UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

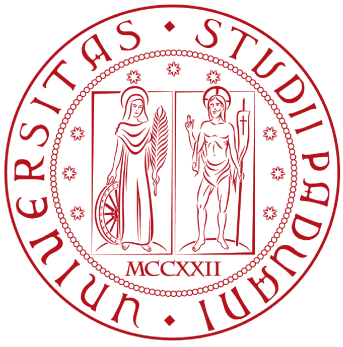
Dipartimento di Matematica

CORSO DI DOTTORATO DI RICERCA IN: Scienze Matematiche
CURRICOLO: Informatica
CICLO: XXIX

Social Relations to Improve Network Resiliency against Failures and Censorships

Coordinatore: Ch.mo Prof. Pierpaolo Soravia
Supervisore: Ch.mo Prof. Mauro Conti

Dottorando: Ding Ding



University of Padua
Department of Mathematics
Doctorate Degree in Mathematics
Curriculum in Computer Science

Social Relations to Improve Network Resiliency against Failures and Censorships

Candidate
Ding Ding

Supervisor
Prof. Mauro Conti
University of Padua, Italy

March 2017

Acknowledgments

First and foremost I want to thank my advisor Prof. Mauro Conti, for his patience, guidance, motivation, and immense knowledge during my Ph.D. study in University of Padua. It has been an honor to be his Ph.D. student. His guidance helped me in all the time of research and writing of this thesis. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the Ph.D. pursuit.

In the second year of my Ph.D., I visited Advanced Computing and Information Systems Laboratory, at the University of Florida (Gainesville - FL, USA) as a Visiting Researcher. There I had the opportunity to know and work with many brilliant people that I also want to thank. I want to thank Prof. Renato Figueiredo for being an invaluable guide for many of my works, a friend, and a coauthor of many of my papers. Also I am grateful to the collaborators and friends I have met in Gainesville: Kyuho Jeong, and Saumitra Aditya.

In the last year of my Ph.D., I visited Lab for behavioral Authentication, Machine learning, and Privacy (LAMP), at New York Institute of Technology (New York City - NY, USA) as a Visiting Researcher. I want to thank Prof. Paolo Gasti and Prof. Kiran Balagani, for countless meetings to stimulate ideas, and thank all the people I have met in New York City.

I thank my Ph.D. colleagues in the SPRITZ group (Moreno, Alberto, Hossein, Riccardo S. and Kanishka), for all the stimulating discussions, for all the fun we have had in the last three years, and for those fabulous coffee, Spritz and Tiramisus.

Last but not the least, a special thanks to my family. Words cannot express how grateful I am to my parents for all of the sacrifices that you've made on my behalf. At the end I would like express appreciation to my beloved wife Chenyun Zhao, for supporting me spiritually throughout writing this thesis and my life in general.

Ding Ding
New York City, Jan 31, 2017

Abstract

During the last decade, the Internet have penetrated our life. An increasing number of people relies on the Internet and its applications. Many client/server based Internet services, such as e-commerce (e.g., Amazon) and social networking (e.g., Facebook), are carefully provisioned and managed to deliver high availability. However, in the recent years, there have been situations where country-scale network fraction of Internet users have been disconnected from the rest of the network. In particular, because of national censorships, natural hazards, misconfiguration or DDoS attacks on links, the network is splitted into two or more areas, and consequently, web services/applications may become unavailable to users. This severely undermine the ability of people to communicate and organize, since many of the services used to discover and relay messages to other users (e.g., Twitter, Facebook) were unavailable. For example, because of a censorship, people in the censored area were not able to use microblogging service (i.e., Twitter). However, owing to the news media property of microblogging, under this situation (i.e., governments performing censorship), people are more willing to rely on this service, in order to spread information. Therefore, these web services/applications, need to be carefully provisioned to maintain high availability, against network failures or censorships.

This dissertation focuses on improving the network resiliency against the network failures/partitions. More precisely, the contribution of this thesis is twofold: (i) we present our design of a social-aware P2P overlay that is able to provide better connectivity comparing with existing structured P2P overlays, and introduce a bootstrapping method for the proposed overlay; and (ii) based on social-aware overlay, we present two applications that are able to work in different network partition scenario. In particular, we present a decentralized microblogging system, named SAND and a decentralized censorship circumvention system, named SeND.

In the first part of this thesis, we focus on solutions to recover the connectivity within a partitioned area. In particular, we present a social-

aware overlay where users have private communication channels to their social friends, enabling virtual private network communication among social peers (even through NATs and firewalls). We compared the connectivity of social-aware overlay with other structured P2P overlays (i.e., Chord and SPROUT). We carried our an extensive simulation that shows the structured P2P overlays routability is severely hampered by country-scale partition events. The proposed social-based unstructured overlay network provides improved routability while maintaining a smaller number of links. Moreover, we present a bootstrapping method for the proposed social-aware overlay. In particular, our bootstrap method allows nodes to easily join the overlay network, by leveraging close neighbors with public IP addresses.

In the second part of this thesis, we present two representative applications based on social-aware overlay. First, we present SAND, a social-aware, network-failure resilient, and decentralized microblogging system. Compared with other decentralized microblogging systems, SAND has the following advantages: (i) SAND is designed for (and hence is able to handle) scenarios where massive correlated failures occur; (ii) the delivery rates of SAND is significantly high (i.e., with SAND-SN, a variant of SAND, peers are able to effectively follow each others updates with 100% delivery rate); and (iii) we evaluated SAND on partitioned networks based on a ground-truth dataset with a real publisher-subscriber distribution. Then, we present SeND, a social network friendship enhanced decentralized system to circumvent censorships. In order to be resilient to current censorship techniques, such as IP address blocking and active probing attacks, with SEnD, users in an uncensored area can act as proxy servers for their social friends in a censored area, allowing them to bypass the censorship. We assessed the effectiveness of SEnD through extensive simulations based on a synthetic dataset, as well as through experiments based on a prototype implementation. We built our synthetic dataset based on parameters obtained from questionnaires administered both inside and outside China (we consider China as a case study of censored area).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the proliferation of applications (e.g., e-commerce and social networking) that rely on the Internet, the connectivity of networks has become an elementary requirement in today's societies [75]. These applications are carefully provisioned and managed to deliver high availability [32, 95]. However, events can happen where a localised portion of the network is partitioned from the rest of the Internet [168, 165]. In particular, these network partition events are caused by (but not limit to) the following reasons:

- **National censorships** [45, 12]: in general, users may be subject to censorship by government or private enterprise entities in control of the network infrastructure that underlies (a subset of) the Internet [126]. For example, during Arab Spring in 2011, in order to limit the spread of news, Egyptian government performed a censorship to their citizens. Egypt — a country with a population of 80 million, including 23 million Internet users — was partitioned from the Internet [11].
- **Natural hazards** [44, 23]: natural hazards such as earthquake and hurricane may destroy the physical network cables [18]. For example, in 2006, a 7 magnitude earthquake struck off the southwest coast of Taiwan [110], and hence Eight submarine cables were cut because of the earthquake. It was catastrophically disrupted Internet services in Asia, affecting many Asian countries.
- **Mis-configurations** [122, 112]: misconfiguration at BGP speakers could result in network failures because of errors in how external BGP information was being propagated to routers inside an autonomous system (AS) [71].

