection* phase, nodes u and v of each edge $e = (u, v)$ exchange $size(S_u)$ and $size(S_v)$, and select the node with the higher value to be a temporary cache. The selected node further contacts the common neighbors of edge e, compares $size(S_n)$ with every node $n \in \{u, v\} \cup (N(u) \cap N(v))$, and selects the node w that has the largest $size(S_w)$ as $TSC(e)$.

The *elimination* phase is similar to the Triad Elimination algorithm. The temporary cache $TSC(e)$ for each edge e contacts every node $w \in CN(u, v)$, compares the $freq(TSC(e))$ with $freq(w)$ and selects node $n \in \{u, v\} \cup (N(u) \cap N(v))$ that has the highest $freq(n)$. Node n marks itself as *black*, marks edges in S_n as *green*, informs its neighbors, and terminates the algorithm.

For the graph shown in Figure 3.3, Table 3.1 lists $size(S_v)$ for each node v. Both nodes 1 and 2 cover three edges, and their common neighbor $CN(1, 2)$ is node 3, which covers

six edges. Therefore, node 3 is selected as $TSC(e1)$. Node 3 is also selected as a temporary social cache for edges $e2, e3, e4$, and $e5$. Since $S_4 = 4 > S_6 = 1$, node 4 is selected for edge $e7$,. During the elimination phase, node 3 is selected as the social cache for edges $\{e1, e2, e3, e4, e5, e6\}$ since it has the highest selection frequency. Node 4 remains the cache for edge $e7$. The results are shown in Table 3.2 (in the Span Elimination section). Note that the $TSCs$ selected by the *Span Elimination* algorithm are the same as the final caches, indicating that Phase 1 alone is efficient in cache selection.

3.3.4 Distributed Social Score Algorithm

The *Social Score* algorithm selects social caches based on a node's *Social Score Probability*, $ss\_prob(v)$, which is calculated according to equation 3.1. The $ss(v)$ in the formula is the *Social Score* (discussed in Section 2.5.3) of node v to measure the centrality of a node in its local network.

$$ss\_prob(v) = \begin{cases} 1 - 1/ss(v) & \text{if ss(v)} \geq 1 \\ 0 & \text{if ss(v)} < 1 \end{cases} \tag{3.1}$$

The *social score* of a node is a combination of the Clustering Coefficient $cc(v)$ [71], the Egocentric Betweenness Centrality $ebc(v)$ [97] and the vertex degree, and is defined by equation 3.2.

$$ss(v) = [(1 - cc(v)) + ebc(v)] * deg(v) \tag{3.2}$$

The Clustering Coefficient quantifies how well connected are the neighbors of a vertex in a graph, and is defined as in Equation 3.3:

$$C_i = \frac{2T(i)}{deg(i)(deg(i) - 1)} \tag{3.3}$$

where $T(i)$ is the number of transitive triads node i is part of. An Egocentric network is a "local" network consisting of an individual node and its immediate neighbors. Betweenness

Centrality measures the influence a node has over the spread of information through the network, and is defined by Equation 3.4:

$$BC(i) = \sum_{s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}} \tag{3.4}$$

where σ_{st} is the total number of shortest paths from node s to t, and $\sigma_{st}(i)$ is the number of those paths that pass through i. Egocentric Betweenness Centrality is the betweenness centrality of a vertex in its egocentric network. It also indicates how many additional pairs of nodes among the neighbors could be connected if the vertex were selected as a social cache. Given the assumption that each node v knows its two-hop neighbors, the clustering coefficient and egocentric betweenness centrality can be locally calculated. Therefore, $ss(v)$ and $ss\_prov(v)$ can be calculated using local information only.

Algorithm 4: Social Score Algorithm - Phase 1

```
calculate ss_prob(v)
if ss_prob(v) > ρ then
   mark itself as black (*social cache*)
   mark edges in S_v as green
   inform every node in N(v)
end if
```

Algorithm 5: Social Score Algorithm - Phase 2

```
while span(v) > 0 do
   calculate ratio(v)
   if (ratio(v) > γ) then
      mark v as black (*social cache*)
      make red edges in S_v as green
   else
      γ− = RATIO_STEPSIZE
      recalculate span(v)
   end if
end while
```

The social score algorithm executes in two stages by utilizing two predefined variables, the *threshold probability*, as ρ, and the *ratio threshold*, as γ. First, each node v calculates its $ss(v)$ and $ss\_prob(v)$, and elects itself as a social cache if $ss\_prob(v) > \rho$, then marks itself as *black*, marks edges in S_v as *green*, and informs its neighbors as presented in Algorithm 4. Next, each

node that has not been elected executes the following steps in iterations locally. In each loop, node v re-calculates its $span(v)$ and color. If every edge in S_v is *green* and thus $span(v)$ = 0, node v marks itself as *grey* and exits the loop. Any node v with $span(v) > 0$ calculates a ratio, *ratio*, $ratio(v)$, which is defined in Equation 3.5. If $ratio(v) > \gamma$, node v elects itself as a social cache, marks itself as *black*, marks *red* edges in S_v to *green*, and notifies its neighbors as shown in Algorithm 5.

$$ratio(v) = span(v)/size(S_v) \tag{3.5}$$

In the second stage, $span(v)$ either decreases or remains the same after each iteration. If node v is elected, $span(v)$ becomes zero. If an edge in S_v is marked as *green* (by another social cache), $span(v)$ decreases. Otherwise, $span(v)$ remains the same. $ratio(v)$ changes with $span(v)$ according to equation 3.5 ,and eventually the algorithm stops once $ratio(v) \leq \gamma$ for node v. Therefore, γ needs to be decreased by step size $RATIO\_STEPSIZE$ after each iteration to cover every edge in the graph, as is shown in Algorithm 5

Node	Social Score	Social Score Probability
1	0.0	0
2	0.0	0
3	16/3	13/16
4	12/3	9/12
5	0.0	0
6	1.0	0

Table 3.3: $ss(v)$ and $ss\_prob(v)$ for each node in Figure 3.3.

Consider again the graph shown in Figure 3.3, the social scores and the corresponding probabilities are listed in Table 3.3. If we set the ρ to be 0.75, nodes 3 and 4 will be elected as social caches during the first stage. In the second stage, nodes 1, 2, 5, 6 re-calculate their *spans*, which are now equal to zero. They mark themselves as *grey* and terminate the algorithm.

3.3.5 Algorithm Complexity

As we discussed in Section 3.1, we utilize a standard synchronous model for computation. During a communication step (or round), each node can exchange one message with each

of its neighbors. The time complexity of the algorithm is measured by the total number of communication steps.

Both the Triad Elimination and Span Elimination algorithms terminate in two rounds. In the first round, each pair of connected nodes u, v exchange their $T(u)$, $T(v)$ or $size(S_u)$, $size(S_v)$ values respectively, to determine the temporary social caches. Note that even for the Span Elimination algorithm, since every node u exchanges $size(S_u)$ with every neighbor, the information is sufficient to elect a temporary cache. In the second round, each temporary social cache $TSC(e)$ exchanges how many times it has been selected with all common neighbors of u and v to make a final decision. Therefore, the algorithms terminate in two rounds.

The Social Score algorithm terminates in: $1 + 1/RATIO\_STEPSIZE$ rounds. In the first round, each node decides whether it has been elected as a social cache by comparing its social score with ρ, and informs its friends about the election. Next, any node that has not been elected executes Algorithm 5, where γ is first set to 1, and decreases by $RATIO\_STEPSIZE$ in each iteration until the algorithm terminates.

The time complexity in terms of rounds and communication complexity (messages) for each algorithm are listed in Table 3.4.

Algorithm	**Time Complexity**	**Communication Complexity**
Transitive Triads	2	$4\|E\|$
Span	2	$4\|E\|$
Social Score	$1 + \frac{1}{RATIO\_STEPSIZE}$	$\|E\| + \frac{\|E\|}{RATIO\_STEPSIZE}$

Time complexity in number of rounds, and the communication complexity in number of messages exchanged for Transitive Triad, Span Elimination and Social Score algorithms.

Table 3.4: Time and communication complexity for distributed algorithms.

3.4 Graph and Social Properties

To evaluate the proposed distributed social cache selection algorithms, we choose five widely used graphs, namely, Facebook graph [147], Enron email graph [81], Coauthor graph [29], Citation graph [27], and Autonomous Systems networks graph [88].

We will present graph characteristics for each dataset, and discuss the classifications of the graphs in Section 3.4.1. Since cache selection algorithms utilize diverse social properties, we will discuss them for each graph in Section 3.4.2.

3.4.1 Dataset Descriptions

Social and Non-Social Graph

We consider the selected graphs as un-directed graphs that fall into three categories: *Social Graph, Semi-Social Graph*, and *Non-Social Graph*.

Facebook, as one of the most popular OSNs, is a typical *Social Graph*, and is used for personal socialization and communication. The Enron graph [81] represents social connections, but only when one person sends an email to another during the data collection period. The Facebook graph represents *cumulative* social connections from the day the user registers with Facebook until the data is collected; while the Enron graph only illustrates *periodical* social connections during the crawling duration for the dataset. Therefore, we consider the Enron graph to be a *Semi-Social Networks* graph. The Coauthor graph [29] and the Citation graph [27] are also *Semi-Social Networks*: the Coauthor graph shows how authors collaborate to produce papers, while the Citation graph shows how papers cite each other. The coauthor graph may show high clustering property, since researchers from the same institutes/labs are more likely to collaborate. Both of them only show partial connections between the human's and the paper's professional life. They give partial information about a person's scientific relationship with other scientists. The graph of routers comprising the Internet can be organized into sub-graphs called Autonomous Systems (AS). Each AS exchanges traffic flows with some neighbors (peers), and is considered a *Non-Social Graph*.

Bipartite and Unipartite Graph

Facebook represents a social graph with a single type of node, *users*, where the edges are the direct relationships between the nodes. On the other hand, the Enron graph can be regarded as derived from an Enron *Bipartite graph*, which includes two types of nodes, the *employee nodes* and the *email nodes*. An employee node is connected to an email node if the employee is either

Metrics		Graph				
		Facebook	*Enron*	*Citation*	*Coauthor*	*AS*
Number of Edges		817090	183831	705084	3742140	23409
Number of Vertex		63731	36692	27400	511164	11174
Degree	max	1098	1383	2468	597	2389
	min	1	1	1	1	
	avg	25.64	10.02	25.70	7.32	4.19
	median	10	3	15	4	2
% of nodes in transitive triads		77%	67%	40%	87%	41%
$size(S_v)$	max	20189	18770	35995	9661	6027
	min	1	1	1	1	1
	avg	190.47	69.46	187.60	30.16	9.53

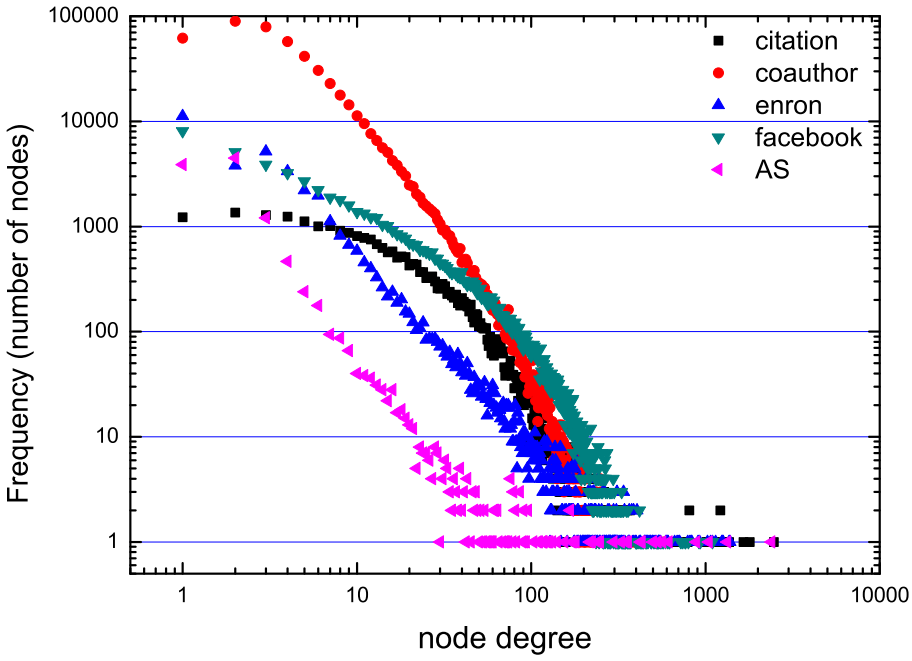
Table 3.5: Statistics of the five graphs.

a sender or a recipient of that email. The edges and nodes in the Enron graph are transformed in the following way: there is an edge between two employee nodes if they are connected by a common email node, that is, if they communicate via email. Similarly, the Coauthor and AS graphs are regarded as Bipartite graphs in the sense that Coauthor graph is derived from a Bipartite graph consisting of paper nodes and author nodes; the AS graph is transformed from a Bipartite graph with router nodes and traffic flow nodes. Facebook and Citation graphs are *Unipartite* graphs, since these two graphs only have a single type of node, the Facebook users, and the papers, respectively.

Graph Statistics

The statistics about the number of vertices/edges and the node degree are listed in Table 3.5. The Facebook graph [147] has 807090 edges connecting 63731 vertices with an average degree of 25.64. The Enron email graph involves 36692 users and 183831 edges with an average node degree of 10.02. The Coauthor graph includes 511164 authors connected by 3742140 links with average degree of 7.32, while the Citation graph has 27400 papers connected by 705084 links with the average node degree equals to 25.70. AS includes 11174 nodes and 23409 edges, and the average node degree is 4.19.

Figure 3.4 is a log-log scale plot of the node degree distribution for the five datasets. The x axis represents the node degree, and the y axis is the number of nodes having that degree. The Bipartite graphs, Coauthor, Enron and AS, exhibit characteristic power law distribution.



The log-log plot of the node degree distributions of the five graphs. The x axis represents the degree, and the y axis represents the number of nodes with the same vertex degree. The bipartite graphs, Coauthor, Enron and AS, exhibit characteristic power law distribution.

Figure 3.4: The log-log plot of node degree distributions.

3.4.2 Social Properties

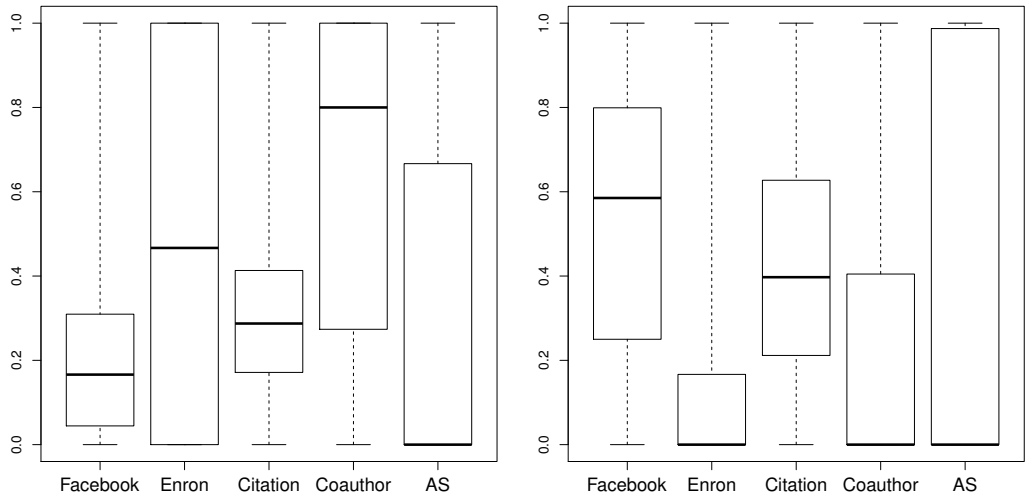
In this section, we will discuss the social properties and measurements utilized in the distributed cache selection methods.

Social Properties for Elimination Algorithms

Table 3.5 displays the percentage of nodes that are involved in at least one transitive triad in each graph. This measurement affects the performance of the *Triad Elimination* method. The Coauthor graph has the highest value, and we believe the reasons are twofold. First, research papers are more likely to be composed by authors from the same lab or institute, and the same group of authors tends to collaborate to produce papers. Second, the Coauthor graph is crawled

from the DBLP conferences; authors from this area tend to submit papers to the same conference over several years. The Facebook and Enron email graphs have higher percentages of nodes involved in transitive triads. This is because of the social network principle: "your friend's friends are more likely to be your friends". The Citation and AS graphs have smaller percentages of nodes involved in a transitive triad compared with the other three graphs.

Table 3.5 also includes statistics on the $Span$ of a node (number of edges covered by a node), as $size(S_v)$. $size(S_v)$ affects two cache selection algorithms: the Randomized algorithm, and the Span elimination algorithm. For any node v, $size(S_v) \leq deg(v) * (deg(v)+1)/2$. Therefore, it is no surprise that on average $size(S_v)$ correlates with $deg(v)$. The Citation graph has the highest value for the $max(size(S_v))$ as well as the $max(deg(v))$.



(a) Clustering Coefficient. (b) Egocentric Betweenness Centrality.

Figure 3.5: Boxplot of Clustering Coefficient and Egocentric Betweenness Centrality.

Social Properties for Social Score Algorithms

The Social Score algorithm utilizes Clustering Coefficient, Egocentric Betweenness Centrality and degree of a vertex.

The boxplot of the Clustering Coefficient for each graph is shown in Figure 3.5(a). The clustering coefficient tells how well connected the neighborhood of a node is. With lower quantile and median equal to the minimum, the clustering coefficient boxplot for the AS graph shows that at least half of the users have a clustering coefficient equal to 0, and neighbors of

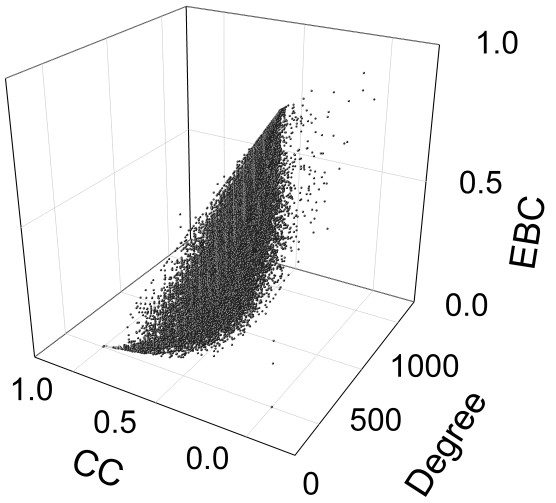
a node tend to be poorly connected. The boxplot for the Coauthor graph has the same upper quantile as the maximum with the median around 0.8. We believe this is because coauthors comprise highly connected local communities for collaboration. The boxplot of the Enron graph is evenly distributed among [0,1]. The boxplots for the two Unipartite graphs, Facebook and Citation, show similar layouts. The median for the Facebook graph is lower than for the Citation graph showing that vertices in the latter connect more closely than those in the Facebook graph.

The boxplot of Egocentric Betweenness Centrality is in Figure 3.5(b). The plots for the two Unipartite graphs, Facebook and Citation, are again similar, with the Facebook graph having a higher median. This illustrates that on average, a vertex in the Citation graph connects more non-connected vertices than does a vertex in Facebook graph. The median and lower quantile equal the minimum value in the Enron, Coauthor and AS graphs (the three Bipartite graphs). This demonstrates that at least half of the vertices are not centered in their egocentric graphs, hence, are unlikely to be selected as caches. The evaluation results in Section 3.5.3 verify this by showing that the number of social caches selected is less than half the number of vertices no matter which cache selection method is used for these three graphs.

We also calculate the *social score*, $ss(v)$, and the *social score probability*, $ss\_prob(v)$ for every node v in each graph. The social score distributions in the 3D coordinate system formed by node degree, Clustering Coefficient(CC), and Egocentric Betweenness Centrality (EBC), are presented in Figures 3.6, 3.7, 3.8, 3.9, and 3.10 with degree as the x axis, clustering coefficient as the y axis, and egocentric betweenness centrality as the z axis. The dots are the social scores, which are calculated based purely on local information.

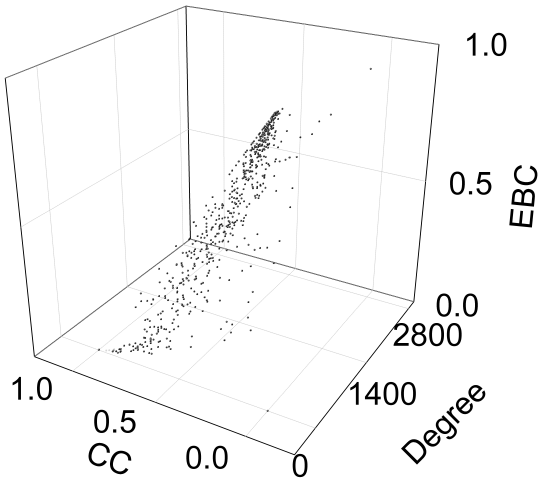
Figure 3.11 shows the boxplot of the social score probability, $ss\_prob(v)$, for each graph. The three Bipartite graphs, Enron, Coauthor, and AS, have medians and lower quantiles equal to the minimum (0). The Unipartite graphs, Facebook and Citation, have medians greater than 0.9 and lower quantiles greater than 0.7. This means that the percentage of vertices elected as social caches in the Facebook and Citation graphs (Unipartite) is greater than in the other three graphs.

Facebook and Citation, as Unipartite graphs, have similar graph and social properties in



Social score plot for Facebook graph in the 3D coordinate system composed of vertex degree, clustering coefficient, and egocentric betweenness centrality (x, y, and z axes, respectively).

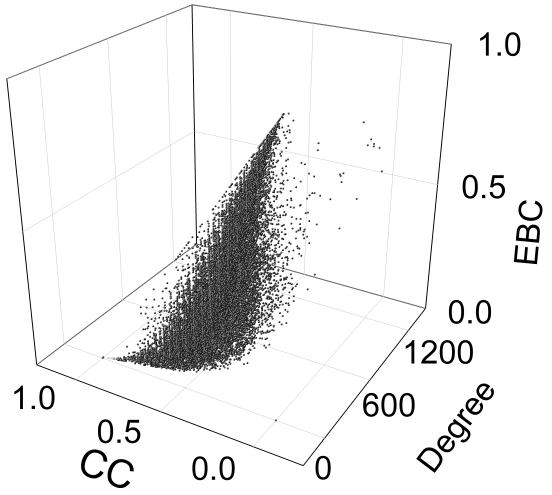
Figure 3.6: Social score plot of Facebook graph.



Social score plot for Enron graph in the 3D coordinate system composed of vertex degree, clustering coefficient, and egocentric betweenness centrality (x, y, and z axes, respectively)..

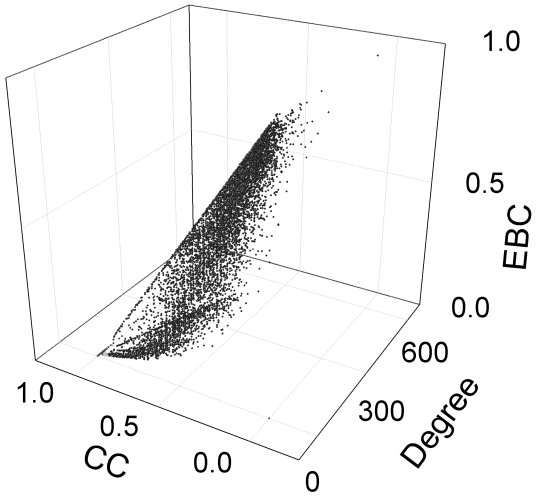
Figure 3.7: Social score plot of Enron graph.

terms of average node degree, percentage of nodes in transitive triads, average number of edges covered ($size(S_v)$), clustering coefficient and egocentric betweenness centrality distributions, as well as social score probability. Since we utilized these social properties in the cache selection methods, we believe that these similarities will translate into similar performance of the corresponding methods.



Social score plot for Citation graph in the 3D coordinate system composed of vertex degree, clustering coefficient, and egocentric betweenness centrality (x, y, and z axes, respectively).

Figure 3.8: Social score plot of Citation graph.

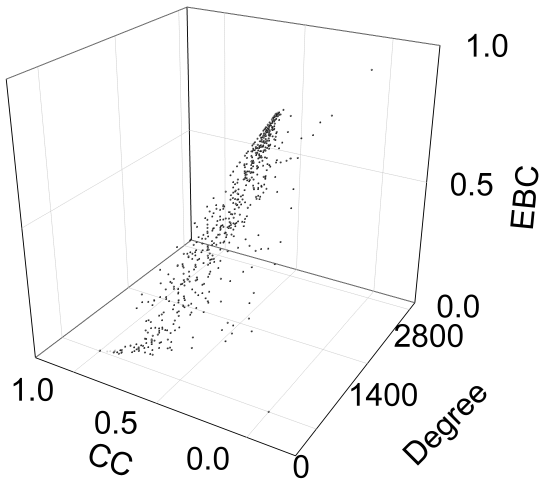


Social score plot for Coauthor graph in the 3D coordinate system composed of vertex degree, clustering coefficient, and egocentric betweenness centrality (x, y, and z axes, respectively).

Figure 3.9: Social score plot for Coauthor graph.

3.5 Evaluation

We evaluate four distributed social cache selection algorithms using the five graphs that range from “Social Graph” to “Non-Social Graph”, and “Unipartite Graph” to “Bipartite Graph” as discussed in Section 3.4 in order to answer the following questions:



Social score plot for AS graph in the 3D coordinate system composed of vertex degree, clustering coefficient, and egocentric betweenness centrality (x, y, and z axes, respectively).

Figure 3.10: Social score plot of AS graph.

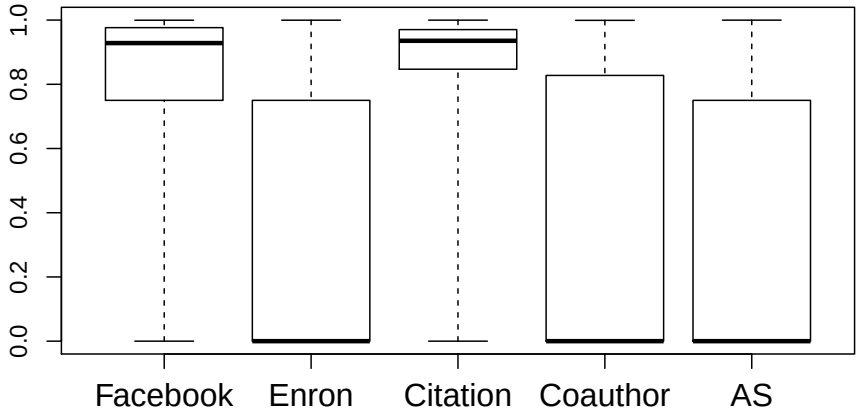


Figure 3.11: Boxplot of the social score probability for each graph.

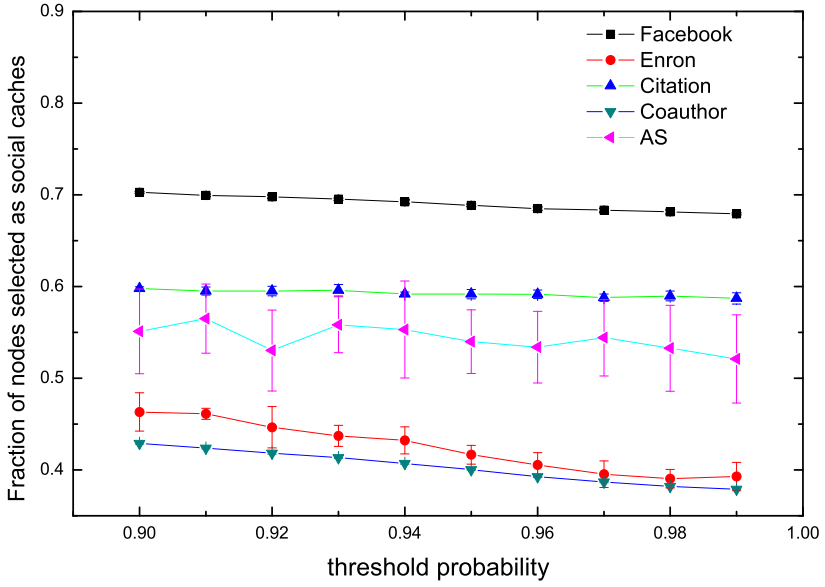
- Which cache selection algorithm performs best in terms of number of caches selected?
- Do the social properties discussed affect the cache selection algorithm performance? If so, how ?
- Do particular graph categories, e.g. social/semi-social/non-social graph, or bipartite/unipartite graph, affect the selection of caches?

We will first present some results regarding the Randomized and Social Score algorithm, and then compare the four algorithms.

3.5.1 Randomized Algorithm

To evaluate the Randomized algorithm, we vary the probability threshold, θ, and run the algorithm 10 times for any given θ on each graph. As we anticipated, the algorithm converges slowly since it is based on randomly generated numbers.

From this section, we will compare the performance of different algorithms given the five graphs. Since each graph has different $|E|$ and $|V|$, we use *fraction* of nodes (edges) for comparison purpose. Figure 3.12 displays the fraction of nodes elected as social caches with error bars (y axis) when varying θ (x axis) for the five graphs. The minimum value of θ we test is 0.90, since the fraction of nodes elected shows a clear pattern of stability for every graph. Specifically, as θ decreases, the fraction of nodes elected remains almost the same for the Facebook and Citation graphs, but increases slightly for the Enron and Coauthor graphs. As for the AS graph, the results zigzag as θ decreases, and we believe it is due to the following two reasons: *(i)* the Randomized method is based simply on randomly generated numbers, which makes it unpredictable; *(ii)* the topology of the AS graph differs from the other four graphs as it is a "Non-Social" graph.



Fraction of nodes elected as social caches (with error bars) by randomized algorithm when varying the θ. The x axis represents θ increasing, and the y axis represents fraction of nodes elected.

Figure 3.12: Fraction of nodes elected as social caches.

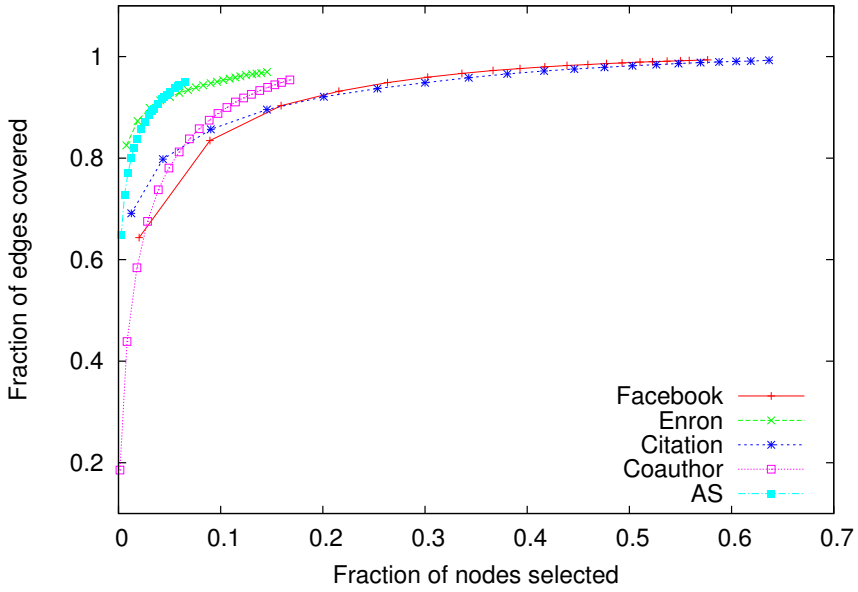
3.5.2 Social Score Algorithm

The performance of the Social Score algorithm is determined by two key parameters: the social score probability threshold ρ and the ratio threshold γ. Therefore, we perform a set of experiments to answer the following questions:

(i) What are the fraction of nodes elected and the fraction of edges marked as *green* after stage 1 (Algorithm 4) when varying the ρ?

(ii) What is the number of social caches elected by the algorithm when varying both ρ and γ?

(iii) How does the algorithm eventually converge?



Fraction of nodes elected (x axis) v.s. fraction of edges marked as green (y axis) after the first phase of the algorithm when varying the ρ. For each line, the left most point represents ρ = 0.995, and the following points represent ρ decreases by 0.05 to 0.9.

Figure 3.13: Fraction of nodes elected v.s. fraction of edges marked as green.

First, we observe stage 1 of the algorithm, where nodes elect themselves based on the social score probability. Figure 3.13 plots the fraction of edges marked as *green* (y axis) depending on the fraction of nodes elected as social caches (x axis). For each line (corresponding to one of the five graphs) the leftmost symbol represents ρ = 0.995, the following symbols represent ρ decreased by 0.05 and the rightmost symbol corresponds to ρ = 0.9. Thus, we increase the

fraction of nodes selected (x-axis) by decreasing the threshold probability. As we decrease ρ, the fraction of social caches elected and the fraction of edges marked as *green* both increases. As we discussed in Figure 3.11, more than 50% of vertices in both the Facebook and Citation graphs have a social score probability greater than 0.9. Therefore, more than 60% of nodes are elected as social caches for these two graphs when ρ reaches 0.9. For the other three graphs, less than 20% of the nodes are elected after the first phase of the algorithm. In each graph, the fraction of edges that are covered after the first phase approaches 100%.