- **Distributed Denial of Service (DDoS) attacks on links** [101, 119]: By flooding only few network links, Crossfire Attack [92] is able to degrade and often cut off network connections to a variety of selected server targets (e.g., servers of an enterprise, a city, a state, or a small country).

These different network partition scenarios result in failure of client/server based applications, if the servers of such applications are unreachable [36]. This situation severely undermines the ability of people to communicate and organize, since many of the services used to discover and relay messages to other users (e.g., Twitter, Facebook) were unavailable [105]. For example, people in partitioned area were not able to use microblogging service (e.g., Twitter) if Twitter servers are located outside of partition area. Owing to the news media property [98] of microblogging services, under these extremely situations (e.g., censorships, and natural hazards) that result in network partitions, people are more willing to rely on this service, in order to spread information [109, 37, 140].

In order to improve the network resiliency against the network failures/partitions, in this thesis, we present a social-aware overlay that is a unstructured peer-to-peer overlay based on social relationship. In particular, by leveraging SocialVPN (IP-over-P2P) tunnels [91], users have private IP channels to their social friends, enabling virtual private network communication among social peers (even through NATs [67] and firewalls [82]).

1.1 Research Motivations and Contribution

The research work presented in this thesis focuses on improving network resiliency against failures and censorships. In particular, in this dissertation, we first present social-aware overlay and compare the connectivity of social-aware overlay with other stuctured P2P overlays (i.e., Chord [142], SPROUT [113]). Hence we discuss the possible bootstrapping issues [128] of social-aware overlay, and present a novel bootstrapping method based on social friends. Finally, based on social-aware overlay, we propose two application: 1) SAND, a Social-Aware, Network-failure resilient, and Decentralized microblogging system; 2) SEnD: a Social network friendship Enhanced Decentralized censorship circumvention system. These applications are decentralized, and are able to work in different network partitions scenarios.

1.1.1 Impact of Country-scale Internet Disconnection on Structured P2P Overlay and Social-aware Overlays

Peer-to-peer systems are resilient in the presence of churn and uncorrelated failures [102]. However, their behavior in extreme scenarios where massive

correlated failures occur is not well-studied [74]. Yet, there have been examples of situations where a country-scale fraction of Internet users have been disconnected from the rest of the network [118]—for instance, when a government cuts connectivity to the outside world as a mechanism for suppression of uprisings.

Contribution: In order to improving network resiliency against network partitions, we propose a social-aware overlay. We consider the effect of such partitions on topology and routing of structured P2P overlay (i.e., Chord and SPROUT) and social-based unstructured P2P overlays (i.e., social-aware overlay). In particular, we consider nodes within a relatively small fraction of the network (2.5% or fewer of the Internet users), and study whether users can communicate with their (n-hop away) social neighbors in a peer-to-peer fashion after the partition. We perform an extensive simulation-based analysis to assess the probability for these communications to be possible. In our analysis, we consider both real and synthetic datasets of online social networks. Our results show that structured P2P overlay routability is severely hampered by country-scale partition events. In addition, the proposed social-based unstructured overlay network provides improved routability while maintaining a smaller number of links. Our evaluation is presented in Chapter 2.

1.1.2 Discussing the bootstrapping issue of social-aware overlay

Bootstrapping is an important functionality required by P2P overlay networks [19]. Nodes intending to participate in such an overlay network initially have to find at least one node that is already part of this network [42]. While structured P2P networks (e.g., Distributed Hash Tables, DHTs) define rules about how to join a overlay [56], unstructured P2P networks still rely on bootstrapping servers until overlay nodes are sufficiently connected [108].

Contribution: We propose and evaluate a method that addresses the overlay bootstrapping problem under the constraints imposed by firewalls and NATs. In particular, we leverage social friends with public IP addresses as Session Traversal Utilities for NAT (STUN) or Traversal Using Relays around NAT (TURN) servers to assist nodes in bootstrapping into overlay. Furthermore, in the evaluation of bootstrapping, by varying several input parameters, from 90% to 97% of nodes in social-aware overlay could bootstrap and reach their direct-neighbors. We present the design of this bootstrapping method in Chapter 3.

1.1.3 SAND: Social-Aware, Network-Failure Resilient, and Decentralized Microblogging System

To overcome the limitations (e.g., vulnerability to central server failure) of current existing microblogging systems (e.g., Twitter [6], and Weibo [7]), researchers have proposed several Peer-to-Peer (P2P) microblogging solutions [13, 163, 14]. However, these solutions also have several issues: (i) they were not designed for (and hence cannot handle) scenarios where massive correlated failures occur [123]; or (ii) their delivery rates were not significantly high (i.e., lower than 85%); or (iii) their working mechanisms were not evaluated on partitioned networks based on a ground-truth dataset with a real publisher-subscriber distribution.

Contribution: We propose SAND: a microblogging system builds upon an overlay where users have private IP connections to their social friends, enabling trusted social communication even on partitioned networks. In addition, we assess the availability of SAND through a simulation-based analysis considering a real-world dataset crawled from Twitter, as well as a synthetic dataset that provides high flexibility to tune network parameters. The results show that SAND is feasible and efficient, also in case of network partitions. For example, in a partitioned network where 98% of network nodes fail, by using SAND-SN, peers are able to effectively follow each others updates (i.e., 100% delivery rate), with acceptable overheads for all published messages in terms of communications (i.e., on average only 66 sent messages/20 received messages per node) and storage (i.e., on average only 5.8 copies of messages stored per node). We present the design and evaluation of SAND in Chapter 4.

1.1.4 SEnD: a Social Network Friendship Enhanced Decentralized System to Circumvent Censorships

The Internet has evolved to play a significant role in providing a medium for spreading news and organizing political activities [10]. Politicians, activists and citizens use the Internet to coordinate their activities and share their ideas in ways that are not otherwise possible via traditional media [97]. For instance, during the Arab Spring in 2011, several mainstream websites (e.g., Facebook, Twitter, YouTube) had a great influence on distributing important information and news [159]. Therefore, dictatorships regard free accessing to Internet content and services as a serious threat [29]. While the Internet is open by design, it is still the case that users can be subject to censorship by governments or enterprises in accessing services and data.