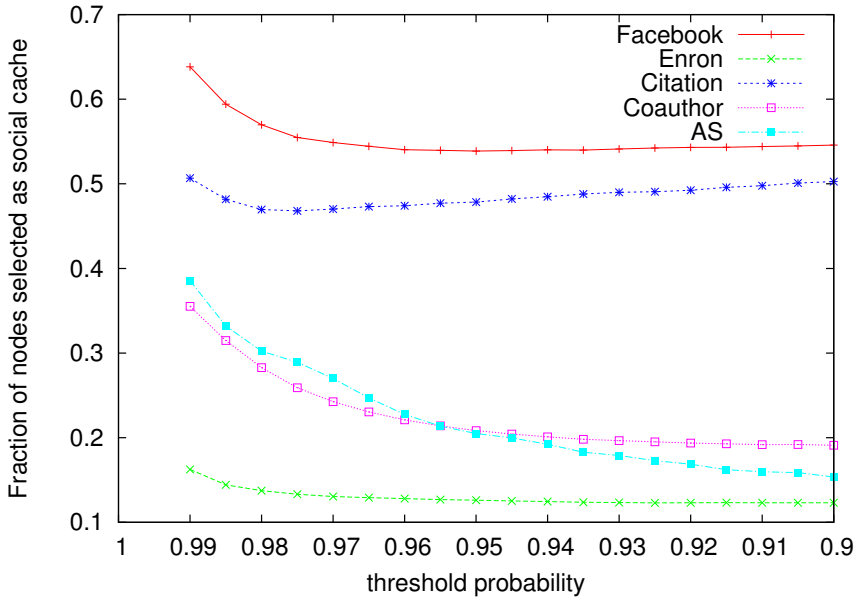
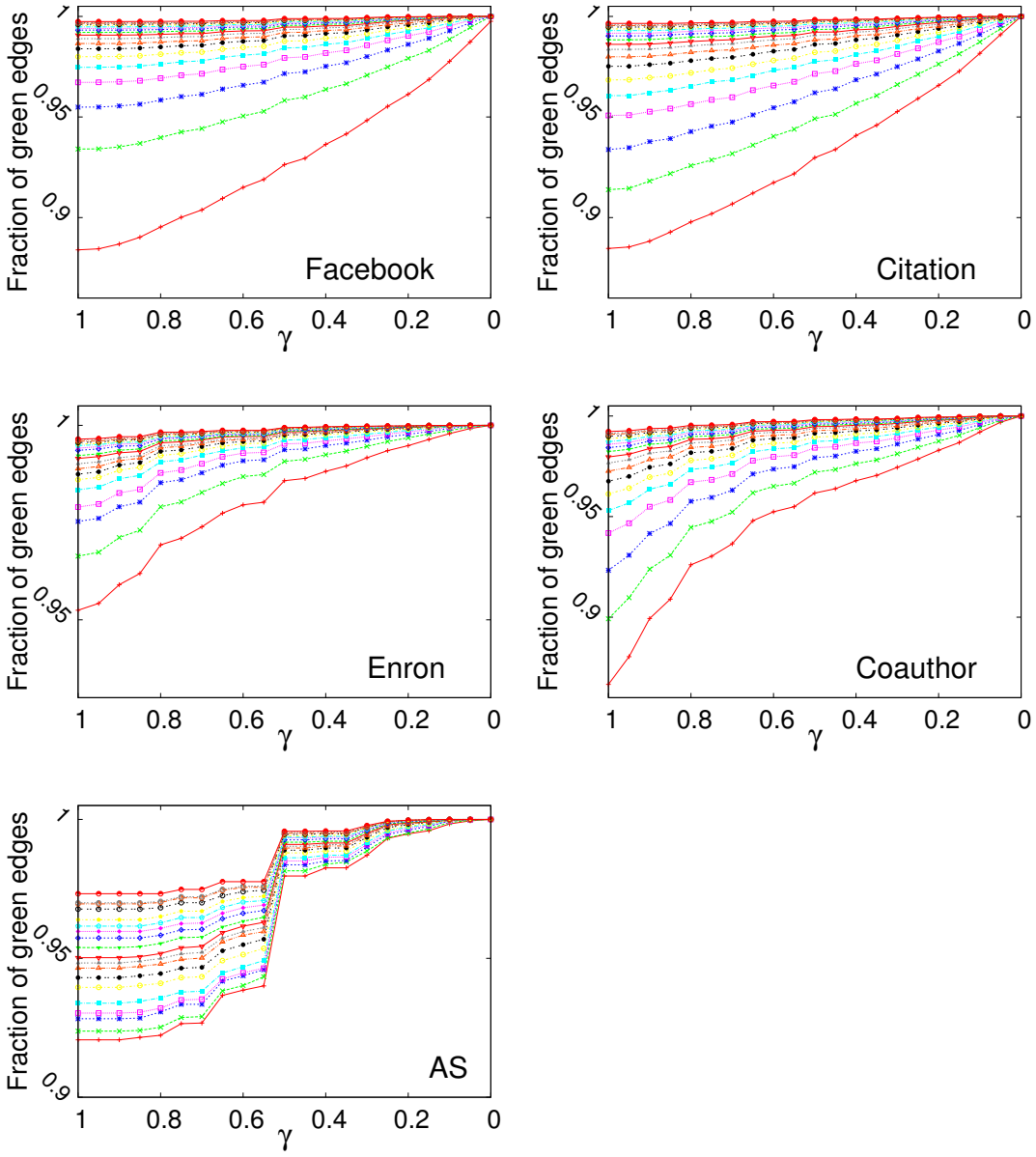


Figure 3.14: Fraction of vertices elected as social caches when varying ρ.

Second, we study the number of caches elected when the algorithm finishes. The results are shown in Figure 3.14. The x axis represents ρ ranging from 0.99 to 0.9, and the y axis is the fraction of nodes elected as social caches. In this plot, γ is first set to 0.95 and $RATIO\_STEPSIZE$ is set to 0.05. The fraction of nodes elected decreases as ρ decreases. This is because decreasing ρ enables stage 1 of the algorithm to elect more social caches, hence, covering a larger portion of the edges in the graph. For the Unipartite graphs, Facebook and Citation, the fraction of nodes elected as social caches reaches a minimum (optimal value) at ρ = 0.97. This is due to the fact that, in the two Unipartite graphs, approximately 30% of the nodes have a social score probability between 0.97 and 0.99, as shown in Figure 3.11. Reaching an optimal value is observed for the Enron graph as well. This can be explained by the fact



Convergence of the Social Score algorithm in the average fraction of green edges when γ decreases (x axis). The different lines in each subgraph, from bottom to top, show ρ increasing from 0.9 to 0.99.

Figure 3.15: Convergence of the Social Score algorithm on five graphs.

that, for the particular probability threshold optimal number, more edges become green during the first phase. For the other two graphs, Coauthor and AS, the number of caches decreases as the probability decreases.

Finally, we discuss how the algorithm converges when varying γ for different values of ρ.

We use the fraction of *green* edges in the graph as the measurement to evaluate the convergence rate. The plots are in Figure 3.15, where the x axis represents γ, and the y axis represents the fraction of *green* edges in the graph. The different colored lines in each subgraph show, from bottom to top, γ varying from 0.9 to 0.99, with a stepsize of 0.05. From Figure 3.15, it can be concluded that the convergence rates for all five graphs are very similar. In the AS subgraph, there is a spike the γ decreases from 0.6 to 0.4. We believe that this is because the number of nodes with degree equal to 1 is non-negligible; these nodes will be elected as social caches only when the ratio threshold decreases to 0.5. Indeed, a large portion, 3866 out 11174 vertices in the AS graph, have degree equal to 1.

3.5.3 Comparison of the Four Algorithms

We evaluate the performance of the four proposed algorithms further by comparing the percentage of nodes selected as social caches. Specifically, we compare them with the *Approximating NDS algorithm* introduced in Section 2.5.2, which is a centralized cache selection method that has an $O(\log m)$ approximation to the cache selection (Neighbor-Dominating Set) problem, where m is number of edges in the graph.

Table 3.6 lists the fraction of nodes selected as social caches when different methods are applied to each graph. (The number of caches selected is also listed in parentheses for comparison purposes.) Because the cache selected by the Randomized method does not change much as a function of θ, we choose $\theta = 0.99$ for the comparison. The results for the Triad and Span elimination methods are an average of ten runs to reduce possible bias. The standard deviation in the experiment is also listed. We choose $\rho = 0.95$ for the Social Score method because it is a reasonable value for all five graphs, as is clear from Figure 3.14. The fraction of nodes selected as social caches for each graph is also plotted in Figure 3.16. From Table 3.6 and Figure 3.16, we observe that the two *Unipartite graphs*, Facebook and Citation, select 20% more than the other three graphs no matter which cache selection method is used, which confirms our initial guess in Section 3.4.2. We believe that the social properties discussed in Section 3.4.2 are the key influencers for social cache selection and placement.

From Table 3.6, it is clear that the *Span Elimination* method selects the least number of

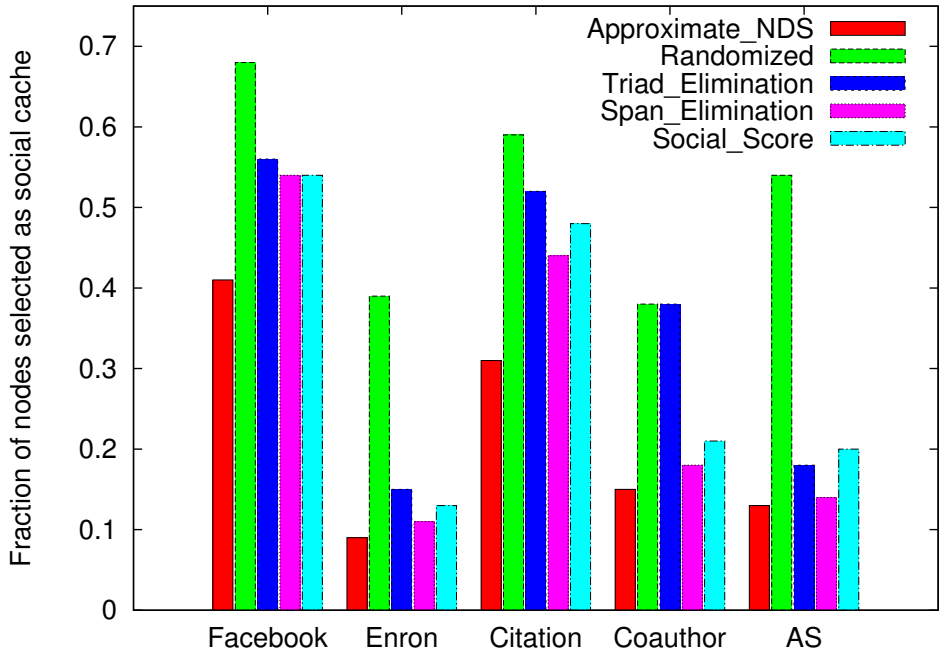


Figure 3.16: The fraction of nodes selected as social caches.

caches outperforming the other three cache selection methods, and approaches the Approximation NDS algorithm for the given graphs. Since the caches selected by the Approximation NDS algorithm can dramatically reduce network connections, caches selected by our proposed method can therefore reduce network connections in great magnitude as well.

Figure 3.16 shows that while all three algorithms, Span Elimination, Triad Elimination, and Social Score perform similarly well, all three are about 15% lower than the centralized best known approximation algorithm, Approximate NDS algorithm. The Span Elimination algorithm performs best in all cases, although occasionally (Facebook data), the performance of Social Score is comparable. We believe this is because the Social Score algorithm is specifically designed for *Social Graphs*, e.g. Facebook. It works better for a *Social Graph* than for graphs in the other categories. The Span elimination algorithm outperforms the Triad elimination, mainly because it takes into account the span of a node, in addition to the number of triads around the node.

3.6 Cache Maintenance

As OSN friends join and leave over time, the underlying social graph changes as well, requiring a re-election of social caches. In this section, we discuss the *Least Cache Re-election* (LCR) mechanism, which maintains the proper set of caches while reducing the re-election overhead as much as possible. The mechanism is inspired by the Least Cluster Change approach [26] proposed for Mobile Ad Hoc networks.

When a node enters or leaves a social network or when new connections are created or existing ones removed, a procedure called *friending* or *de-friending* takes place. With respect to friending, three scenarios are of interest for our purpose: a cache node friends a non-cache, a non-cache friends a non-cache, and a cache friends a cache. In order to reduce the maintenance overhead, the LCR does not perform re-elections when a non-cache node friends a cache or when a cache friends another cache. Although these actions may result in redundant caches, the existing set of caches will still be a Neighbor-Dominating set.

When a non-cache node friends another non-cache node, a cache reelection occurs to ensure that a cache node is available for this newly formed connection. In this case, the two non-cache nodes u and v will exchange their friend lists and calculate their common neighbor set $CN(u, v)$. If there exists at least one cache in $CN(u, v)$, no re-election is needed. Otherwise, a new cache node needs to be elected between u and v. Depending on which cache selection method is used, they compare the corresponding measurements, (e.g.: exchange $T(v), T(u)$ if the Triad Elimination method is in use) and select a cache according to the algorithm. This procedure is further explained in Algorithm 6.

Algorithm 6: Least Cache Re-election for Friending

```
when node u and v are friending
if u is a non-cache and v is a non-cache then
    exchange the social measurement
    select the one with higher value
else
    do nothing
end if
```

Similarly, we consider the following scenarios when de-friending occurs: a cache node de-friends a cache, a cache de-friends a non-cache, and a non-cache de-friends a non-cache. In

the situation where a cache de-friends a cache, or a non-cache de-friends another non-cache, no re-election is needed. In the scenario where a cache de-friends a non-cache, the re-election is needed to ensure that every friend of the non-cache can receive its social updates. We adopt a simple approach by letting the non-cache node notify its neighbors about the de-friending in order to initiate a cache re-election. Note that, in case of a cache miss, a node can always go to the original node to get the latest social updates. We believe the "Least Cache Re-election" mechanism is a good trade-off between cache availability and maintenance overhead.

3.7 Discussion

In this section, we discuss several aspects that we did not cover in this chapter but can be further explored.

Network Dynamics and Availability. The performance of SocialCDN is directly influenced by network dynamics and content availability. In Section 3.6, we described possible directions for handling nodes' joins and leaves as well as unexpected failure situations. Detailed experimental evaluation with realistic nodes' reliability data should be examined.

Load Balancing. How to balance the network traffic associated with the cache nodes is another important issue. Formulating a new optimization problem by adding an additional constraint related to the load (in terms of number of connections or traffic per cache node) is another direction that can be explored.

Privacy. SocialCDN assumes that users know their immediate friends and friends-of-friends. In this dissertation, we have not investigated scenarios of nodes misbehaving or Sybil attacks, which are other directions of future work.

3.8 Summary

In this chapter, we presented SocialCDN, a novel social content dissemination system for Distributed Online Social Networks based on social caches. By caching social updates on social caches, SocialCDN enables the efficient data dissemination among social buddies through fewer network connections. We proposed four fully distributed cache selection methods for SocialCDN based on different social properties: the Randomized, Triad Elimination, Span

Elimination, and Social Score algorithms. Empirical evaluations on five well known graphs show that the Span Elimination algorithm has the least time complexity in terms of communication steps, and selects the least number of social caches for any given graph. We also found that the overall performance of the cache selection algorithms depends on social properties and measurements.

Algorithms	**Graph**				
	Facebook	*Enron*	*Citation*	*Coauthor*	*AS*
Number of Vertex	63731	36692	27400	511164	11174
Centralized Apprx NDS	0.41(26288)	0.09(3370)	0.31(8517)	0.15(79672)	0.13(1505)
Randomized (p = 0.99)	0.68(43291)	0.39(14416)	0.59(16061)	0.38(193399)	0.54(6079)
Triads Elimination	0.56(35913.6)/(29.42)	0.15(5410.7)/(36.19)	0.52(14355)/(33.56)	0.38(194527.9)/(32.85)	0.18(1998.3)/(12.88)
Span Elimination	0.54(34096.7)/(6.88)	0.11(3929.1)/(7.62)	0.44(12116.0)/(6.55)	0.18(92089.3)/(21.87)	0.14(1585.1)/(3.31)
Social Score ($p = 0.95$)	0.54(34331)	0.13(4627)	0.48(13107)	0.21(106530)	0.21(2290)

Table 3.6: Fraction of nodes selected as social caches.

Chapter 4

Multiparty Voice Communication Over Vehicular Social Networks

4.1 Introduction

A great number of people spend one or more hours driving each day. These daily roadway commutes are highly predictable and regular, and provide a great opportunity to form virtual mobile communities. However, even though these commuters are already physically present in the same location, they are limited in their ability to communicate with each other.

In this chapter, we propose a framework for building such communities, which we call Vehicular Social Networks (VSNs), to facilitate better communication between commuters driving on roadways. Vehicular Social Networks enable motorists to exchange real-time messages with others in their proximity, so that the motorists can get to know the road conditions in real time. Hence, drivers can determine the optimized routes to their destinations based on the real-time traffic information, reduce the time and gas consumed waiting in the traffic, and make their road trip more enjoyable. VSNs can be used as platforms for various vehicular and traffic related applications. In this dissertation, we will present one of the applications, called RoadSpeak.

Just as people taking mass transit often pass the time socializing with those around them, motorists could benefit from social interactions if they were given broader social opportunities. While people are driving, the practice of texting or emailing is forbidden. Therefore, a method of using voice messages is by far the most reasonable for the Vehicular Social Networks. Systems that enable users to communicate through voice messages in real-time are called *Multiparty Voice Communication* systems, for example, conference calls or a CB radio. Unfortunately, existing Multiparty Voice Communication (MVC) systems cannot be scaled to large numbers of users and do not provide adequate access controls. For example, Skype only

allows 25 participants to be involved in a voice chat group, and the host has to have sufficient network bandwidth as well as CPU power, and memory to do so.

RoadSpeak is a scalable MVC system over Vehicular Social Network that allows motorists to automatically join voice chat groups along popular roadways, and socialize with others in real-time. RoadSpeak achieves scalable and interruption free communication through the use of voice message buffering, automated moderation and flow control. We have implemented a RoadSpeak prototype on Nokia N95 smart phones with 3G cellular networking for voice message transfer. We have also built an MVC simulator to perform large-scale simulations that compare RoadSpeak with existing MVC systems. The results of our evaluation prove the feasibility of RoadSpeak and demonstrate that it performs comparably to traditional MVC systems while supporting substantially larger groups of users.

Section 4.2 presents the related work. In Section 4.3, we will introduce Vehicular Social Networks, present the VSN model and discuss how to form the VSN groups. In Section 4.4, we will present the design and implementation of RoadSpeak. We will also discuss how the MVC system with automated moderation in RoadSpeak differs from other Multiparty Voice Communication systems, and how they can provide large scale real-time voice communication. Section 4.5 presents the automated moderation and voice buffering at both client and server sides, and how they are used to provide collision free communication in RoadSpeak. In Section 4.6, we will present the evaluation for RoadSpeak as well as Multiparty Voice Communication System. Specifically, Section 4.6.1 presents the benchmark and field trial evaluation of RoadSpeak. Section 4.6.2 describes the multiparty voice communication simulator we built, and the simulation results by comparing the RoadSpeak system with other MVC systems. We will discuss the open questions in Section 4.7, and conclude the chapter in Section 4.8.

4.2 Related Work

Traditional online social networks such as Facebook [48] and LinkedIn [90] are essentially a virtual view of physical communities, however they are agnostic to location and proximity. Recently, there has been an increasing trend towards mobile social networks that can help people

connect with their physical communities and surroundings. SocialNet [138] infers shared interests between people by storing the IDs of the devices nearby and analyzing this co-location data collected over a long period of time. Social Serendipity [46] is another similar mobile phone-based application using Bluetooth and a database of user profiles to recommend face to face interactions between nearby users who share common preferences. MobiSoc [14] is a middleware that enables the formation of mobile social networks and provides a platform for managing the social state of physical communities.

Vehicular networks have become increasingly popular recently due to the interest from the government as well as from auto manufacturers. Various types of applications that make use of vehicular networks have been proposed, ranging from improving safety [108], traffic management [156], emergency management [122], urban sensing, multimedia sharing [73], and gaming [115] to disseminating advertisements to passengers [87]. All existing work in the field of vehicular networks considers links between vehicles to be spontaneously formed as a result of the vehicles being in wireless range of each other.