Contribution: In this thesis we propose SEnD, a fully-distributed censorship circumvention system. This system is resilient to current censorship techniques, such as IP address blocking [153] and active probing attacks [158]. It builds upon an overlay, where users have peer-to-peer virtual

private communication tunnels to proxies within their social network. With SEnD, users in an uncensored area can act as proxy servers for their social friends in a censored area, allowing them to bypass the censorship. We assessed the effectiveness of SEnD through extensive simulations based on a synthetic dataset, as well as experiments based on a prototype implementation. We built our synthetic dataset based on parameters obtained from questionnaires administered both inside and outside China (we consider China as a case study of censorship area). The results show that SEnD is feasible, efficient and scalable. For example, when the proportion of concurrent active users is less than 60%, 99.9% of these users are able to find proxy servers. The design and evaluation of SeND are presented in Chapter 5.

1.2 Other Research Contributions of the Candidate

During the Ph.D. program, the candidate contributed to the solutions of other research area that are not included in this thesis. In the following, we briefly introduce these works.

Security and Privacy Challenges of Brain-Computer Interface: Brain-Computer Interfaces (BCI) are becoming increasingly popular in medical and non-medical areas [148, 152]. Unfortunately, manufacturers of BCI devices focus on application development, without paying much attention to security and privacy related issues [24]. Indeed, an increasing number of attacks to BCI applications underline the existence of such issues [114]. For example, malicious developers of third-party applications could extract private information of users [93]. We focused on security and privacy of BCI applications. In particular, we classified BCI applications into four usages scenarios: 1) neuromedical applications, 2) user authentication, 3)gaming and entertainment, and 4) smartphone-based applications. For each usage scenario, we discussed security and privacy issues and possible countermeasures.

Privacy Issues of Smart Health: Together with the development of technologies such as those for ubiquitous computing, data mining, Internet of Things (IoT) and wireless sensor networks (WSNs), the concepts of smart cities and mobile health (m-Health) have emerged [68]. Along the same line, the smart health concept (s-Health), understood as a context-aware healthcare paradigm for smart environments, improves the quality of healthcare systems within smart cities [16]. However, s-Health may encounter some privacy and security issues [138]. For example, in order to obtain the current location and health conditions of citizens, these citizens might be continuously monitored, which could be seen as a privacy invasion [61]. We described an application within the s-Health paradigm. In particular, our

approach allows to effectively deal with citizens who have respiratory conditions. Our application example suggests low-pollution routes to citizens in order to lessen their respiratory-related problems, and proactively activates water sprays in fountains to reduce the effect of pollution or pollen. Besides the description of the application, the main contribution of the article is the analysis of the emerging privacy issues of the proposed application and the discussion of possible countermeasures.

Maximize Lifetime of Wireless Sensor Networks: In order to reduce the energy consumption in Wireless Sensor Networks (WSNs) [54], researchers have proposed several approaches based on duty cycle operation (i.e., sensor nodes periodically switch between sleeping mode and awake mode) [144]. However, these solutions also have several issues: (i) they assign the same duty cycle ratio to all the nodes, hence they did not consider the energy hole problem; or (ii) they assign different duty cycle ratios while failing in considering the energy consumption in network construction phase; or (iii) they do not rely on the cluster-based structure [167]. We propose DDR — a hierarchical clustering approach leverages duty cycle operation to maximize lifetime of WSNs. In particular, we assigned different duty cycle ratios for nodes according to their distance to the sink. Moreover, we were the first to model the energy consumption considering both *network construction phase* and *data gathering phase*. We assessed DDR through extensive simulation-based analyses on the OMNet++ platform. The result shows that, DDR is feasible and efficient. For example, with DDR, the lifetime of the network extends 32% comparing with the approach applied with uniform duty cycle, without prolonging the end-to-end delay.

1.3 Outline

The organization of this dissertation is as follows.

In Chapter 2, we present the social-aware overlay, and compare the connectivity of social-aware overlay with other P2P structured overlays. In particular, Section 2.1 summarizes the related work. In Section 2.2, we give several definitions to better evaluate connectivity of the network outage. In Section 2.3 we introduce the overlays we considered in the evaluation. Next, in Section 2.4 we explain how we prepare the network datasets. In Section 2.5, we analyze the basic parameters for the partition. In Section 2.6, we describe our simulations and analysis to evaluate different P2P overlay approaches. We discuss the potential improvement of the topology and routing for the overlay networks in Section 2.7. Finally, we conclude the chapter in Section 2.8.

In Chapter 3, we present a novel bootstrapping method for users to join social-aware overlay. In particular, in Section 3.1, we summarizes the related works. In Section 3.2, we introduce two definitions, and discuss the

types of nodes we considered in this chapter. In Section 3.3, we present our bootstrapping method. We evaluate our method in Section 3.4. In Section 3.5, we discuss the incentives for the users to help other people. Finally, we conclude the chapter in Section 3.6.

In Chapter 4, we present SAND, a social-aware, network-failure resilient, and decentralized microblogging system. In particular, Section 4.1 summarizes the related work. Section 4.2 describes the main design of SAND, and three variants of SAND: SAND-BR, SAND-PF and SAND-SN. In Section 4.3, we discuss the dataset used in the evaluation, and how we select a partitioned area. In Section 4.4, we first introduce the experimental setup, and then report the results of our evaluation. In Section 4.5, we first discuss how a user bootstraps into the overlay after partitions in SAND. Then we discuss the effect of offline nodes. Aftermost, we point out incentives for the end users to adopt SAND. Finally, we conclude the chapter in Section 4.6.

In Chapter 5, we present SEnD, a social network friendship enhanced decentralized system to circumvent censorships. In particular, In Section 5.1, we discuss the background of this work. Section 5.2 summarizes the state of the art. Section 5.3 describes the main design of SEnD. Section 5.4 discusses the preparation of the simulation. In Section 5.5, we report the evaluation results. In Section 5.6, we discuss the effect of offline nodes, and point out incentives for supporters to help their friends/friends of friends to bypass censorship. Then we show a method to increase the number of available supporters. Finally, we conclude the chapter in Section 5.7.

1.4 Publications

Part of the research presented in this thesis and developed during my PhD program produced peer-reviewed workshop, conference and journal publications. The complete list of published and currently submitted works, is shown in Section 1.4.1 (conference and workshop publications) and Section 1.4.2 (magazine and journal publications).

1.4.1 Conference and Workshop Publications

[C1] D. Ding, M. Conti, and A. Solanas. Impact of Country-scale Internet Disconnection on Structured and Social P2P Overlays. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, WoWMoM 15, pages 19, 2015 **(Acceptance Rate 21.00%)**

[C2] Q. Li, D. Ding, and M. Conti. Brain-Computer Interface Applications: Security and Privacy Challenges. In *1st Workshop on Security and Privacy in Cybermatics*, IEEE CNS/SPiCy 15, 2015

[C3] D. Ding, M. Conti, and A. Solanas. A Smart Health Application and its Related Privacy Issues. In *Smart City Security and Privacy Workshop*, IEEE SCSP-W 16, pages 15, 2016.