Vehicular networks are a special case of mobile ad-hoc networks (MANETs) and delay tolerant networks (DTNs). In the field of MANETs and DTNs, various papers have proposed applying knowledge about social networks in order to improve networking protocol performance. In [33], the authors propose a routing algorithm for MANETs that, based on the small world phenomenon seen in social networks, identifies bridge nodes for routing based on their centrality characteristics. [103] shows how knowledge of nodes' social interactions can help improve the performance of mobile systems. [13] studies the impact of users' social relationships and movement patterns on the performance of routing protocols in opportunistic networks. Similarly, a social network based overlay for publish-subscribe communication in DTNs has been studied in [155]. None of the existing works, however, consider vehicular networks as the underlying network that can make use of a social network overlay.

A number of projects have described ways of spontaneously creating mobile social networks in different network settings. Bottazzi, et.al. [15] describe a middleware for managing location-dependent relations in mobile social networks and for propagating the social networks' visibility up to the application layer. The Haggle [72] project has performed extensive studies

of human mobility traces in opportunistic networks and extracted information related to clustering, and community structure.

Our work is distinct from all the earlier work in that RoadSpeak is the first application built on a vehicular network with an underlying social network. We believe that our work will inspire existing vehicular applications to leverage the underlying social connections that exist in vehicular networks and form vehicular social networks.

4.3 Vehicular Social Networks

In this section, we present the Vehicular Social Networks (VSNs) framework, to facilitate better communication between commuters driving on roadways. Specifially, in Section 4.3.1, we introduce Vehicular Social Networks in detail. In Section 4.3.2, we discuss the VSN infrastructure, deployment model, and privacy issues. Section 4.3.3 presents the Vehicular Social Network group definition, and how to form a Vehicular Social Network group.

4.3.1 Introduction to Vehicular Social Network

Social Networks allow people with common interests to come together to form virtual communities. Mobile Social Networks connect people who are already together (sharing the same locality), enabling them to take advantage of their close proximity to form more tightly-coupled, possibly ad-hoc, virtual communities. In this socially-driven virtual world, people can form instant villages that mirror and facilitate real-world interaction.

Today, services such as Path [116] utilize mobile phones to exchange information between users regarding their location in order to identify opportunities to meet. Friends can find each other when they happen to be close by, and people can find others they have never met before who share their interests as well as their location. Mobile social networking services are successful primarily because they take direct advantage of the often periodic physical locality patterns of the users to improve their knowledge about opportunities to socialize.

An unexplored source of physical locality, which can be taken advantage of for mobile social networking, is the daily roadway commute. A great number of people, especially in densely populated ares, spend one or more hours each day driving between home and office.

Since these commutes typically take people over highways and other well frequented corridors, they are highly predicable and regular. Day after day, the same people travel along the same roadways at the same time. Therefore, the opportunity exists to form periodic virtual mobile communities. We call these virtual communities *Vehicular Social Networks* (VSN).

The key property for building Vehicular Social Networks is that roadways, and the Vehicular Social Network overlaid thereon, provide a sufficient and regular concentration of people who may wish to socialize. People partake in all forms of entertainment that are available, while commuting to and from work and other destinations. They listen to music, listen to and participate in radio talk shows, chat with friends over hands-free phones, etc. In fact, in analogous commuting situations, such as on buses or trains, it is common for people to socialize and communicate with others around them. We believe that a substantial opportunity exists for people to organize into social groups and expand their social connectedness while traveling on roadways, for the purpose of entertainment, to pass the time during their commute, to inform each other about traffic conditions ahead, or even to ask for help in case of an emergency.

We envision three major reasons for Vehicular Social Network formation: *(i)* for entertainment, *(ii)* for utility, and *(iii)* for emergency purposes. Entertainment based Vehicular Social Networks are formed for people to share common interests, for example, to discuss recent social and political issues, sports, or any other interesting or entertaining topic. Entertainment based Vehicular Social Networks primarily serve to occupy people during their long and boring commutes, similar to the way radio programs provide entertainment. Utility based Vehicular Social Networks are formed to facilitate non-essential, yet helpful connections. For example, local towns or states might establish Vehicular Social Networks to discuss interesting local events or points of interest along popular highways, which travelers might use to query for advice in selecting a local restaurant or hotel. There might also be Vehicular Social Networks established to coordinate carpooling or communicate roadway conditions (accidents, congestion, etc). Finally, Vehicular Social Networks can become particularly useful in case of an emergency. Such emergency-based Vehicular Social Networks might serve as a way for people to ask for and provide assistance to each other in critical situations. Enterprises can set up Vehicular Social Networks for employees as they travel to and from work. In case of an emergency, such as a

flat tire or car accident, an employee can contact others who are close by for help.

4.3.2 Vehicular Social Network Model

A Vehicular Social Network is a social network that is enabled by: *(i)* spatial locality, and *(ii)* temporal locality. Roadways naturally corral drivers into very regular and predictable patterns. Aside from very exceptional cases, vehicles are constrained to travel on a roadway, which is natural spatial locality enforced by the fact that drivers must use.

There is also temporal locality in the way many drivers utilize roadways. For most commuters, this temporal locality is brought about by various recurring daily deadlines. Commuters must arrive at work on time, and are usually required to remain at work until some later fixed time. Commuters base their daily routine on temporal locality.

Although spatial and temporal locality are enablers of VSNs, they are not the motivation for their formation. As with any social network, it is the social aspects that define the utility of the network, and VSNs share the same goals as any other social network. They exist to allow people to form new connections, to expand their social connectedness, and to support various social applications, based mostly upon their common interests.

Vehicular Social Network Deployment

In order to realize a VSN, we must first consider the following basic issues related to VSN formulation and deployment. These issues can be grouped into three categories: *(i)* state maintenance, *(ii)* VSN centrality, and *(iii)* communication.

Any state related to VSN specifications and user profiles must be maintained in order to support user-defined criteria for VSN creation. This state can be stored either at trusted central servers or locally with users. Additionally, a state can be made persistent, to support stable VSNs, or it can be considered transient, in the case of ad-hoc VSN formation.

VSNs may be formed with differing centrality. A location-centric VSN is one that is formed from the perspective of some fixed location. Membership in location-centric VSNs is restricted to those drivers matching the predefined social criteria and passing through predefined locations with some recurrence frequency. An example of a location-centric VSN would be the network

of drivers crossing the George Washington Bridge. At the other end of the spectrum, are VSNs that are formulated from the driver's perspective. A driver-centric VSN fixes each driver at its center, regardless of location, and connects others to that driver as he encounters them on the roadways. In a driver-centric VSN, the strength of connections between drivers is determined by their level of commonality in social interests as well as by the frequency of their recurring encounters.

Finally, we must choose a model to support inter-vehicular communication. There are a number of models in existence today. Among them, are car-to-car approaches including Vehicular Ad-Hoc Networks, centralized communication infrastructure approaches such as cellular communication, and various hybrid approaches.

Infrastructure Requirements

There is a broad range of infrastructures that can be assumed to support Vehicular Social Networks. First, we assume that vehicles are equipped with some form of wireless communication. This may include wireless-enabled smart phones (e.g., 3G/4G data access, WiFi, WiMAX, etc.), or vehicles may be equipped with embedded vehicular computing systems that support car-to-car communication (C2C). External to the vehicles, we can assume either a fully deployed road-side infrastructure, which supports car-to-infrastructure (C2I) communication, no infrastructure at all, or some deployment model in between, such as a CarTel-like [73] in-situ organic WiFi network approach. The infrastructure assumptions we make will have a substantial impact on the design of any VSN-based systems and applications.

Location Validation

Although location is part of the definition, it may or may not be enforced by the VSN. When it is not enforced, a user who provides false location information would still be able to participate in a VSN, contrary to the VSN specification properties. When location is enforced, the VSN must use some method to validate the location information provided This could be either *(i)* direct validation, for example, using a road-side infrastructure to periodically sample user positions; or *(ii)* cross-validation performed between users.

Privacy

Since a VSN may use driver-specific time-location information, there is a privacy issue beyond that traditional social networks must handle. The privacy requirements for VSNs will differ based on their specification properties. A VSN model that includes cross-validation will directly expose user location information to users around them. Therefore, it may be important to include mechanisms to anonymize the data with respect to the individual users, while only exposing location information to a central authority. Alternatively, a model that does not include cross-validation might only require users to expose location information to a centralized, trusted, authority or to an anonymizer. In either case, this data must be handled as highly sensitive to its users and the system must not directly leak this information to other users or allow indirect inferences based on the information that is exposed.

4.3.3 Vehicular Social Network Group

As we discussed, the primary goal of a Vehicular Social Network is to facilitate communication between people who are already physically present in the same location, but who are limited in their ability to communicate with each other. In most of the existing social scenarios, such as a cafe, nightclub, etc., people can easily communicate with each other since they are physically close by. They are not restricted by their activity as they are when driving on roadways. Today, if people wish to speak with each other while occupying a vehicle, they must either be collocated in the same vehicle, join a radio talk show, or initiate a direct connection to each other (e.g., make a cell phone call). Vehicular Social Networks take advantage of the common situations drivers on the same road segments share, to allow opportunities for them to socialize and communicate in an easy and natural manner.

Vehicular Social Networks may provide a simple asynchronous communication protocol for drivers to participate in discussions in a given *Vehicular Social Network Group*. Users can join and participate in the Vehicular Social Network group that matches their interests. As part of this dissertation, we developed RoadSpeak, which enables drivers to form and join Vehicular Social Network groups for socialization.

Vehicular Social Network Group Creation

A vehicular Social Network group is defined by a profile consisting of three factors: *(i)* the time interval, *(ii)* the location or road segment, *(iii)* and the group's interests.

For example, a user might create a VSN group for the time slot starting at 4 PM and ending at 6 PM. Similarly, a user could also define a VSN group for a specific section of a roadway, and this VSN group would include any registered user who is currently located on that section of the roadway. Location properties could be specified using either coordinates (i.e., GPS values) or some logical description of the location (e.g., miles 501-576 on Route 1). These temporal and spatial factors in the VSN group definition may limit membership to specified roads and time intervals. Finally, VSNs can be specified using group's interests on which traditional social networks are formed. These are user-defined keywords that provide a meaningful definition of the purpose or collective interests of the users included in the VSN group.

The VSN group profile is defined by the group owner (which can be the manager, an organization, or simply a user) when the group is created. The privilege of group ownership can be transferred to any of the group members by the current group owner.

Users are admitted to groups based on matching their user profiles to the group's profile upon registration. Admission can be free or it can be restricted by certain requirements that the user must prove, such as a location frequency threshold for a specified roadway segment during a specified time slot. By limiting group membership based on the frequency with which the users travel the roadway, a VSN group may be limited to commuters, the best target for this type of system, rather than including transient travelers.

4.4 RoadSpeak

In this section, we will discuss RoadSpeak, the first Vehicular Social Network application that allows groups of motorists to socialize and communicate with each other by means of voice messages in a real-time fashion, utilizing automated moderation and flow control. Users of RoadSpeak can automatically join Voice Communication Groups formed along popular roadways. As an application of VSN, RoadSpeak groups also enforce time and location as part of the profile definition.

We have implemented a prototype of the RoadSpeak system, which will be discussed in Section 4.4.5. RoadSpeak client runs on a user's smart phone, and geo-matches her location with the group profile in which she is interested. The user is then automatically logged on to one of her groups of interest, and can participate in that group's discussion via real-time voice messages.

Communication via voice messages is referred to as a Real-Time Multiparty Voice Communication (MVC) system. Multiparty Voice Communication has been investigated extensively over the past 40 years, most recently under Massively Multi-player Online Game (MMOG) scenarios. In a typical MVC system, speech samples are collected from an audio device, compressed, broken into packets, and transmitted to the receiver. At the receiver, the speech samples are retrieved, decompressed, reassembled, and sent to the audio device for playback.

Today, a number of MVCs exist as commercially and freely available products. Push to Talk over Cellular (PTT PoC) [120] is a service option for cellular phone networks that permits subscribers to use their phones as walkie-talkies. DT-Talkie [100] enables PTT PoC communications over infrastructure-less and latency-challenged environments by employing a DTN [45] architecture. SimPhony [86] presents a mobile voice communication system built on a PDA that supports one-to-one or one-to-many communication with voice messages. TeamSpeak [1] and Ventrilo [146] are popular Internet based MVC systems frequently used under MMOG scenarios.

RoadSpeak's Multiparty Voice Communication system differs from the traditional MVC system in the following ways: in a traditional MVC system, the number of active participants in a group is effectively limited by audio collisions, occurring when participants speak simultaneously or when one participant is interrupted by another. Such collisions increase the frustration level of participants, because they slow the rate of progress in the conversation and force participants to repeat their messages at the next opportunity. To enable scaling to large audio communication groups, RoadSpeak allows individual participants to submit audio messages in an interruption-free manner through voice message buffering and flow control techniques. Multiple participants may submit messages concurrently, thereby eliminating audio collisions. These voice messages are interleaved by the RoadSpeak server and transmitted to all clients.

The server implements a flow control mechanism by stopping clients from sending more messages when the buffer is full. Both the RoadSpeak client and the server explore the voice message buffers to improve the users experience.

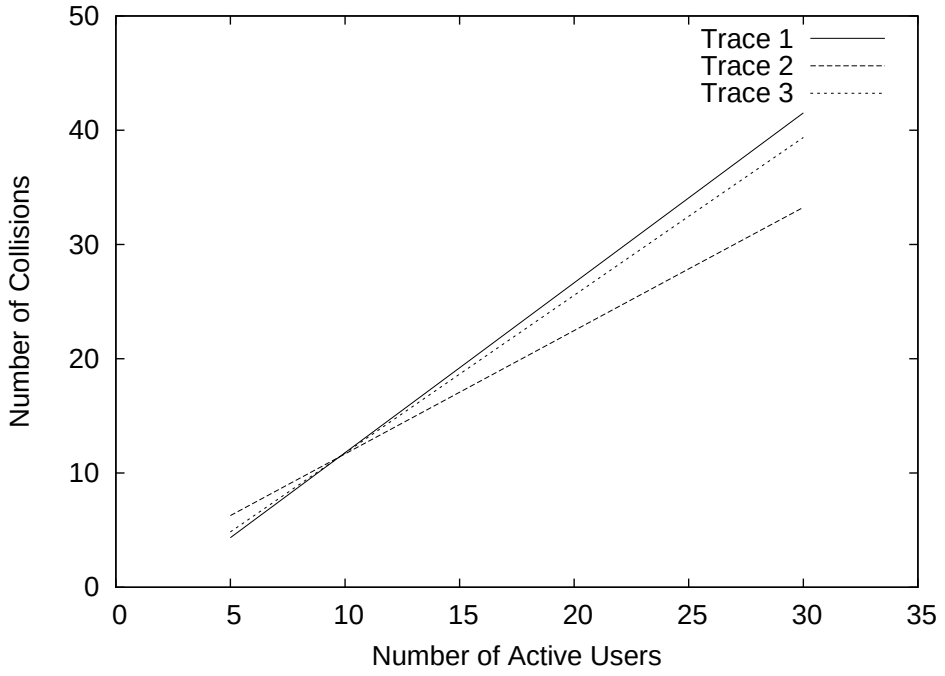
In Section 4.4.1, we discuss the problem of collision in Multiparty Voice Communication (MVC) system. Section 4.4.2 provides an example of how RoadSpeak can be used for on-road socialization by motorists. We give an overview of RoadSpeak in Section 4.4.3. The architecture and design, both server-side and client-side, are discussed in Section 4.4.4. The implementation of RoadSpeak is presented in section 4.4.5.

4.4.1 Collision in Multiparty Voice Communication

Most social settings that involve sizable groups of people are also likely to involve significant numbers of simultaneous speakers. Research has revealed significant periods when several participants were simultaneously generating audio traffic [17, 16]. In general, voice patterns in MVC scenarios consist of *talkspurts* and *silence periods*. Farber et al. [50] show that talkspurt periods follow an exponential distribution in traditional telephony, while Papp and GauthierDickey [1] demonstrate that talkspurts follow a Weibull Distribution.

To better understand the patterns of collision rates in multiparty communication scenarios, we use three sets of traces from real multiparty IRC (Internet Relay Chat) text chatting sessions [54]. Because multiparty audio communication trace data is lacking, we use text chatting as a coarse approximator for audio chatting with respect to the occurrence of collisions.

The total size of the traces we analyzed is over 2 MB, consisting of 13.5 hours and 10747 lines of text-chat sessions. For our analysis, we define a *collision* under text-chat to have occurred when multiple users communicate within the same (short-scale) time window (the *collision time window*). Using this definition for a collision, we scan the text-chat session traces sequentially to determine where collisions occur. Figure 4.1 presents the results of our analysis. For each trace, we plot the regression curve for the number of collisions as we vary the number of actively participating users (denoted as *active users*). Since there is no way to define session within the text-chat traces, we are unable to determine directly who the active users are, so we define an *active user* as a person who communicates a minimum of five times



The lines in the figure are the regression curves of the number of collisions that occur in each of the three text chat traces with a one second collision window. The regression curves illustrate the positive correlation between number of active users and text-chat collisions.

Figure 4.1: Regression curves of collisions in text chat sessions.

during a ten minute period. The collision time window used to generate the results shown in the figure is one second long.

From the figure, we observe that for each set of traces, as the number of active users scales from 5 to 30, the number of collisions increases from 5 to 40. For each trace set, the plotted regression curve shows a clear positive correlation between number of active users and number of collisions. Although this may be acceptable in a text chat scenario, such a trend would quickly become intolerable in a traditional MVC scenario.

4.4.2 RoadSpeak Functional Scenario

To better illustrate the RoadSpeak concept, we provide an example RoadSpeak scenario. This scenario is only one of a number of possible usage cases and is chosen to demonstrate how a typical user, Joe, might utilize the RoadSpeak system. Beyond this example, there are many other foreseen and un-forseen scenarios in which RoadSpeak could be useful for motorists

(e.g., during an emergency, or to communicate interesting observations, etc.). Joe, a RoadSpeak user who currently participates in a RoadSpeak audio socialization group during his afternoon commute home, would like to start participating in a new socialization group during his daily morning commute. From his home computer, he goes to the RoadSpeak web portal, logs on using his account, and browses through the various RoadSpeak groups available along the route he takes between home and office. He chooses to register a sports talk group active on his work route between 6 AM and 8 AM, since he is usually on the highway from 6:30 AM - 7:15 AM. Finally, he verifies that he has the most recent version of the RoadSpeak client installed on his smart phone.

The next morning, Joe gets into his car, sets his smart phone in its cradle, and puts on his hands-free kit. As he travels to work, the RoadSpeak client running on his smart phone tracks his location via the built-in GPS receiver, and when he enters the highway at 6:32 AM, the client contacts a RoadSpeak server and automatically adds Joe to the RoadSpeak sport group that he registered with.

Joe receives an audible alert from his phone, notifying him that he is joining the chat group. He introduces himself to the group and begins listening to the current sports talk voice chat messages that his RoadSpeak client receives from the server. As he listens, he is happy to note that he is familiar with a number of the other participants who also belong to his regular afternoon RoadSpeak chat groups.

After a while, Joe intervenes and speaks to the group. The RoadSpeak client captures and transmits his voice message to the server. As the server continues to send out messages in the sequence it receives them, it sends Joe's message to the other participants in the group. Fran and Harry both hear Joe's question to the group and respond about the same time. Under a typical multiparty voice communication system, this would lead to an audio collision. Under RoadSpeak it does not, since the messages are buffered at the client's end and sent to the server once completed. The server, after receiving voice messages, serializes the messages into a sequential order and transmits them individually to the other group participants.

Finally, as Joe leaves the highway, the RoadSpeak client running on his smart phone detects his departure and notifies him that he will leave the sports socialization group, shortly. Joe

transmits a farewell message to the group and continues along his route to work. In addition to the scenario described above, Joe could also join a RoadSpeak audio socialization group of interest while he is at his home or workplace, if the group (optionally) does not enforce location.

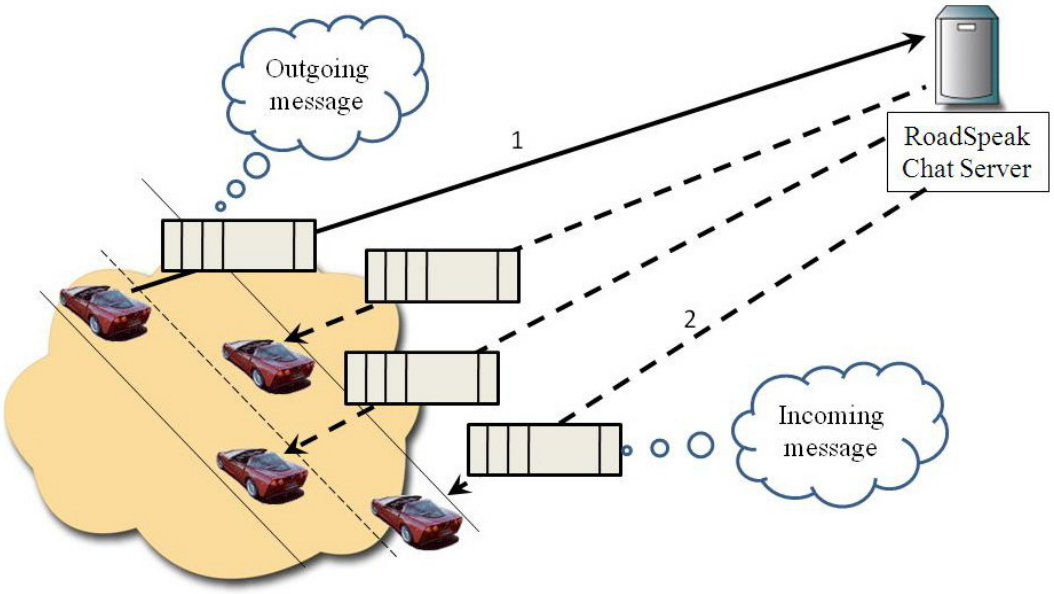
4.4.3 Overview of RoadSpeak

The overall design of RoadSpeak is guided by two goals. The first goal is to *increase the scalability of RoadSpeak real-time audio socialization by providing interruption-free communication*. The second goal is to *enable a consistent user experience by enforcing fair and equal access to the audio channels*. RoadSpeak provides interruption-free multiparty voice communication by *enabling participants to submit messages whenever they like*, and by *buffering voice chat messages*. Once submitted, these messages are sequentially ordered by the RoadSpeak server, and delivered to RoadSpeak clients in order, for playback. With RoadSpeak, audio collisions can never happen.

Figure 4.2 illustrates the paths of audio messages in RoadSpeak. RoadSpeak clients, after recording audio message from users, will put the audio messages in the "Send" buffer, which is responsible for sending audio messages to the server over the network. The RoadSpeak server buffers incoming messages and transmits them to the other audio socialization participants. Buffering messages at the server side decouples the sender from the receivers, such that once a message has been submitted by a speaker, it is guaranteed to reach any actively logged in clients, ensuring message delivery. The clients, after receiving the incoming audio messages, will further buffer them in the receive buffer before playback.

These buffers not only function as regular buffers, but also work together with automated moderation and flow control techniques to act as a key enabler for interruption-free and scalable voice communication. They also support novel social properties, such as message serialization, message pause/resume, and so forth. We will discuss these techniques in Section 4.5.

Furthermore, in order to enable a consistent user experience, RoadSpeak enforces quotas on participant speaking time. This ensures that all participants are allowed a fair share of the voice chat channel, preventing any one user from dominating a group. A Quota Manager controls



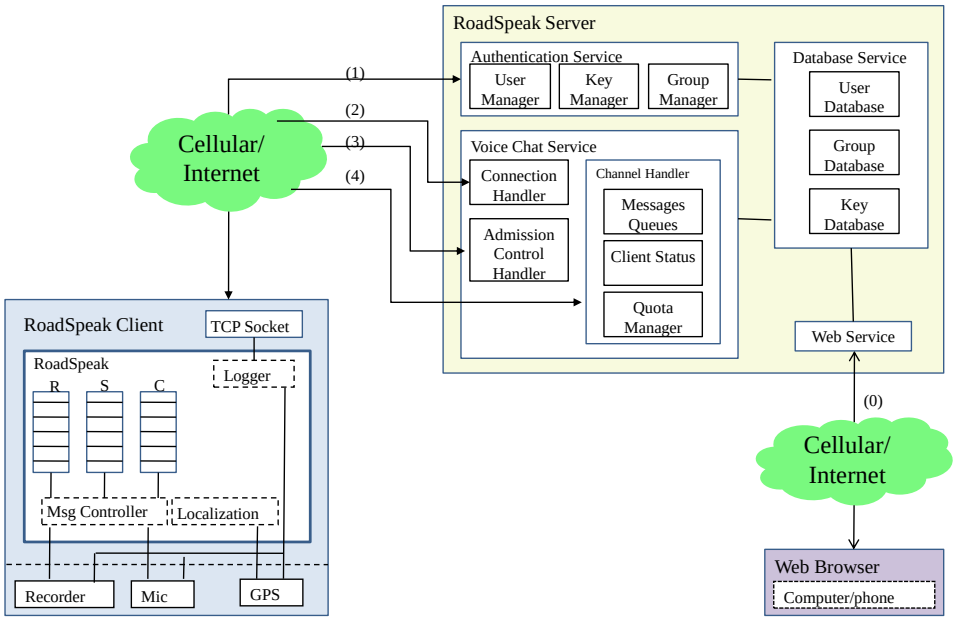
RoadSpeak clients send voice messages to the server through a wireless network (1). The server, after receiving messages from clients, transmits the messages to all clients in the voice chat group (2).

Figure 4.2: RoadSpeak Overview.

both a user's total speaking time and the individual lengths of each message sent by a user. Each user in the group is assigned a quota, and once a user reaches her quota, she can no longer speak to the group until her quota has been refreshed by the server. RoadSpeak also provides an application-level flow-control mechanism. From the perspective of the RoadSpeak server, this flow-control mechanism is used to adjust the message rates of clients. From the client perspective, flow-control allows a participant to discard recently created messages, prior to transmission.

4.4.4 RoadSpeak Architecture and Design

The RoadSpeak architecture is shown in Figure 4.3 and is composed of two main components: the RoadSpeak server and the client. The server handles authentication, access control, flow control, and message delivery. The client software, once downloaded and installed on a smart phone, handles all message capture and playback functions. Finally, there is a web portal through which a user connects in order to organize the groups they own, create new groups,



RoadSpeak clients can use their browsers to access the web server (0). An authentication service handles user authentication (1). (2), (3) and (4) handle group admission and chatting procedures.

Figure 4.3: RoadSpeak Architecture.

and join existing groups owned by others. In the following subsections, we will describe each component in more detail.

RoadSpeak Server

The RoadSpeak server includes the following components: the Authentication Service, the Audio(Voice) Chat Service, the Database Service, and the Web Service which are shown in Figure 4.3. Furthermore, the RoadSpeak server maintains three message buffers: the Send Buffer, the Receive Buffer and the Control Buffer. These buffers, together with Automated Moderation and Flow Control will be discussed in Section 4.5.

All persistent user and group data is stored in a database and accessed by both the Authentication Service and the Audio Chat service.

User Creation and Authentication. Users can browse the RoadSpeak web portal to create an account in the system ((1) in Figure 4.3) and log in. Users download a client application to

their smart phones (or embedded vehicular PCs), and in order to access groups, login through the client. Both user account creation and login authentication are handled over a secure web protocol (i.e., HTTPS) by the User Manager. At the time of an account is created, the client generates an asymmetric cryptographic key pair (using PKI) and registers its public key (K_{PUB}) with the User Manager.

Once a user has been authenticated to the system, she is free to create new groups, to invite other users to the groups she owns, or to request membership to groups owned by other users. Group invitations and join requests are forwarded to other users (potential participants and group owners, respectively) to either accept or decline. In the case where a group is owned by multiple users, the owners vote to either allow or deny new users into the group.

Group Membership. A RoadSpeak voice communication group is defined by its location, time interval (e.g.,10AM-12PM), and interests. Groups also have specific characteristics, such as public or private. Public groups are open to all users as long as they satisfy the group requirements, while private groups require the explicit permission of the group owner. Groups can also be defined as open or moderated. In an open group, anyone can freely communicate with members of the group, whereas in a moderated group, permission to communicate is explicitly granted by the group owner and can be revoked at any time. Finally, a group can be hidden or advertised. Hidden groups are only advertised to those users who are allowed to participate in the groups, while advertised groups are visible to all.

Admission to a group is handled by the Group Admission Manager. To be admitted to a group, a user submits his profile. This data allows the Group Admission Manager to verify that the user attempting to join the group, actually satisfies the group's properties. If the Group Admission Manager validates the user, and allows her to join the group, the group key (K_G) is transmitted back to the client to be used by the client in the future for that group.

Connection Handler and Admission Control. A user chooses which group he wishes to participate in, once he has logged into the RoadSpeak system (through the local client on his smart phone). A client may only enter one RoadSpeak audio communication group at a time. When a RoadSpeak client connects to a RoadSpeak server (see (2) in Figure 4.3), the Connection Handler spawns a new Admission Control Handler (ACH) and hands the connection to the

ACH to perform admission control (see (3) in Figure 4.3).

To initiate Admission Control, a RoadSpeak client transmits the group name, the user's name, his profile, and the SHA1 hash of the group key (sha1(K_G)) to a RoadSpeak server. This information is encrypted by the client, using its private key (K_{PRIV}), prior to being sent to the server. The server validates the user based on the user's information and the hash of the group key submitted by the user. Once the user drives past the boundary of the location of a group she has previously registered with, the ACH hands the client connection off to the Channel Handler for the requested group ((4) in Figure 4.3).

Channel Handler and Message Handling. The Channel Handler spawns one thread per client connection in each RoadSpeak group. These Channel Handler threads share a global data structure, called the *Server Voice Message Buffer*. Note that server voice message buffers are group-specific, meaning that each RoadSpeak group has a separate buffer; while the Channel Handlers are user-specific, meaning that the connection from each client in a RoadSpeak group is handled by one Channel Handler thread.

The server voice message buffer maintains a list of message chunk rows *(<message#, chunk#, chunk data, user name, status flags>)*. Each row represents a chunk of a voice message that has been submitted to the server by a user participating in that group chat channel. Channel Handler threads also share this buffer with the Receiver Handler, and the Send Handler associated with this group. The Receiver Handler puts each newly received message chunk from clients in a RoadSpeak group into the voice message buffer. Message chunks in this list are sent out to the participants in the same RoadSpeak group, one message at a time in the order that was defined collaboratively by both the client and the server by the Send Handler.

Each Channel Handler thread also maintains a local data structure, called the Client State List (as a part of the client message buffer) that maintains a set of client state rows (*<message#, chunk#>*). There is one row per message being sent to the client. Each row tracks the last message chunk sent to and acknowledged by the client.

The Channel Handler thread can also send *pause* and *resume* messages to their associated clients. These higher priority messages are placed on the clients' Control Message Buffer when received and are handled by the receive thread. When a *pause* is received from the server, the

client receive thread notifies the client send thread, to *pause* all chunk streaming from the client to the server. Upon receiving a *resume*, the client receive thread notifies the client send thread to begin chunk streaming, once again.

User Event Notification. There are a number of conditions that must be reported to the user as feedback during a voice communication session. For each of these, the client provides audible and visual feedback to the user.

RoadSpeak Smartphone Client

The RoadSpeak client was developed to be downloaded to smart phones, carried by users. The client software, after a successful log in, periodically reads a user's GPS location and automatically joins the user into an appropriate group, based upon the set of groups she has selected (from the web interface), her present location, and the current time. To do so, RoadSpeak performs geo-matching, which is further discussed in Section 4.7.

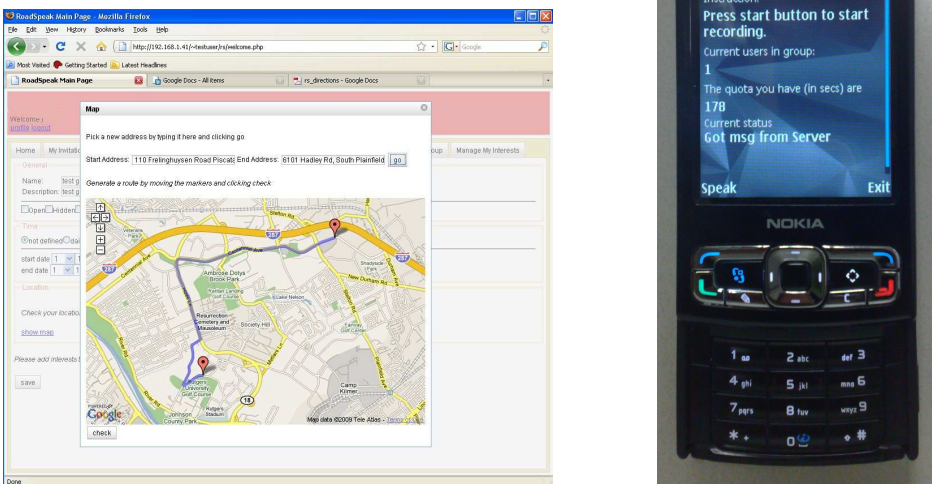
The RoadSpeak client maintains three message buffers (these will be discussed in detail in Section 4.5), as shown in Figure 4.3: *(i)* the send buffer, *(ii)* the receive buffer, and *(iii)* the control buffer. These buffers are used to queue and control the outgoing voice messages, incoming voice messages, and flow control messages, respectively.

As in other real-time multiparty voice communication systems, audio data is captured by the client through the smart phone microphone. Unlike other MVC systems, RoadSpeak buffers the audio messages into the client send buffer to be sent reliably to the RoadSpeak server.

Any new audio messages received by a client are placed on the receive buffer. The audio message playback thread selects messages from the receive buffer for playback to the user based upon how the individual user interacts with the receive buffer.

The RoadSpeak server can also send high priority flow control messages to clients. These higher priority messages are placed in the client-side control buffer. A pause directs a client to cease all message sending to the server and a resume notifies the client to begin sending once again.

Client-side send, receive, and control buffers will be discussed in section 4.5 as well as the server-side buffers. These voice and control message buffers play a key role in improving the



(a) Web interface screenshot of a user defining a new VCG.

(b) Client software executing on a Nokia N95 smart phone.

Figure 4.4: RoadSpeak server and client screenshot.

RoadSpeak architecture as well as in providing a unique and customized user experience.

4.4.5 RoadSpeak Implementation

The RoadSpeak server is developed in Java and the client is a J2ME Midlet for Nokia N95 smart phones. The client can use either 3G/4G or WiFi connectivity to access the server. The web portal provides group and user maintenance, and management functionality, and is developed using PHP and Javascript for the Apache2 web server. RoadSpeak groups associated with a certain region are returned to a visualization service that displays the regions using the Google maps API with the JMaki toolkit. Screenshots of the web portal and the smart phone client are presented in Figure 4.4(a) and Figure 4.4(b), respectively.

4.5 Flow Control and Automated Moderation

RoadSpeak uses voice message buffering, automated moderation and flow control at both the server and the client side to enable scalable and collision-free multiparty voice communication. The flow control of RoadSpeak together with automated moderation enables the following

two properties: *(i)* the individual's *Personal Behaviors*, and *(ii)* the *Social Behaviors* with other users. Personal behaviors include all behaviors that do not involve other users, and identify how a user manipulates a voice message, either an outgoing message or an incoming message, such as a pause/resume/skip audio message. Social behaviors include enabling control messages, emergency messages, etc. Flow control and voice message buffering also enforce rules such as controlling audio message length, controlling total speaking time, and assigning a speaking quota to each user in the RoadSpeak groups, etc. In this section, we will discuss the flow control together with voice message buffers used by the RoadSpeak system from both the server and client sides.

4.5.1 Client Flow Control and Voice Message Buffer

As discussed in Section 4.4.4, each RoadSpeak client maintains a set of three buffers: *(i)* the send voice message buffer; *(ii)* the receive voice message buffer, and *(iii)* the control message buffer. These buffers are used to buffer and control both the outgoing and the incoming voice messages, as well as the control messages that are received from the server.

Client Send Control and Send Buffer

As in any other multiparty voice communication systems, RoadSpeak voice messages are captured by the client's audio message capture thread (e.g., using the smart phone microphone). Captured voice messages are then broken into fixed-sized message chunks, and placed into the client send buffer before being sent out to the server.