1.4.2 Magazine and Journal Publications

[J1] D. Ding, M. Conti, and R. Figueiredo. Country-scale Internet Disconnection: Impact and Recovery on P2P Overlays. Under Submission at: IEEE Transactions on Networking.

[J2] D. Ding, M. Conti, and R. Figueiredo. SAND: Social-Aware, Network-Failure Resilient, and Decentralized Microblogging System. Under Submission at: IEEE Transactions on Services Computing.

[J3] D. Ding, K. Jeong, S. Xing, M. Conti, R. Figueiredo, and F. Liu. SEnD: a Social Network Friendship Enhanced Decentralized System to Circumvent Censorships. Under Submission at: ACM Transactions on Internet Technology.

[J4] Q. Li, D. Ding, S. Prasad, M. Conti, and F. Liu. DDR: a Hierarchical Clustering Approach based on Different duty cycle ratios to Maximize Lifetime of Wireless Sensor Networks. Under Submission at: Computer Communications.

Chapter 2

Impact of Country-scale Internet Disconnection on Structured P2P Overlay and Social-aware Overlays

The Internet has become pervasive in our daily life, and an increasing number of people heavily relies on it [117]. Many client/server based Internet services, such as e-commerce and social networking, are carefully provisioned and managed to deliver high availability [129]. However, events can happen where a localised portion of the network is partitioned from the rest of the Internet, due to natural hazards [41], national censorships [45], etc. For example, on the evening of January 27, 2011, Egypt—a country with a population of 80 million (around 1.1% of world population), including 23 million Internet users (around 0.64% of world Internet users)—was partitioned from the Internet. The connectivity of the network *inside* the country was recovered only after five days [21]. In this case, for the people in Egypt, the *outside* networks—and any services hosted outside the country—were unavailable [20]. This severely undermined the ability of people to communicate and organize, since many of the services used to discover and relay messages to other users (e.g., Twitter, Facebook) were unavailable [124, 27].

In related work, the authors in [151, 59] study bypass routing paths for users *outside* the partition as a mitigation strategy. However, these works do not try to recover the connectivity *within* the partitioned network. In fact, users inside the outage area would benefit from using a P2P overlay to improve the likelihood of communicating with each other, even if they cannot reach the outside network [103]. However, it remains unclear how well a P2P overlay can recover connectivity within a relatively small outage partition

(e.g., Egypt disconnected from the rest of the world) [74]. Users within a small region that is partitioned away from the larger P2P overlay may become unable to communicate with their peers within the partition. This can be caused by: 1) routing outages in the overlay; 2) the inability to find suitable nodes to bootstrap into the overlay; or 3) connectivity constraints imposed by firewalls and Network Address Translators (NATs) that prevent peer-to-peer communication.

In this chapter, we study the impact of country-scale partitioning on P2P communication among people that are socially related (e.g., relatives, friends and colleagues). In particular, we focus on analyzing the ability of users within the partition to reach their n-hop away social friends, through P2P overlay messaging, after the partition event. The underlying assumption is that users are more likely willing to communicate with close friends [157]. Furthermore, they may reach friends-of-friends and peers that are further away in the social graph, in order to: 1) help bootstrap them into the overlay; 2) assist in endpoint discovery/exchange for NAT traversal; or 3) store/relay messages in order to implement features such as a decentralized Twitter-like micro-blogging system [141].

In our study, we consider three representative P2P overlays: structured, i.e., *Chord* [142]; structured with social friendships added, i.e., *SPROUT* [113]; and unstructured overlay with social friend-to-friend links, i.e., *social-aware* overlay. We point out that there is an incentive in using our approach, even before the network outage. In normal circumstances, the social-based overlay network provides a motivation for users to bootstrap into and maintain connections over time, prior to a partition event. For instance, the open-source SocialVPN [91] and commercial P2P VPNs such as Hamachi [3] demonstrate that these systems have usefulness in personal computer environments, by establishing end-to-end secure virtual networks among users to allow them to share data, play games, and access remote desktops privately with peers. In a mobile computing environment, the incentive for users to maintain their overlay VPN connectivity is that it allows devices to serve data (e.g., photos, streaming media), query data, and receive notifications (e.g., private updates) from social peers. Furthermore, in addition to overlay links to devices of social peers, the unstructured overlay can accommodate ad-hoc links for connections established through short-lived relationships, e.g., based on geographical proximity. For instance, in a recent event that happened in Hong Kong, over 100,000 people uses Firechat to communicate over ad-hoc links [1]; these transient links may be created after a partition happens, in contrast to the more persistent links that social peers can establish and maintain prior to partition.

Our evaluation is based on a dataset derived from a real online social network: Foursquare [2]. Furthermore, we also use a synthetic dataset based on Foursquare network metrics. The synthetic dataset allows for flexibility in the evaluation, where we can select input parameters to generate networks

with different metrics. We then perform a comprehensive assessment of the topological and routing connectivity inside the partitioned network.

Figure 2.1 gives a bird's eye view of the networks and overlays considered in this chapter. In Figure 2.1, the physical network level is at the bottom. It illustrates physical distribution of endpoint devices and the links among them. In the top part of Figure 2.1, we represent the social relations among people who own devices. For ease of exposition, in this chapter we assume that a person uses only one communicating device. Nonetheless, the methodology provides a basis for studying the behavior with multiple devices as discussed in Section 2.7. In the middle of Figure 2.1, we illustrate the different overlay networks considered in our work, and the mapping between physical and overlay network. Finally, Figure 2.1 shows the device mapping for Alice (only this mapping is shown for clarity).

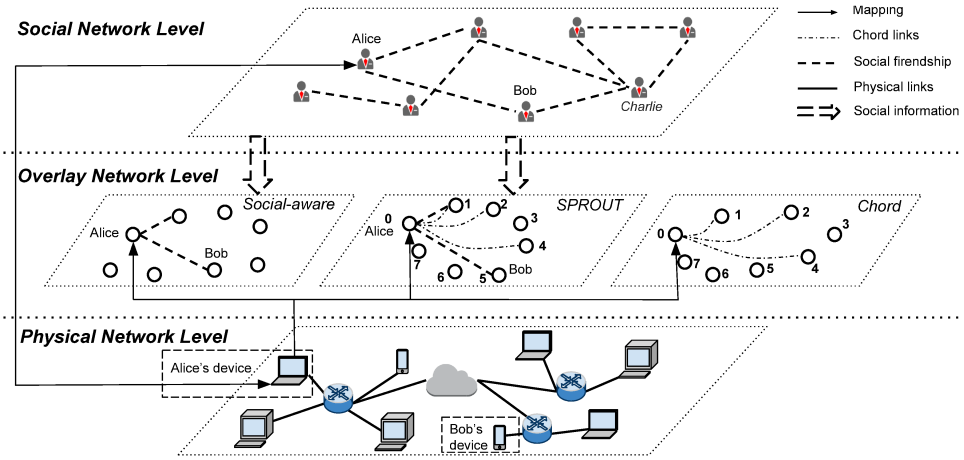


Figure 2.1: Overview of the different considered layers: physical network (endpoints and physical network links), social network (friendships), and the three overlay networks (two of which leverage social network level information).