The Send Handler module at the client side of the RoadSpeak prototype allows users the ability to control the voice messages in the send buffer in two ways. First, it allows a sender to *cancel* a newly captured message prior to transmission if the sender realizes that she/he no longer wants to send out a particular message. Second, it enables the user to *pause* the captured messages being transmitted to the server and *resume* them later. In case the user does not want to *cancel*, or *pause* the voice message, the client's Send Handler thread streams message chunks to the server, and awaits the server's acknowledgement.

Client Receive Handling and Receive Buffer

The client's Receive Handler module is in charge of handling the received message chunks. After receiving the voice message chunks from the server, the receive thread in the Receive Handler module places them into the client temporary buffer, and sends acknowledgements back to the server. The client's Audio Playback thread in the Receive Handler module is notified by the receive thread whenever all chunks for the lowest numbered message have been received and the message is complete. The voice message playback thread then removes the chunks, one at a time, from the temporary buffer and places the voice messages into the receive buffer, so that the voice messages can be streamed to the user in order. (e.g., using the smart phone's speaker).

While the voice messages are in the receive buffer, the RoadSpeak protocol Receive Handler enables the clients to customize them in two ways. Users of the RoadSpeak system can *discard/skip* any received messages whenever they find the messages less interesting, or when the buffer is full. They can also *pause/resume* the playback of any voice message in the receive buffer. For example, if they get a phone call while listening to the voice messages, they can *pause* the playback of the voice message, answer the call, and *resume* listening when they finish making the phone call.

In this way, RoadSpeak client-side voice message buffers provide a great deal of freedom to the individual users. The send and receive voice message buffers can be pre-defined to control the total number of outgoing voice messages, and the overall number of incoming voice messages. The server can also send control messages to define these parameters based on the number of participants in the group.

4.5.2 Server Flow Control and Voice Message Buffer

The RoadSpeak server, as discussed in Section 4.4.4 maintains a *Server Message Buffer* for each RoadSpeak group. The server message buffer is used to buffer and control the incoming and outgoing voice messages, and is shared by all Channel Handler threads, the Send Handler threads and the Receive Handler threads of the same group. These handlers and the voice message buffering work collaboratively to provide a collision-free and scalable RoadSpeak

socialization channel/group.

Once the server Receive Handler receives a message chunk from any participant in the group, it places the message chunk on the voice message buffer. While the messages are in the buffer, RoadSpeak allows social operations and social regulations on these messages. Furthermore, a Client State List for each client keeps track of whether and which message chunk in the voice message buffer has been sent to the corresponding client. The Send Handler thread, after checking the Client State List, will access the buffer and stream chunks to its associated client.

Sending Order.

First Come First Send. RoadSpeak enables the Server Send Handler to send the messages in the order that the server receives them. This is a primitive method of sending out voice messages. In this situation, the flow control with buffering works with no differently from a regular buffer.

High Priority Goes First. As we discussed, RoadSpeak is a system that enable drivers in the same vicinity to socialize with each other, but it can also be used as a platform for emergency messages. These messages are considered to be high priority, and should be specially treated so as to be sent to the clients as soon as possible.

4.5.3 Client-Server Flow Control and Client Control Message Buffer

The server Channel Handler threads can also send *pause* and *resume* control messages to their associated clients. These higher priority messages are placed on the clients' control message buffer when received and are handled by the Control Handler thread. When a *pause* is received from the server, the client Control Handler thread notifies the client Send Handler, to pause chunk streaming from the client to the server. Upon receiving a *resume*, the client Control Handler thread notifies the client Send Handler to begin chunk streaming once again.

4.6 Evaluation

In this Section, we will discuss the evaluation for both RoadSpeak application as well as Multiparty Voice Communication. Specifically, we utilize microbenchmark experiments and on road field trial to evaluate RoadSpeak application. We design and implement a Multiparty Voice Communication simulator to compare MVC system used by RoadSpeak with other MCV systems.

4.6.1 RoadSpeak Evaluation

The goal of the evaluation is to study the performance of RoadSpeak, addressing the following three questions:

- What is the base performance of RoadSpeak under a typical client workload with different network connectivity?
- How does the performance of RoadSpeak scale as more clients are added to the system, under a more realistic workload?
- Eventually, what is the usability of the RoadSpeak system, with human interactions?

In order to answer the first two questions, we perform two types of benchmark experiments to evaluate RoadSpeak performance: single client micro benchmarks and multiple client emulation, which will be discussed in section 4.6.1.

In order to answer the last question, we perform a real field trial with four volunteers to evaluate the usability of RoadSpeak, and to study the human interaction within the RoadSpeak system, which will be discussed in section 4.6.1.

RoadSpeak Benchmark Evaluation

For our benchmark and emulation platform, we used an Intel Dual Core 1.86GHz Linux PC with 1 GB of RAM.

RoadSpeak Single Client Microbenchmarks

The single client microbenchmark experiments measure the performance of RoadSpeak running over a 3G cellular network (Verizon 1xEVDO), as well as a WiFi network (802.11g). We define the end-to-end message transmission latency as the time taken to transmit a message from one RoadSpeak client to another RoadSpeak client through the RoadSpeak server. We measure the latency to transmit a 1-second message, and a 5-second message. Each result reported is the mean of 10 measurements. The average message transmission latency over the WiFi and 3G network connections is reported in Table 4.1.

	Message Length	
Connection Type	1 Second	5 Seconds
WiFi	0.03(0.00)	0.12(0.01)
3G	1.24(0.01)	2.95(0.34)

Message transmission latency in seconds over WiFi and 3G for differing message lengths. Standard deviations are reported in parentheses.

Table 4.1: Message Transmission Latency (Microbenchmark).

RoadSpeak Multi-Client Emulation

For the multiple client emulation experiments, we use a set of scripted RoadSpeak clients with a synthetically generated workload (around 5 messages per minutes) in order to evaluate how the server and client queues scale. We ported RoadSpeak to PCs for this set of experiments, and emulate 3G and WiFi network connections by injecting a synthetic delay before the transmission and reception of every message.

In the first experiment, we fix the number of speaking clients at 10. From Figure 4.5, we observe that the effect of message length on the overall message transmission latency is linear. Apart from *short*, *normal*, and *long* messages, we also consider a *mixed* workload scenario where each message is chosen to be short, normal, or long with equal probability. Since long messages dominate the transmission time, the mixed workload scenario shows results similar to the long message scenario. Message transmission delay can be further broken down into four components:

(i) server queue delay;

(ii) client send queue delay;

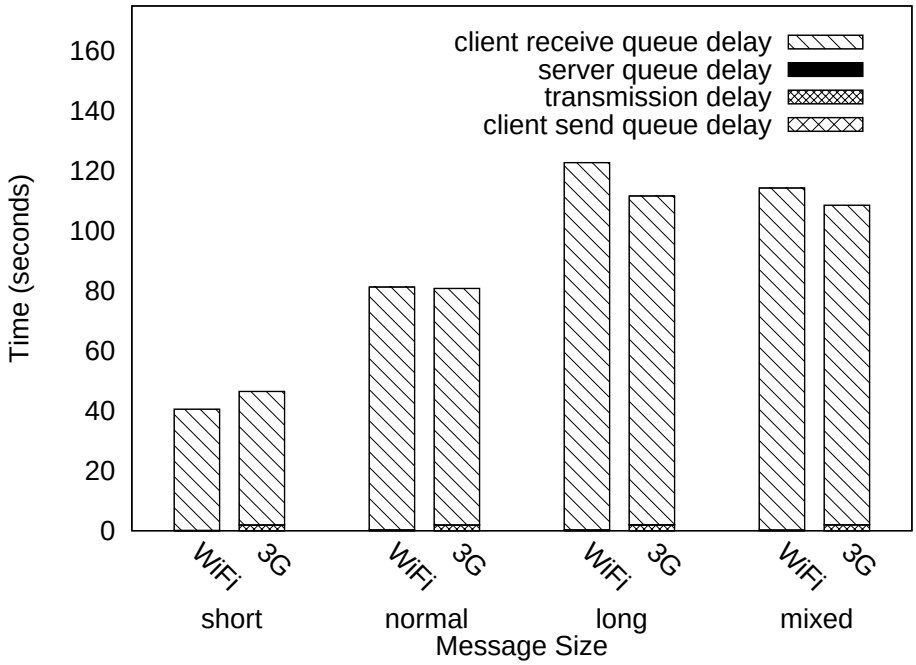
(iii) client receive queue delay;

(iv) propagation delay;

To evaluate the impact that increasing the number of speaking clients has on message transmission latency, we fix the message length at the worst case value, i.e., long messages, and vary the number of clients from 2-30 speakers. From Figure 4.6, we observe that message latency initially grows linearly until the number of users is increased to 10, after which it grows sub-linearly. Since the rate of message playback for a client is fixed, once the workload approaches this capacity, system performance stabilizes.

Finally, from both Figures 4.5 and 4.6, we observe that the overall message transmission time is dominated by the client receive buffer time. This is the length of time that messages are retained in the receiver buffer before playback/discard/skip. These messages are forwarded by the RoadSpeak server to the receivers. RoadSpeak performs reliable message delivery, so the more messages a client receives, the longer the client buffer waiting time. As we discussed in Section 4.5, the RoadSpeak prototype allows clients the ability to control the messages in the client-side voice message buffer in two ways. First, it allows a sender to *cancel* a message in the client-side send buffer, if the sender realizes that she no longer wants to send out a particular message. Second, a receiver may *skip* a message in the client-side receive buffer, for example, the message is uninteresting or the receiver's buffer is full. The results in Figures 4.5 and 4.6 represent the worst case system performance when clients never *cancel* or *skip* messages.

The purpose of these emulation experiments is not to realistically simulate either the RoadSpeak scenario or the multiparty audio conversation scenario. Rather, the purpose is to better understand how the system subcomponents directly affect the performance of the system under a synthetic load. From the results of these experiments, the client receive queue delay appears, at first, to be very large when compared to what a user would experience in a typical conference call. In fact, though, when comparing RoadSpeak queue times and conference call waiting times, we also have to consider that a conference call does not allow users to submit messages concurrently. Instead, a user will "buffer" her messages in her head, while waiting for her turn to speak. RoadSpeak, on the other hand, allows a user to speak her messages as they occur to



Message transmission latency (in seconds) over 3G and WiFi as message length is varied. Group size is fixed to ten.

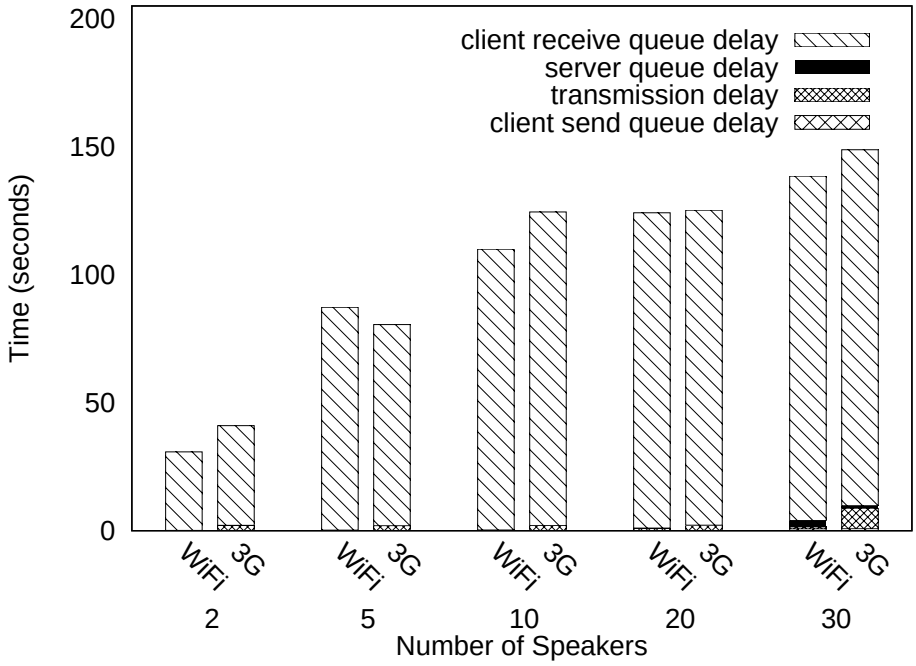
Figure 4.5: Effect of Message Length on Transmission Latency.

her, and edit them while they are still in the send buffer.

In Section 4.6.2, we show a comparison between conference calls and RoadSpeak in terms of the time it takes to have a message heard by all other parties on the call, which we believe to be the most relevant metric to understand the actual user experience.

RoadSpeak Field Trial Evaluation

We perform a field trial with real users to better understand the system performance and human interactions in RoadSpeak. Our field trial consists of four participants (colleagues in our lab), each of whom was equipped with a Nokia N95 smart phone with 3G Internet access. The participants were briefly trained on how to use the RoadSpeak system. The experiment was performed when the whole group went to a restaurant. While driving to the destination, the participants used RoadSpeak to communicate with each other.



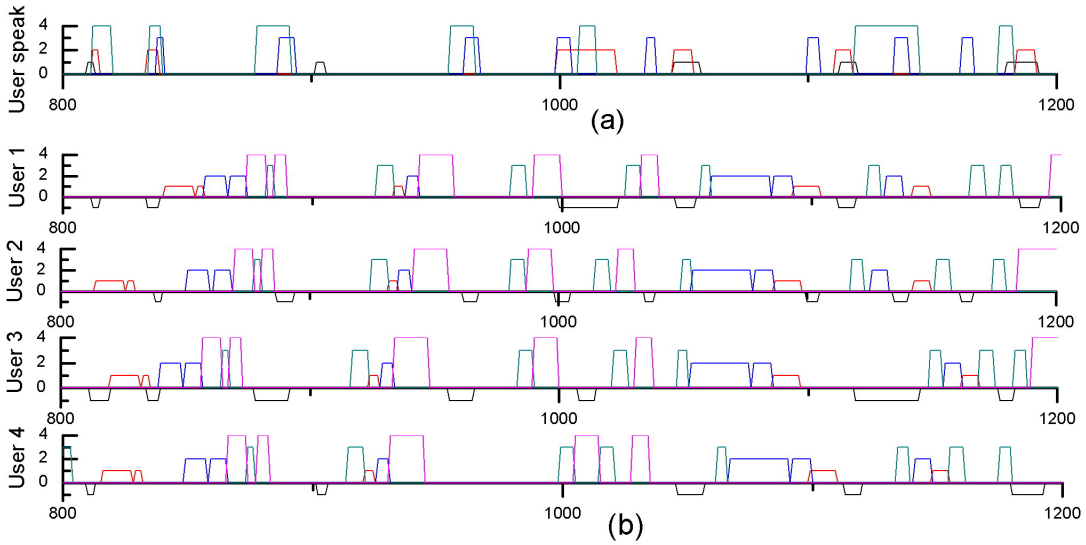
Message transmission latency (in seconds) over 3G and WiFi as group size is varied. Message length is set to *long* messages.

Figure 4.6: Effect of Group Size on Transmission Latency.

Experiment Dataset Characteristics

The entire experimental trace consists of 72 voice messages over a duration of 1200 seconds. The combined speaking time (silence periods removed) is 514 seconds, and the average number of messages per user is 18. The maximum message length is 27 seconds, the average message length is 7 seconds, and most messages are between 4 and 7 seconds long.

To better understand human conversation patterns in the RoadSpeak system, we plot the speaking and playback message durations for the field study trace. Figure 4.7 presents the time series plot for the trace period of 800 to 1200 seconds. We choose this period since it is the most active period of the trace. For each individual graph, the y-axis represents the user identifier number, e.g. a peak at $y = 1$ denotes that user 1 is speaking. When $y = 0$, all users are silent. Finally, the elapsed session time is shown along the x-axis. Figure 4.7(a) presents the speaking time for all users, while each of the four subplots of Figure 4.7(b) presents the speaking time for an individual user ($y = -1$) and the message playback times ($y > 0$) for all users.



Visualization of a four-user RoadSpeak conversation. The x-axis is elapsed time in second. In (a), a line with $y = n$ represents the time when User n is speaking. In (b), a line with $y = n$ represents message playback time from speaker n, and $y = -1$ represents when the respective user speaks.

Figure 4.7: RoadSpeak Conversation Visualization.

From Figure 4.7(a), we observe that audio collisions occur whenever multiple lines overlap. Although this is only a four-user experiment, we observe that there are numerous audio collisions. As we will demonstrate through simulation in Section 4.6.2, when the number of users increases beyond four, audio collisions occur much more frequently.

From Figure 4.7(b), we observe the beneficial effects of the collision-free communication provided by RoadSpeak. For the $y > 0$ section of each subplot, we see that there are no occurrences of overlapping lines. We also observe, by comparing the $y > 0$ and $y < 0$ sections of each subplot, that users are free to submit new voice messages, without causing interruptions to the flow of the voice-chat session.

4.6.2 Multiparty Voice Communication Evaluation

The performance of RoadSpeak depends on various factors, including the number of users, the behavior characteristics of users, policies of the buffers from both server-side and client-side, human-driven flow control and system initiated flow control. To better understand the impact of each of these factors, we build a model of human conversation to feed into a multiparty voice

communication simulator. We use this simulator to compare the performance of RoadSpeak with traditional MVC systems, which do not have any kind or system flow control. We use a conference call as one of the representatives for the traditional MVC systems. Note that we do not simulate network conditions since we want to compare the base-case (ideal) performance of RoadSpeak with other multiparty voice communication systems.

Multiparty Voice Communication Simulator

There are two approaches to building a multiparty voice communication simulator, a micro-simulator that takes into account each participant's speaking pattern and interactions among the participants, or a macro-simulator that uses aggregate statistics about the conversations. We build a micro-simulator because we are interested in the effects of interruption-free communication on the experience of individual users.

	Message Length	
Simulation Scenario	Conference Call	RoadSpeak
Talkspurt	$\lambda = 2.30$, $k = 1.18$	$\lambda = 7.6$, $k = 2.6$
Silence	$\lambda = 13.53$, $k = 0.60$	$\lambda = 58.0$, $k = 1.7$

Table 4.2: Weibull distribution parameters for MVC simulator.