Contribution: To the best of our knowledge, this chapter is the first to assess topological connectivity and message routability in structured and social P2P overlays, *within* a country-scale network that is partitioned from the Internet. In particular, we assess the connection probability of social peers via Chord and SPROUT overlays (structured), as well as in a novel social-aware overlay (unstructured) that we propose in this chapter as an extension of SocialVPN [91]. For all simulations performed, the results show that, in small partitions including less than 2.5% of overall nodes, Chord overlay network is almost totally disconnected. Furthermore, social-aware overlay network with geographical routing achieves better probability of successful message delivery compared to SPROUT overlay network. Considering that the social-aware overlay network requires fewer links than SPROUT does

(the former is a sub-graph of the latter), and other routing algorithms in social-aware overlay network may provide improved routability, we conclude that a social-aware overlay with unstructured routing is a better approach to handle such outages.

2.1 Related Work

Several previous works proposed the methods that use the topology *outside* the outage area to mitigate the influence of partition network disconnectivity. In [151], the authors proposed a robust algorithm to efficiently improve reliability of the network by utilizing redundancy deployed in advance. RiskRoute [59] mitigates consequences of Internet outages by proactively identifying routes that avoid areas of high network risk. However, these works do not analyze or recover connectivity *within* the partitioned area.

Structured peer-to-peer overlay networks [142, 116] are resilient with respect to routing in the presence of churn (a small fraction of nodes in the network fails). Furthermore, P2P systems are able to recover connectivity after a partition event when relative large fractions of nodes are in the partition, e.g., a sudden partition that splits an overlay in two partitions of the same size [111]. However, the recovery behavior in a very small fraction (e.g., 1%) of the overlay has not been considered; this is the focus of this chapter.

In [28, 43], authors proposed distributed social networking (DOSN) systems, which are designed to provide OSN capabilities (e.g., the ability to post news, follow friends) in a decentralized way. In our chapter, rather than creating a DOSN, we leverage existing social networking infrastructures to bootstrap a communication overlay that connects users in a peer-to-peer fashion. While this overlay can be potentially used as a layer upon which to build a DOSN, in itself it is not a DOSN—it only provides a means for private peer-to-peer end-to-end communication among social peers, leveraging existing centralized OSNs for discovery of users/devices and for public key exchange. Moreover, the techniques described in [28, 43] are based on structured Distributed Hash Table (DHT), the key technique of Chord. However, in our study, we demonstrate that Chord becomes disconnected within a small outage area.

2.2 Definitions and Notations

The social and overlay networks studied in this chapter can be defined as $G^s = (V^s, E^s)$, $G^{oX} = \left(V^{oX}, E^{oX}\right)$, respectively, where X is one of the following: Chord (CH), SPROUT (SP) and social-aware (SA). V^s (V^{oX}) is set of vertices/nodes in the social (overlay) network, while E^s (E^{oX}) is set of edges in the social (overlay) network. For instance, $G^{oCH} = \left(V^{oCH}, E^{oCH}\right)$

is the notation used for Chord overlay network. Besides Chord, in our simulation, we also implement SPROUT and social-aware overlays, represented as $G^{oSP} = \left(V^{oSP}, E^{oSP}\right)$ and $G^{oSA} = \left(V^{oSA}, E^{oSA}\right)$, respectively.

In our quantitative analysis, we make use of the following concepts to evaluate the topological and routing connectivity of social peers after the partitioning.

Definition 1. *n-**social-hop friends**: In the social graph G^s, if two nodes $v_0, v_n \in V^s$ can be connected by an n-hop shortest social path $p_s = (v_0, ..., v_n)$, v_0 and v_n are defined as n-**social-hop friends**.*

For example, Alice and Bob are directly connected in the social network as shown in Figure 2.1. In other words, they are 1-*social-hop friends*. Alice and Charlie are not direct friends, but they have a common neighbor—Bob. Then, we refer to Alice and Charlie as 2-*social-hop friends*. In this chapter, we analyze the probability that a node v_0 can reach its n-***social-hop friend*** (v_n) *after* the partition.

Definition 2. *m-**overlay-hop path**: In the overlay graph G^{oX}, $u_0, u_m \in V^{oX}$ are overlay nodes mapped from n-**social-hop friends**. If there exists a m-hop overlay path $p_o = (u_0, ..., u_m)$ (m is the shortest overlay path), we refer to u_0, u_m as being connected by a m-**overlay-hop path**.*

For example, in Figure 2.1, Alice (node 0) and Bob (node 5) are connected by a 1-*overlay-hop path* in SPROUT overlay and social-aware overlay networks. However, in the Chord overlay network, the *overlay path* between Alice (node 0) and Bob (node 5) has 2 *hops* (Note that, to keep the figure simple we do not draw the Chord link between node 4 and node 5). In the simulation, we analyze the probability of finding the *m-overlay-hop path* between *n-social-hop friends* in the overlay after partition.

Definition 3. ***After Partition Topological-connected Friends Probability** (AP-TFP) refers to the probability that, after the partition, in the overlay network there exists an m-**overlay-hop path** between two n-**social-hop friends**, where $v_0(u_0)$, $v_n(u_m)$ are within the partitioned area. Note that this path may not be discoverable by the overlay's routing algorithm.*

Definition 4. ***After Partition Routing-connected Friends Probability** (AP-RFP) refers to the probability that, after the partition, applying the P2P overlay's routing algorithm, $v_0(u_0)$ can successfully route a message to its n-**social-hop friend**, $v_n(u_m)$ by an m-**overlay-hop path**, where $v_0(u_0)$ and $v_n(u_m)$ are in the partitioned area.*

In our simulation, we calculate the *AP-TFP* and *AP-RFP* for *n-social-hop friends* ($n = 1, 2, 3, 4$) considering the overlay that remains within the partitioned network. In other words, when the partition happens, we consider whether for *n-social-hop friends* there exists an *m-overlay-hop path*

(*AP-TFP*), and whether such path is discoverable by the routing algorithm (*AP-RFP*).

2.3 Considered overlay networks

We evaluate the connectivity of network outages based on Chord [142] and SPROUT [113] overlays, as well as the social-aware overlay we proposed in this chapter.

Chord Overlay Network: In Chord, the overlay is organized into a ring-based topology. It uses a simple identifier-based routing to allow nodes to reach arbitrary nodes in the network, and supports services such as DHT. However, routing in Chord relies on structured links, which can be severely disrupted if the network is partitioned into a small sub-graph.

SPROUT Overlay Network: The SPROUT overlay network builds upon Chord: starting with Chord links, it adds links to online friendships from social network (i.e., Foursquare) to augment the network topology. The SPROUT routing protocol follows the same identifier-based structured routing approach of Chord, except that social links have higher priority to route messages, since they are considered more trusted. Thus, routing can also be significantly disrupted by a country-scale partition. Since we need to compare with the social-aware network and the links in such overlay is undirected, we consider the directed Chord links as undirected in the Chord and SPROUT overlays.

Social-aware Overlay Network: We propose a novel social-aware overlay network as an extension of SocialVPN [91]. In particular, we use the social friendships to form overlay edges, resulting in an unstructured topology (thus, the social-aware and Chord overlay topologies can be seen as sub-graphs of SPROUT). Overlay edges in such systems allow 1-hop friends to communicate directly to each other over a private tunnel, as in SocialVPN, where an online social network provider can be used to discover and exchange self-signed VPN public key certificates with friends, but communication among friend's devices is peer-to-peer. As an extension, we leverage the user's location information to perform geographical routing [106] through overlay peers. Note that there are additional routing protocols that can be used on the social-aware overlay network, such as source-based routing [65], limited-token flooding [65] and multi-dimensional routing [87]. While we focus on one social-aware overlay protocol (geographical routing) in the connection probability section, other routing protocols may provide improved performance (at the expense of more messages and/or higher bandwidth). In Section 2.7, we discuss routing protocols (on social-aware overlay) in more detail. One of the advantages of the social-aware unstructured overlay is that the topology is built such that: 1) links are established based on friendship and trust, increasing the resiliency against

Sybil attacks; and 2) the topology exploits the probability distribution of friends with respect to geographical proximity [15] to maintain connectivity within the partitioned area.

2.4 Considered Datasets

Our evaluation is inspired by the Egyptian Internet censorship case. Around 1.1% of worldwide population (0.6% of worldwide Internet users) were located within the Egyptian geographic boundaries where the partition took place [45]. In order to simulate the connectivity of such a partition, the initial number of nodes in the dataset must be large. Furthermore, we also need geographical information for every node to decide whether a node is in the partition, and to route messages in the social-aware topology. In this chapter, we analyze the connectivity inside the partition considering both a real social network dataset (obtained from Foursquare online social network) and a synthetic dataset.

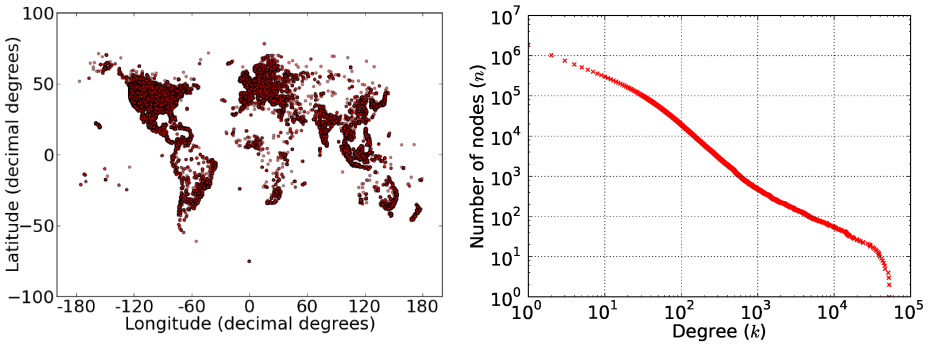
2.4.1 Foursquare Dataset

We have obtained the Foursquare dataset from [104, 136]. It has been chosen because it provides geo-location information associated with nodes. The raw data consist of 2,153,469 users and 27,098,473 directed edges. In order to analyze networks with undirected edges, we convert the directed edges to undirected ones using the following principle: if there are bi-directional edges between two users, we can assume they are friends to each other; otherwise, we ignore the edges. With this approach, we trim the redundant data as follows. First, if there is only one directed edge between two users, we assume that they are not friends to each other. Furthermore, we delete users whose location information are missing and whose degree are zero. As a result, the network used as input to our simulations includes 1,852,592 users and 8,640,350 undirected edges, with the clustering coefficient 0.105. Figure 2.2a shows the geographical distribution of the Foursquare users in the obtained dataset. Figure 2.2b shows the degree distribution of the Foursquare dataset. For each degree k, there are n number of users that have at least k number of friends. The average degree of this network is 9.3.

Since the Foursquare dataset does not provide temporal information about online and offline nodes, we assume that all the nodes are of online users in our simulation. Therefore, we calculate the *AP-TFP* and *AP-RFP* assuming an overlay network spanning the whole Foursquare network.

2.4.2 Synthetic Dataset

We also use a synthetic dataset since it can provide greater flexibility in the choice of parameters. We can tune the input parameters to generate net-



(a) Users geo-distribution (b) Degree distribution

Figure 2.2: Foursquare network metrics.

works with different metrics (e.g., different number of nodes in the partition, different average degree and average clustering coefficient).

In order to generate and analyze synthetic networks, we use NetworkX [4], a third-party library for Python. We generate a social network (including nodes and their friendships) based on a modified *Nearest Neighbor Model* [134]. This model follows the observation that two people sharing a common friend are more likely to become friends. We create the synthetic dataset using metrics that approximate those found in the Foursquare network. The average degree and average clustering coefficient of the initial synthetic network are approximately 10 and 0.16, respectively.

In addition, we create an algorithm to assign the location information to each node. Our algorithm is based on the following observation [15]: shorter is the distance among two nodes, higher is the probability for them to be friends. The procedure to determine geo-location information for each node is detailed in Algorithm 6. For each node i in the *nodesList* ($1 \leq i \leq N$; N is the total number of nodes in the network), the algorithm assigns location information to it. If at least one of i's friend has been assigned location information already, the algorithm assigns the location information to i based on one of i's random friends, $neighborWithLocation$. The geo-location information is assigned according to literature [15] that i is more likely to be closed to $neighborWithLocation$ (line 11 in Algorithm 6). Otherwise, if all the i's friends have not yet been assigned location information, the algorithm will assign i the geographical location through uniform distribution in the network (line 14 in Algorithm 6). The location information in synthetic network is based on the Cartesian coordinate system.

Figure 2.3 shows the node distribution in the considered synthetic dataset and how we simulate the partitioning event. Nodes are distributed in a Cartesian coordinate system. The outage area that includes k nodes is defined as: the smallest circular area that has the center in the origin and

includes k nodes. Therefore, after the partition, we get two subnetworks from the whole network: the outage area, and the outside area. The size of the partition can be controlled by adjusting the number of nodes in the partition. Therefore, we can evaluate the *AP-TFP* and *AP-RFP* on different sizes of partitions.