Simulation Parameters

The simulator depends on the following simulation parameters.

- *The talkspurt period.* (ON period, or message length) This parameter represents the length of talkspurt, and follows a Weibull distribution. A talkspurt period follows and is followed by a silence period.

- *The silence period.* (OFF period) This parameter represents the length of each silence period and also follows a Weibull distribution. A silence period similarly follows and is followed by a talkspurt period.

- *Aggressiveness.* This is an adaptive parameter that defines the willingness of a speaker to intervene in the current conversation. Each speaker is given an initial aggressiveness probability generated as a uniformly random value between 0 and 1. If a speaker loses a

collision round, her aggressiveness is increased as a function of her current aggressiveness. Once a speaker completes her current message, her aggressiveness is reset to its original value.

Talkspurt and Silence Period Modeling

Prior research on multiparty voice communication systems suggests that both talkspurt and silence period follow a Weibull distribution [1]. We would like to study whether the field trial experiment also follows this trend. We analyzed the logs from our field trial (described in Section 4.6.1) and confirmed that the talkspurt and silence periods fit Weibull distributions. We use the parameters from Papp and GauthierDickey [1] for conference calls, and compute the best-fit parameters empirically from our field trial for RoadSpeak. Table 4.2 lists the best-fit parameters we derived for talkspurt and silence periods, as well as the parameters we use for conference calls. The model shows some interesting trends in human conversation patterns while driving. For example, 90% of talkspurts are shorter than 11 seconds, whereas 90% of silence periods are shorter than 117 seconds. Silence periods are much longer than talkspurts because during driving, people need to concentrate on the road ahead, watch traffic lights, etc.

The simulation begins by defining *number_of_speakers* in the conversation. Each speaker is assigned an *aggressiveness* value. The length of talkspurt and silence are generated from the Weibull distribution with the parameters discussed in Table 4.2.

Simulation Scenarios

The socialization simulation is used to simulate the following scenarios: RoadSpeak, RoadSpeak w/o Flow Control, and Conference Call. For simplicity, we use Joe and Tom to represent two speakers.

In a conference call scenario, when Joe intends to speak, he first checks the channel to see if another user is speaking. If Tom is speaking, Joe checks if he is more aggressive than Tom. If so, he stops Tom and starts speaking, otherwise he goes back to waiting until he has the next chance to speak. It is unlikely that someone will interrupt a speaker when he has just started to speak. We therefore define the parameter *speaking_constraint_time*, which allows the current speaker to continue speaking for *speaking_constraint_time* seconds before he can be interrupted by other speakers.

Metrics

Since we are studying human interactions and socialization during a conversation, new metrics are needed to quantify different conversational and social models. Our goal is to study the trend of waiting time and delay in each scenario especially at scale. Following, are the metrics we use to evaluate the Conference Call scenario:

- *TimeToSpeak:* the time that a user waits until he has a chance to speak.
- *TimeToFinish:* the time that a user needs to finish his statement, possibly after being interrupted several times.
- *Conference Call Delay:* the time from when a user starts to speak until the completed message reaches the other participants, possibly after multiple interruptions.

The following metrics are used to evaluate the RoadSpeak scenario:

- *TimeToSpeak:* the time a speaker waits, due to RoadSpeak flow control, before he can speak.
- *Buffering Delay:* message delay due to buffering at both the client and the server side.

In each scenario, we vary the number of speakers from 2 to 30. For each test, we mark half of the speakers as aggressive speakers and the other half as passive speakers. We chose 30 as the maximum group size since the conversation patterns are revealed at and beyond 20 speakers per group. In the Conference Call scenario, more than 80% of the voice messages were interrupted by others when the number of speakers in the group was greater than 10, which makes it very difficult for a speaker to complete a message.

Simulation Results

From Figure 4.8 we observe that, for all three scenarios, as the number of speakers increases, message completion time also increases. The time for both RoadSpeak and Conference Call increases linearly, while RoadSpeak w/o Flow Control increases dramatically after the number of speakers in the group exceeds 15. RoadSpeak performs better than Conference Call in all cases.

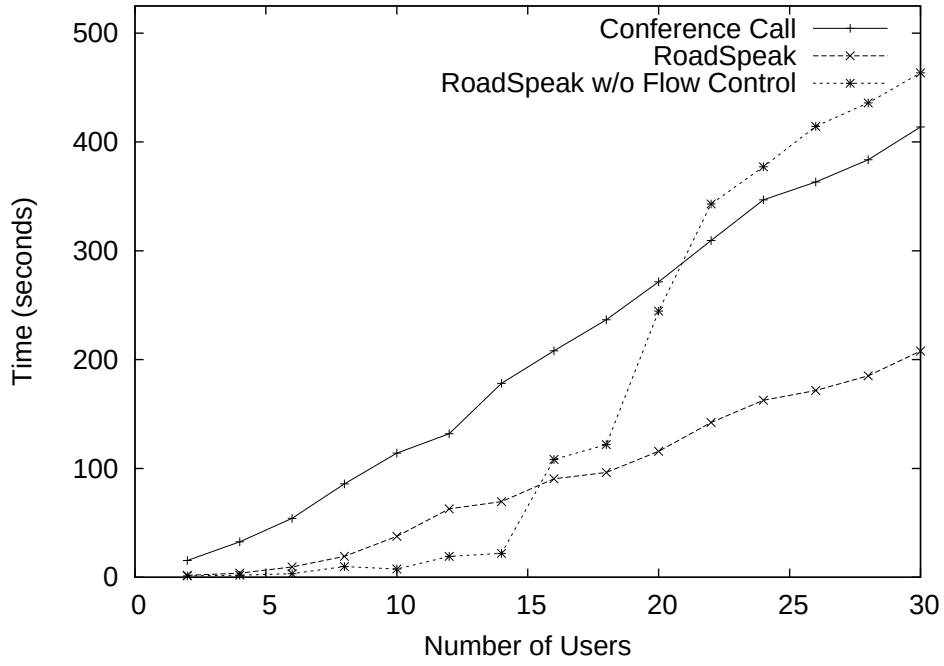


Figure 4.8: Message Completion Time.

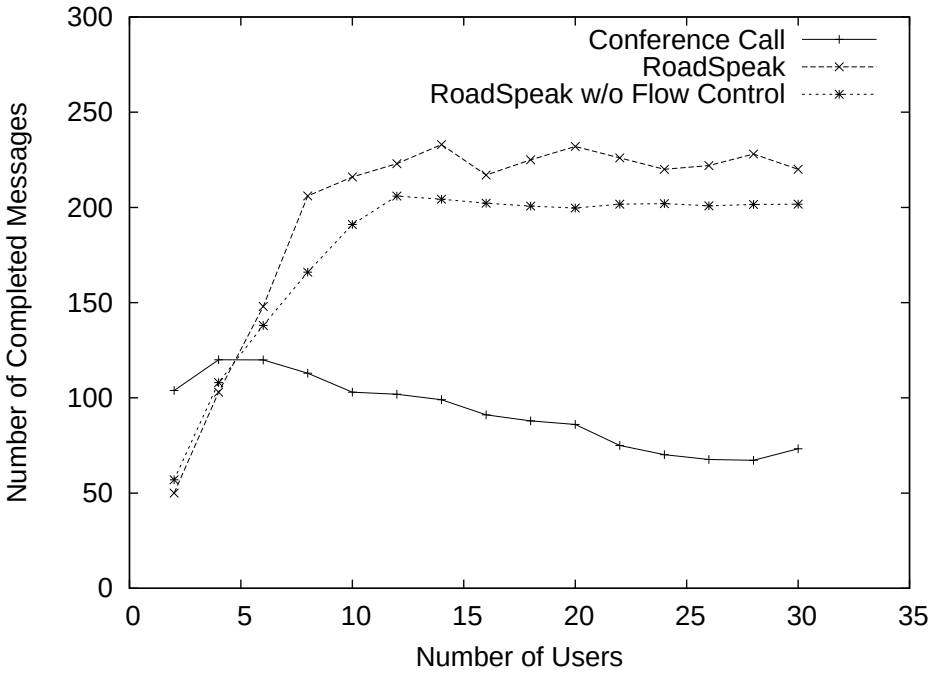


Figure 4.9: Message Completion Rate.

RoadSpeak also performs better than RoadSpeak w/o Flow Control beyond 15 speakers, due to the fact that flow control enables stable scaling, preventing server overload and subsequent

performance degradation.

Finally, we compare the message completion rates for the three scenarios in Figure 4.9. Here we observe that, as the number of speakers in a group increases, both RoadSpeak cases demonstrate a higher message completion rate than does Conference Call (beyond five speakers). As the number of speakers increases, the rate of audio collisions for Conference Call also increases, causing the message completion rate to drop dramatically. RoadSpeak also consistently outperforms RoadSpeak w/o Flow Control, maintaining a higher message completion rate, throughout. Finally, the message completion rate for both RoadSpeak cases is ultimately limited by the size of the client receive buffer capacity, once the rate of concurrent message generation exceeds 60 seconds worth of messages generated per minute.

4.7 Discussion

RoadSpeak vs. Conference Call Systems

RoadSpeak allows interruption-free communication, and employs automated moderation, message buffering, and flow control to enable the system to scale to large groups. On the other hand, classical real-time multiparty voice communication systems, such as conference call systems, allow audio collisions, and shift the onus onto the speakers (or a human moderator, if available) to resolve these collisions. There is an inherent tradeoff between RoadSpeak and other MVC systems. The interruption-free model employed by RoadSpeak can often result in out-of-order conversations, since old messages may remain in the system. This is because the RoadSpeak server, unlike a human moderator, cannot understand the semantics of a message. Other MVC systems, on the other hand, do not scale well and can experience high collision rates with large groups. We believe that this tradeoff brings RoadSpeak and other MVC systems into different domains. RoadSpeak is more applicable for entertainment or socializing purposes by large numbers of users, while conference call systems are more suitable for business usage with a limited number of participants.

Geo-Matching

Motorists need to focus on the road while they are driving, so the system cannot expect users to manually select a vehicular social network group. To aid the drivers, geo-matching and spatial-temporal enforcement mechanisms are used to: *(i)* provide automatic group joining, and *(ii)* allow only valid users currently driving on the proper roadway to join a group constrained to that roadway. When a user logs into the RoadSpeak system from her smart phone, the RoadSpeak client connects to the server to retrieve the waypoints for each group that user has registered and stores the waypoints in an R-Tree in local smart phone storage. Periodically, the RoadSpeak client reads GPS locations and compares these coordinates with the waypoints stored in the R-Tree to determine if the motorist's current location matches the spatial-locality preferences of any of her preferred groups. When a match occurs, the client automatically joins the user to the group.

Spam

Unwanted messages such as advertisements and spam represent another potential issue in RoadSpeak. Reputation systems are often useful in large online communities in which users may frequently have the opportunity to interact with other users with whom they have no prior experience. In such a situation, it is helpful to base the decision whether or not to interact with that user on the prior experiences of other users. Reputation systems may also be coupled with an incentive system to reward good behavior and punish bad behavior. RoadSpeak can establish and utilize a reputation system to prevent unwanted messages. Users of RoadSpeak can be assigned reputations at the beginning. Low ratings from other users may diminish one's reputation. When the reputation of a user is below a certain threshold, RoadSpeak should not publish messages from him, or should revoke his admission to the group.

Driver Distraction

Although it is important to consider the safety of drivers who use cell phones while driving, we believe that, with advances in automated vehicular control (e.g., automatic cruise control, obstacle avoidance, drive-less cars, etc.), driver distraction will be less of an issue in the near

future.

4.8 Summary

In this chapter, we proposed the concept of Vehicular Social Networks, which enable people traveling along the same roadways at the same time to form virtual mobile communities. We also present the design, implementation, and evaluation of RoadSpeak, a mobile, inter-vehicle, multiparty voice communication system that allows motorists to socialize with each other along popular roadways by utilzing automated moderation, flow control and voice message buffering. We presented the mechanisms to discover and form RoadSpeak socialization groups, which take into account both time and locality, along with user interests. We implemented a customized Real-time Multi-party Voice Communication simulator for comparing the scalability of RoadSpeak with traditional MVC systems. Our experimental results show that the use of voice message buffering, interruption-free communication and flow control allows RoadSpeak to scale to large groups of users.

Chapter 5

Conclusions and Future Directions

In this dissertation, we have identified the practical and theoretical challenges in two emerging unconventional Online Social Networks, Distributed Online Social Networks and Vehicular Online Social Networks. We have presented novel approaches to address these challenges, and demonstrated their feasibility and effectiveness through user studies, benchmarks, emulation and simulation experimentations.

The main contribution of this dissertation is to explore the design of the two infrastructure for efficient social content dissemination and to enable new social presences using both theoretical and practical approaches. To answer several challenges posed by DOSN infrastructure, we propose Social Butterfly and SocialCDN, which use Social Caching techniques for efficient social content dissemination in DOSNs. To expand the online social network application domains, we proposed Vehicular Social Networks, which provide a platform to form social network groups for socialization of vehicular drivers, especially daily commuters. To satisfy the requirements for real-time communication over Vehicular Social Networks, we proposed RoadSpeak, which enables real-time multi-party voice communication by utilizing automated moderation and flow control.

- **Social Butterfly.** Building Distributed Online Social Networks (DOSNs) brings many challenges to the OSN *Core Service* and *Database/Storage* layers. In this dissertation, we addressed the challenge of efficient social update distribution in DOSNs using *Social Caches*. Social Caches are the selected nodes in a DOSN that act as "local bridges" for data dissemination. The neighbors of the Social Caches push their own social updates to the caches, and fetch social updates of their friends from the corresponding Social Caches. We formulate the selection of Social Caches as the *Neighbor Dominating Set* problem, and proposed the *Approximate NDS* and the *Social Score* centralized algorithms

to solve the problem. We also presented the social properties we discovered about social updates and social relationships in Chapter 2. Simulation on a real online social network dataset showed that the use of Social Caches can reduce the social network connections by an order of magnitude.

- **SocialCDN.** The selection of Social Caches using centralized algorithms requires knowledge of the entire social graph, which, when reflected to the infrastructure level, requires a peer-to-peer look-up server. In Chapter 3, we proposed a set of fully distributed social cache selection algorithms, which only requires the node to know its local information, i.e., its one hop neighbors. The four algorithms, namely Randomized, Triad Elimination, Span Elimination, and Social Score, select Social Caches based on different social properties. We also presented the empirical evaluation results of the four algorithms on five well known graphs: the Facebook, Enron email, co-author, citation, and AS graphs. The Span Elimination algorithm has the least time complexity in terms of communication steps, and selects the least number of Social Caches for any given graph. The Social Score algorithm works better on a social graph.

- **Vehicular Social Network.** The *Application Service* layer controls what applications are supported by an OSN. Although current OSNs cover a large domain of applications, we believe that they have not yet reached the vehicular drivers who commute between home and office on a daily basis. In Chapter 4, we proposed Vehicular Social Networks (VSNs) to expand the existing OSN infrastructure to reach these drivers. Drivers who are in the same proximity can use the Vehicular Social Networks to form mobile virtual communities, VSN groups, which are defined by enforcing temporal and spatial regulations in the format of a triple $<$*road segment, time duration, interest*$>$. That is, motorists who are driving on the same road segment at the same time and are willing to socialize about particular interests can form VSN groups. VSN can be used for many purposes, such as entertainment, or emergency alerts. In an entertainment VSN, users can share with their VSN friends information related to the group topic. In the emergency situation, only emergency traffic information messages are allowed. We also presented mechanisms to discover and form these groups in Chapter 4.

- **RoadSpeak.**

  Real-time multiparty voice communication systems (MVC), such as a conference call, enable small groups of users to participate in real-time, simultaneous conversations. However, the main drawback to current MVC systems is that they do not allow meaningful communication to scale up to a large number of users because of *audio collisions*. We presented our findings about audio collisions in a group of users in Section 4.4.1, and discovered that the collision rate increases as the number of users increases in the group. We proposed RoadSpeak, which utilizes *message buffering*, *automated moderation* and *flow control* as to enable inter-vehicle *collision-free* multiparty voice communication in Vehicular Social Networks. Buffers at both the client side and the server side enable scalable and collision-free multiparty voice communication by enforcing certain rules, which regulate both an individual's personal behaviors and her social behaviors. Client-side buffers enable users to pause/resume/re-order/cancel both incoming and outgoing messages to provide a customized social experience. Server-side buffers control the voice communication message sequence and eliminate staled or junk messages in each group, thereby enabling meaningful conversation. RoadSpeak also allows important control messages to be sent by the server to control the buffers at the client side. We designed and implemented a RoadSpeak prototype, as well as both server-side and client-side automated moderation and flow control schema. A field trial was also performed to prove the feasibility of RoadSpeak. In order to compare RoadSpeak and other Multiparty Voice Communication System, we implemented an MVC simulator, and the simulation results showed that RoadSpeak can scale to a larger group of users.

The main contribution of this dissertation is exploring challenges in two unconventional Online Social Networks, Distributed Online Social Network and Vehicular Social Networks, by utilizing theoretical and practical approaches. More specifically, the dissertation makes two contributions: we proposed Social Butterfly and SocialCDN, a set of centralized and fully distributed Social Caching techniques, respectively, that provide efficient dissemination of social updates over DOSNs. Second, we proposed Vehicular Social Networks (VSN), a platform that

enables drivers to socialize; and RoadSpeak, an application which utilizes automated moderation and flow control to enable real-time, ephemeral, and multiparty voice communication over Vehicular Social Networks (VSNs).

5.1 Future Directions

Looking forward, we believe the success and development of Online Social Networks are relying heavily on the infrastructure, performance and functionalities. Future work on OSNs will range from Application Service Infrastructure, Social Presence Infrastructure, and Core Service Infrastructure to Database/Storage Infrastructure. We believe other novel OSNs will also be proposed as a result of the growth of the OSNs to satisfy the requirements of both users and providers.

This dissertation suggests future work in two areas: smarter Social Caching and enhanced Vehicular Social Networks.