Algorithm 1 Assign geo-location information for each node.

```
1: procedure geneLocation(nodesList)
2:     nodesWithLocation ← ∅
3:     for each node ∈ nodeList do
4:         neighborsWithLocation ← ∅
5:         neighbors ← node.neighbors()
6:         for each neighbor ∈ neighbors do
7:             if neighbor ∈ nodesWithLocation then
8:                 neighborsWithLocation.add(neighbor)
9:             end if
10:        end for
11:        if len(neighborsWithLocation) > 0 then
12:            neighborWithLocation ←
                   randomSelect(neighborsWithLocation)
13:            node.setLocationBased(neighborWithLocation)
14:        else
15:            node.setLocationRandomly()
16:        end if
17:        nodesWithLocation.add(node)
18:    end for
19:    return nodeList
20: end procedure
```

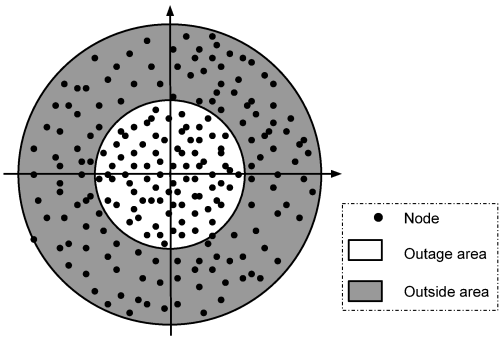


Figure 2.3: Example of node distribution and outage area modeled in the synthetic network (edges are not shown).

2.5 Analysis of Basic Network Parameters

In this section, we analyze the basic parameters of different network partitions in both Foursquare and synthetic networks. Note that, in the graph reported in this chapter, error bar indicates confidence interval (95%).

2.5.1 Foursquare Dataset

The Foursquare dataset contains information for users distributed across the world, Since it contains a sample of a relatively small number of nodes, it is not representative to simply take a particular country as a candidate partition. Instead, we have identified a geographical area that yields approximately the same fraction of nodes of the entire network—a state in the United States. In our evaluation, in order to simulate the Egyptian Internet censorship case, we consider the states of Indiana or Louisiana as partition area. Our choice is motivated by the fact that the percentage of nodes in Indiana over the entire dataset (1.2%) is very close to the one of Egyptian population over the whole worlds. Similarly, the percentage of nodes in Louisiana (0.6%) over the entire dataset is very close to the one of Egyptian Internet users over the whole worlds. The average degree and average clustering coefficient of Indiana are 4.1 and 0.1, while the ones of Louisiana are 3.4 and 0.095. All the nodes in this network are mapped onto a ring with 2,097,152 (2^{21}) positions in the Chord and SPROUT overlays.

2.5.2 Synthetic Dataset

We generate the synthetic network according to the description in Section 2.4.2. Due to limitations in the simulator's platform, the initial network includes 1,000,000 nodes. In the Chord and SPROUT overlay networks, by leveraging consistent hashing function, all the nodes are mapped onto a ring that has a total of 1,048,576 (2^{20}) positions. The placement is done using a hash function for randomness. The details of the initial network parameters are summarized in Table 2.1.

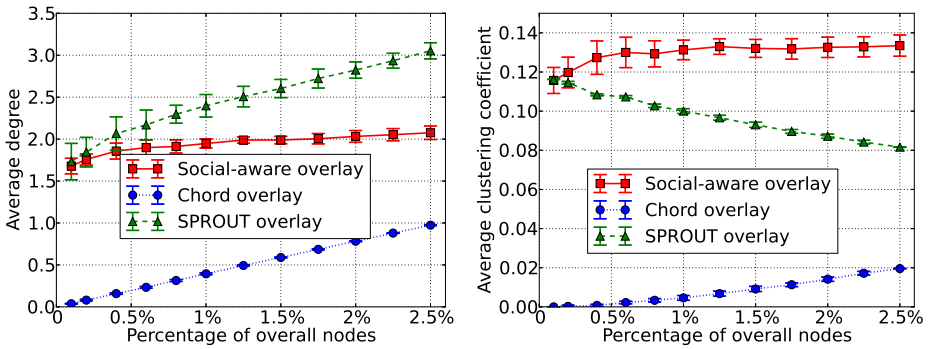
Parameters	Average Degree	Average Clustering Coefficient
Chord	38.95	0.08
Social-aware	9.80	0.16
SPROUT	48.75	0.06

Table 2.1: Synthetic network metrics before partition.

Figure 2.4 shows changes of average degree and average clustering coefficient, considering partitions with different numbers of nodes. In particular, from Figure 2.4a, we can observe that the average degree of social network is greater than the one of Chord network, when the number of nodes in the

outage area is less than 25,000 (2.5%). The average degree of three overlay networks grows while the partition size increases. This is because the larger the number of nodes in the partition, the higher the possibility for path to exist between nodes and their neighbors.

The average clustering coefficients are shown in Figure 2.4b. In [15], authors show that friendships in social network is such that, nodes that are geographically closer to each other have higher probability to become friends. In contrast, there is no concept of social-friendship nor geographical location in Chord network; hence, links are established solely on the basis of identifiers.



(a) Degree (b) Clustering coefficient

Figure 2.4: Synthetic network metrics as a function of fraction of nodes within outage area.

2.6 Evaluation of Connection Probability

In the Foursquare network, we consider Indiana and Louisiana as the partition areas. The synthetic network has 1,000,000 nodes, and we model partition areas that include 1,000 to 25,000 nodes (0.1% to 2.5% of nodes). We carry on simulations to analyze the *AP-TFP* and *AP-RFP* on each size of partition.

In the Foursquare dataset, each simulation run is based on the same topology, while in the synthetic dataset, in each simulation run we create a network with new topology. In our evaluation, we perform 50 simulation runs for both Foursquare and synthetic dataset. For each simulation, first, we randomly select 5,000 pairs of *n-social-hop friends* ($n = 1, 2, 3, 4$) located in the outage area. Then, we consider only nodes and links inside the outage area. Consider t, to be number of pairs of *n-social-hop friends* that can be connected from topological point of view, the value of *AP-TFP* is given by $t/5000$. Analogically, if there are r pair of *n-social-hop friends* that can be connected from routing point of view, the *AP-RFP* is $r/5000$. For these

metrics, in the following we report the average of the results obtained for the 50 simulation runs.

2.6.1 After Partition Topological-connected Friends Probability (AP-TFP)

In this section, we first consider the Chord overlay and connectivity among *n-social-hop friends*. Since Chord does not include any social information, both *AP-TFP* and *AP-RFP* are not related to the number of hops between the *social friends*. Simulation results show that, for Chord, both *AP-TFP* and *AP-RFP* are 0 when the fraction of nodes in the partition is less than 2.5%. As we increase the fraction of nodes in the partition, Figure 2.5 shows that in the Chord overlay there is an increase in *AP-TFP* when the fraction of nodes in the partition is greater than 3%. Nonetheless, while connected from a graph perspective, friends are not routable (*AP-RFP* is near 0%). For the Foursquare network with both Indiana (1.2% of overall nodes) and Louisiana (0.6% of overall nodes) as the outage areas, simulation shows that the *n-social-hop friends* cannot be connected by topology in the Chord overlay.