5.1.1 Smarter Social Caching

Social Caches are utilized in both Social Butterfly and SocialCDN for the efficient dissemination of social updates in Distributed Online Social Networks. The social cache selection problem, like the Neighbor Dominating Set problem, is solved by both the proposed centralized and the distributed approximation algorithms. The design of the approximation algorithms utilized several social properties, such as the clustering coefficient, the egocentric betweenness centrality, the social score, the transitive triad and the span of a node. There are several other potential social properties that we can use for smarter selection of social caches. *(i)* Social traffic patterns. The algorithms we proposed in this dissertation did not consider social traffic patterns. Future work should include incorporating social traffic patterns into the design of the cache selection algorithm, since nodes that generate more social traffic should have a greater chance of being selected as caches. *(ii)* Social tie strength [58]. The algorithms presented in this dissertation treated all users in the social graph the same, as either friends or strangers. In reality, relationships could fall anywhere along this spectrum. Another potential direction for social cache selection includes using social tie strength as one of the criteria, and selecting

friends with strong social ties as social caches. *(iii)* Network dynamics and availability. The dynamic of peers, online or offline, known as *churn*, is an inherent property of any P2P network, including the Distributed OSN. P2P churn and the availability model should also be added to social cache selection. *(iv)* Limited resources. In this dissertation, we assumed that each node in the network had the ability to cache social updates for friends. However, in reality, nodes may have limited storage, bandwidth, etc., and can only cache social updates for k friends. This is yet another issue that should be considered in social cache selection.

5.1.2 Enhanced Vehicular Social Networks

The automated moderation and flow control schema is used by RoadSpeak to enable collision-free real-time multiparty voice communication in Vehicular Social Networks. Besides enabling the MVC, another potential application for RoadSpeak is to enable *Traffic Audio Tweets*. Today, there are a few early stage Audio Twitter services, such as Tvider [142], and Chirbit [25]. They allow users to record audio clips to share with friends. Compared with traditional text tweets, audio tweets contain more social presence and are more convenient. Traffic audio tweets differ from existing ones in several ways: and corresponding buffering techniques are required. *(i)* Traffic audio tweets are temporal messages, and need to be heard by other parties before the traffic event disappears. Otherwise, the tweets are stalled tweets and will mislead the drivers. Thus, the buffer needs to be able to separate and remove stalled audio tweets from the non-stalled ones. *(ii)* The volume of traffic audio tweets is strongly correlated with traffic situations. For example, if there are no traffic incidents or jams, very few traffic audio tweets will be generated. However, when there is a traffic accident, a burst of tweets may be generated by drivers passing by. Therefore, the buffer must be able to eliminate repeated messages, or send *digests* to the clients to avoid sending redundant messages.

References

[1] *Characterizing and Modeling Multiparty Voice Communication for Multiplayer Games*, Berlin, Germany, 08/10/2008 2008.

[2] Swarup Acharya and Stanley B. Zdonik. An efficient scheme for dynamic data replication. Technical report, Providence, RI, USA, 1993.

[3] Application layer protocols. "http://en.wikipedia.org/wiki/Category:Application_layer_protocols".

[4] Appleseed. http://opensource.appleseedproject.org/.

[5] The 20 best music social networks. http://pulse2.com/2010/02/11/the-20-best-music-social-networks/.

[6] Enkh-Amgalan Baatarjav, Ram Dantu, and Santi Phithakkitnukoon. Privacy management for facebook. In *Proceedings of the 4th International Conference on Information Systems Security*, ICISS '08, pages 273–286, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 181–190, New York, NY, USA, 2007. ACM.

[8] Roberto Battiti and Franco Mascia. Reactive and dynamic local search for max-clique: Engineering effective building blocks. *Computers & Operations Research*, 37:534–542, March 2009.

[9] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proc. VLDB Endow.*, 2(1):766–777, August 2009.

[10] F. Biocca and K. Nowak. Plugging your body into the telecommunication system: Mediated embodiment, media interfaces, and social virtual environments. *Communication technology and society*, pages 407–447, 2001.

[11] Biznik. "http://www.biznik.com".

[12] Blogspot. http://www.blogspot.com.

[13] C. Boldrini, M. Conti, and A. Passarella. Impact of social mobility on routing protocols for opportunistic networks. AOC Workshop, 2007.

[14] Cristian Borcea, Ankur Gupta, Achir Kalra, Quentin Jones, and Liviu Iftode. The mobisoc middleware for mobile social computing: challenges, design, and early experiences. 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, 2008.

[15] Dario Bottazzi, Rebecca Montanari, and Alessandra Toninelli. Context-aware middleware for anytime, anywhere social networks. *IEEE Intelligent Systems*, 22(5):23–32, September 2007.

[16] P. Boustead, F. Safaei, and M. Dowlatshahi. Dice: Internet delivery of immersive voice communication for crowded virtual spaces. IEEE Virtual Reality 2005 Proceedings (VR2005),, 2005.

[17] John Bowers, James Pycock, and Jon O'Brien. Talk and embodiment in collaborative virtual environments. SIGCHI conference on Human factors in computing systems: common ground table of contents, 1996.

[18] Danah M Boyd and Nicole B Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13:210–230, 2008.

[19] danah michele boyd. Friendster and publicly articulated social networking. In *CHI '04 extended abstracts on Human factors in computing systems*, CHI EA '04, pages 1279–1282, New York, NY, USA, 2004. ACM.

[20] Branchout. "http://www.branchout.com".

[21] Sonja Buchegger, Doris Schiöberg, Le Hung Vu, and Anwitaman Datta. Peerson: 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*, Nürnberg, Germany, March 31, 2009.

[22] Alina Campan and Traian Marius Truta. A clustering approach for data and structural anonymity in social networks. In *In Privacy, Security, and Trust in KDD Workshop (PinKDD*, 2008.

[23] Clustering coefficient.

[24] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *7th ACM MOBICOM*, Rome, Italy, July 2001.

[25] Chirbit. "http://www.chirbit.com".

[26] Ching chuan Chiang and Mario Gerla. Routing and multicast in multihop, mobile wireless networks. In *in Multihop, Mobile Wireless Networks, in Proc. IEEE ICUPC '97*, 1997.

[27] The kdd competition, citation graph. "http://www.sommer.jp/graphs/".

[28] Classroom. "http://www.classroom.com".

[29] The dblp computer science bibliography coauthor graph.

[30] Cofoundr. "http://www.cofoundr.com".

[31] Leudo Antonio Cutillo, Refik Molva, and Thorsten Strufe. Privacy preserving social networking through decentralization. In *Proceedings of the Sixth international conference on Wireless On-Demand Network Systems and Services*, 2009.

[32] cyworld. "http://www.cyworld.com".

[33] E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. MobiHoc, 2007.

[34] Introduction to social network methods. http://datasift.com/.

[35] Online dating vs. social networking. http://wraltechwire.com/business/tech_wire/opinion/story/2449164/.

[36] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical Review Letters*, 16, April 2005.

[37] Diaspora.

[38] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, and Bill Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 2002.

[39] Ben Dodson, Ian Vo, T.J. Purtell, Aemon Cannon, and Monica Lam. Musubi: disintermediated interactive social feeds for mobile devices. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 211–220, New York, NY, USA, 2012. ACM.

[40] Distributed online social networks list. "http://en.wikipedia.org/wiki/Distributed_social_network".

[41] Distributed online social networks list.

[42] Distributed social networking protocol.

[43] Distributed social networking protocol.

[44] Distributed social networking technologies.

[45] Dtn. http://en.wikipedia.org/wiki/Delay-tolerant_networking.

[46] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, 2005.

[47] eharmony. http://www.eharmony.com.

[48] Facebook. "http://www.facebook.com".

[49] Facebook hits 900 million users .

[50] J. FARBER. Network game traffic modelling. 1st workshop on Network and system support for games, 2002.

[51] Fbi seeks data-mining app for social media. "http://www.gnip.com/".

[52] Facebook app leak user info.

[53] Flickr. http://www.flickr.com.

[54] Freenode irc. http://freenode.net/.

[55] friendster. "http://www.informationweek.com/government/security/fbi-seeks-data-mining-app-for-social-med/232500552".

[56] When did facebook become so uncool? http://www.cnn.com/2012/04/10/tech/social-media/facebook-uncool-instagram/index.html?hpt=hp_bn11.

[57] Palla Gergely, Imre Derényi, and Tamás Vicsek. The critical point of k-clique percolation in the erdos-renyi graph. *Journal of Statistical Physics*, 128(1-2):219–227, 2007.

[58] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the Conferece on Human Factors in Computing Systems (CHI '09)*, page 220, 2009.

[59] Glassdoor. "http://www.glassdoor.com".

[60] Gnip. "http://www.gnip.com/".

[61] Google+. "https://plus.google.com/".

[62] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, WPES '05, pages 71–80, New York, NY, USA, 2005. ACM.

[63] 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.

[64] Keith N. Hampton, Lauren Sessions Goulet, Lee Raine, and Kristen Purcell. Social networking sites and our lives. *Pew Internet and American Life Project*, June 2011.

[65] Lu Han, Badri Nath, Liviu Iftode, and S. Muthukrishnan. Social butterfly: Social caches for distributed social networks. In *SocialCom/PASSAT*, Boston, MA, October 2011.

[66] Lu Han, Magdalena Punceva, Badri Nath, S. Muthukrishnan, and Liviu Iftode. Socialcdn: Caching techniques for distributed social networks. In *P2P*, pages 191–202, 2012.

[67] Lu Han, Stephen Smaldone, Pravin Shankar, James Boyce, and Liviu Iftode. Ad-hoc voice-based group communication. In *PerCom*, pages 190–198. IEEE Computer Society, 2010.

[68] Derek Hansen, Ben Shneiderman, and Marc A. Smith. *Analyzing Social Media Networks with NodeXL: Insights from a Connected World*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[69] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114, August 2008.

[70] Carrie Heeter. Being there: the subjective experience of presence. *Presence: Teleoper. Virtual Environ.*, 1(2):262–271, May 1992.

[71] P. W. Holland and S. Leinhardt. Transitivity in structural models of small groups. *Small Group Research*, 2:107–124, 1971.

[72] Pan Hui, Eiko Yoneki, Shu Yan Chan, and Jon Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, MobiArch '07, 2007.

[73] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 125–138, New York, NY, USA, 2006. ACM.

[74] imeem. `http://www.imeem.com`.

[75] Instagram. `http://www.instagram.com`.

[76] Instagram hits 15 million users on ios. `http://www.macgasm.net/2011/12/07/instagram-hits-15-million-users-ios-working-android-app/`.

[77] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65. ACM, 2007.

[78] Jobster. `"http://www.jobster.com"`.

[79] Dean Povey John and John Harrison. A distributed internet cache. In *Proceedings of the 20th Australian Computer Science Conference*, pages 5–7, 1997.

[80] Kaixin. `"http://www.kaixin001.com"`.

[81] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *In First Conference on Email and Anti-Spam (CEAS) (2004)*, 2004.

[82] Klout. `"http://www.klout.com/home"`.

[83] David Knoke and Song Yang. *Social network analysis*, volume 154 of *Quantitative applications in the social sciences*. Sage, Los Angeles, CA, 2nd ed edition, 2008.

[84] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09. ACM, 2009.

[85] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[86] Vidya Lakshmipath and Chris Schmandt. Simphony: a voice communication tool for distributed workgroups. Ubiquituous Computing, 2004.

[87] S.-B. Lee, G. Pan, J.-S. Park, and M. Gerla. Secure incentives for commercial ad dissemination in vehicular networks. MobiHoc, 2007.

[88] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, 2005.

[89] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, July 2003.

[90] LinkedIn. "http://www.linkedin.com".

[91] Worldwide membership of linkedin. http://press.linkedin.com/about.

[92] Wiki social network sites list. http://en.wikipedia.org/wiki/List_of_social_networking_websites.

[93] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[94] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *In Proceedings of ACM SIGMOD*, pages 93–106, 2008.

[95] Livejournal. http://www.livejournal.com.

[96] 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.

[97] P.V. Marsden. Egocentric and sociocentric measures of network centrality. *Social Networks*, 24(4):407–422, 2002.

[98] match. "http://www.match.com".

[99] E. Michael Maximilien, Tyrone Grandison, Kun Liu, Tony Sun, Dwayne Richardson, and Sherry Guo. Enabling privacy as a fundamental construct for social networks. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, CSE '09, pages 1015–1020, Washington, DC, USA, 2009. IEEE Computer Society.

[100] Teemu Karkkainen Mikko Pitkanen Md. Tarikul Islam, Anssi Turkulainen and Jorg Ott. Practical voice communications in challenged networks. 1st Extreme Workshop on Communication, 2009.

[101] Giuliano Mega, Alberto Montresor, and Gian Pietro Picco. Efficient dissemination in decentralized social networks. In *Proc. of the 11th IEEE P2P Conference on Peer-to-Peer Computing (P2P'11)*, pages 338–347. IEEE, August 2011.

[102] Nasrullah Memon, Jennifer Jie Xu, David L. Hicks, and Hsinchun Chen. *Data Mining for Social Network Data*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[103] A. G. Miklas, K. K. Gollu, K. K. W. Chan, S. Saroiu, K. P. Gummadi, and E. de Lara. Exploiting social interactions in mobile systems. Ubicomp, 2007.

[104] Mixi. "http://www.mixi.com".

[105] J. Moody and D.R. White. Structural cohesion and embeddedness: A hierarchical concept of social groups. *American Sociological Review*, pages 103–127, 2003.

[106] MySpace.

[107] Myworkster. "http://www.myworkster.com".

[108] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: A scalable traffic monitoring system. MDM, 2004.

[109] Rammohan Narendula, Thanasis G. Papaioannou, and Karl Aberer. My3: A highly-available P2P-based online social network. In *Proc. of the 11th IEEE International Conf. on Peer-to-Peer Computing (IEEE P2P'11)*, 2011.

[110] Network traffic distribution.

[111] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *Phys. Rev. E*, 68(3), Sep 2003.

[112] Ning. http://www.ning.com.

[113] Alexandra Olteanu and Pierre Guillaume. Towards Robust and Scalable Peer-to-Peer social networks. In *SNS'12*.

[114] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.

[115] C. E. Palazzi, M. Roccetti, S. Ferretti, G. Pau, and M. Gerla. Online games on wheels: Fast game event delivery in vehicular ad-hoc networks. V2VCOM, 2007.

[116] Path. http://www.path.com/.

[117] Christopher Peery, Francisco Matias Cuenca-Acuna, Richard P. Martin, and Thu D. Nguyen. Wayfinder: Navigating and sharing information in a decentralized world. In *Proceedings of Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2004.

[118] Picasa. "http://www.picasa.com".

[119] pinterest. http://www.pinterest.com/.

[120] Push to talk over cellular. http://en.wikipedia.org/wiki/Push_to_talk.

[121] Ren ren. http://www.renren.com.

[122] M. Roccetti, M. Gerla, C. E. Palazzi, S. Ferretti, and G. Pau. Crystal ball: How to scry the emergency from a remote vehicle. IPCCC 2007 / NetCri07, 2007.

[123] ryze. "http://www.ryze.com".

[124] Oliver Schneider. Trust-aware social networking: A distributed storage system based on social trust and geographic proximity. FU Berlin, Berlin, Germany, January, 22, 2009.

[125] John P. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, January 2000.

[126] 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, 2010.

[127] John Short, Ederyn Williams, and Bruce Christie. volume 206. John Wiley & Sons Ltd, 1976.

[128] skyrock. "http://www.skyrock.com".

[129] Stephen Smaldone, Lu Han, Pravin Shankar, and Liviu Iftode. Roadspeak: enabling voice chat on roadways using vehicular social networks. In *Proceedings of the 1st Workshop on Social Network Systems*, SocialNets '08, pages 43–48, New York, NY, USA, 2008. ACM.

[130] Introduction to social network methods. http://faculty.ucr.edu/~hanneman/nettext/.

[131] Social media. http://en.wikipedia.org/wiki/Social_media.

[132] Soundclick. http://www.SoundClick.com.

[133] Soundcloud. http://www.soundcloud.com.

[134] Spotify. http://www.spotify.com.

[135] Roni Stern, Meir Kalech, and Ariel Felner. Searching for a k-clique in unknown graphs. In *the International Symposium on Combinatorial Search (SoCS)*, pages 83–89, 2010.

[136] Teachstreet. http://www.teachstreet.com.

[137] Termwiki. http://www.termwiki.com.

[138] Michael Terry and Elizabeth D. Mynatt. Social net: Using patterns of physical proximity over time to infer shared interests. In *In Proceedings of Human Factors in Computing Systems (CHI 2002*, 2002.

[139] Transport layer protocols. "http://en.wikipedia.org/wiki/Category:Transport_layer_protocols".

[140] tripadvisor. http://www.tripadvisor.com.

[141] M. Tsvetovat and A. Kouznetsov. *Social Network Analysis for Startups: Finding Connections on the Social Web*. Real Time Bks. O'Reilly Media, 2011.

[142] Tvider. "http://www.tvider.com".

[143] Twitter. "http://www.twitter.com".

[144] Twitter twittes dataset.

[145] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks, 2000.

[146] Ventrilo. http://www.ventrilo.com.

[147] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 37–42, New York, NY, USA, 2009. ACM.

[148] Vkontakte. http://www.vt.com.

[149] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Number 8 in Structural analysis in the social sciences. Cambridge University Press, 1 edition, 1994.

[150] Weibo. "http://www.weibo.com".

[151] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge Univ Press, 2011.

[152] Xing. "http://www.xing.com".

[153] yelp. http://www.yelp.com.

[154] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. In *IEEE INFOCOM*, 2004.

[155] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. ACM/IEEE MSWiM, 2007.

[156] Jungkeun Yoon, Brian Noble, and Mingyan Liu. Surface street traffic estimation. *MobiSys*, 2007.

[157] Youtube. http://www.youtube.com.

[158] Pooya Moradian Zadeh and Mohsen Sadighi Moshkenani. Mining social network for semantic advertisement. In *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology - Volume 01*, ICCIT '08, pages 611–618, Washington, DC, USA, 2008. IEEE Computer Society.

[159] Word of mouth and brand advocates stats. "http://www.zuberance.com/resources/resourcesStats.php".