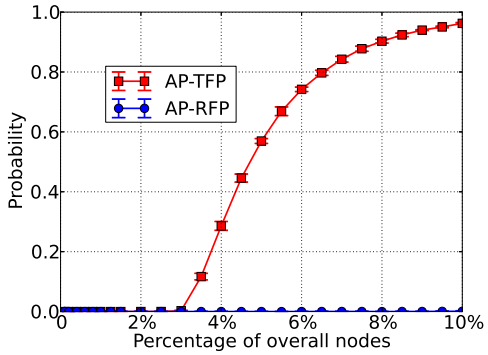
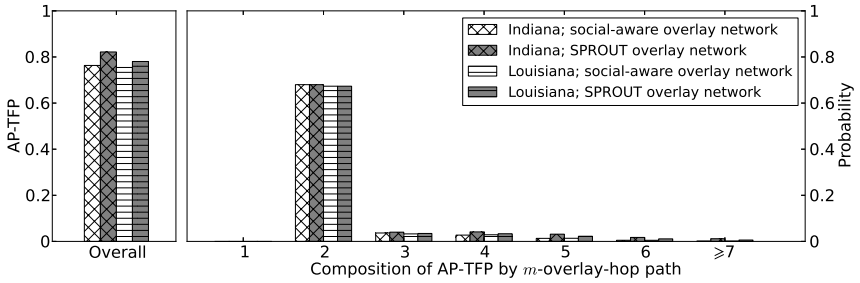
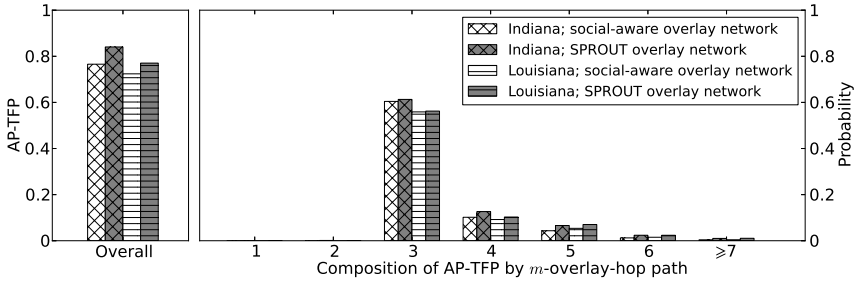


Figure 2.5: *AP-TFP* and *AP-RFP* on Chord overlay; synthetic dataset; 0%-10% nodes within outage area.

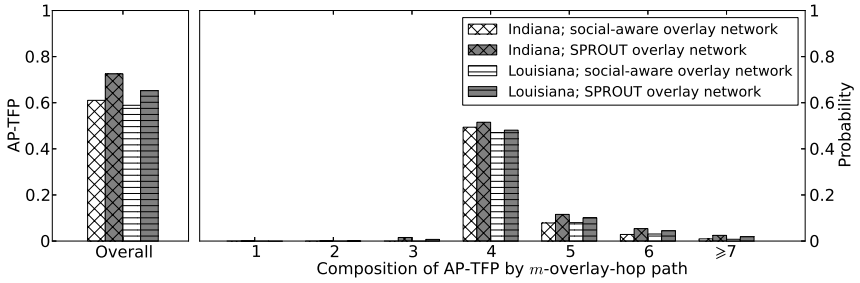
Since the SPROUT and social-aware overlays include social link information, nodes in these overlays are by definition able to reach their friends (*1-social-hop friend*). Figure 2.6 shows the *AP-TFP* on the SPROUT and social-aware overlays of the Foursquare network. In particular, Figure 2.6a, Figure 2.6b, and Figure 2.6c show the results of the *AP-TFP* for two nodes that before the cut are *2-social-hop, 3-social-hop* and *4-social-hop friends*, respectively. For each sub-figure in Figure 2.6, if two *n-social-hop friends* can be connected by the overlay hops, we also evaluate the composition of *overlay-hop Path* between them (shown on the right side of the figures). Since the average degree of Indiana is greater than the one of Louisiana (shown in Section 2.5.1), the *AP-TFP* in Indiana is greater.



(a) *2-social-hop friends*



(b) *3-social-hop friends*



(c) *4-social-hop friends*

Figure 2.6: *AP-TFP* in Foursquare network. Left box accumulates across all *m-overlay-hop paths* in the right box.

Figure 2.7 shows the *AP-TFP* on both SPROUT and social-aware overlays, under different partitions of the synthetic dataset. For each *n-social-hop friends* (n = 2, 3, 4), the *AP-TFP* increases as number of nodes in the partition increases. To explain this behavior, consider the case of *3-social-hop friends*: the *AP-TFP* of social-aware (SPROUT) overlay is 0.34 (0.35) at the partition with 0.1% overall nodes, and increases to 0.94 (0.98) at the partition with 2.5% overall nodes. This is because the average degree of the partition increases as the partition size increases (shown in Figure 2.4a).

In addition, we can observe from Figure 2.7, in a same partition, the *AP-TFP* of *shorter-social-hop friends* is greater than the one of *longer-social-hop friends*. This is because the social path between *n-social-hop friends* is

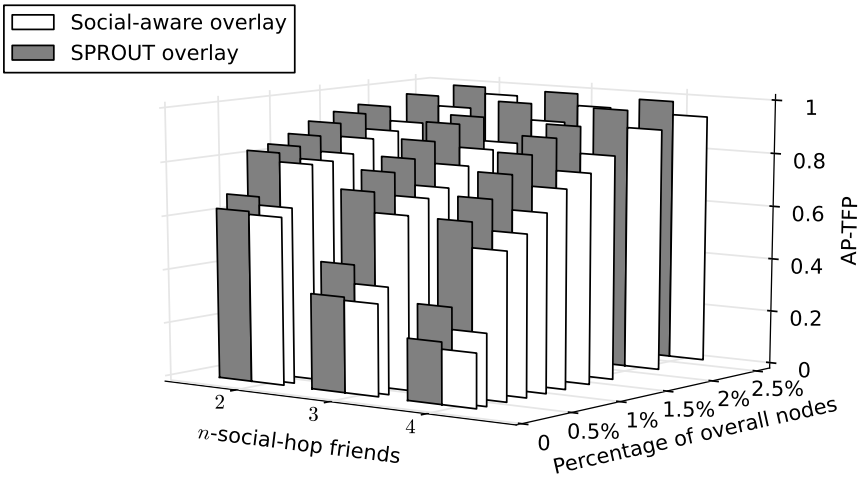


Figure 2.7: *AP-TFP* (synthetic dataset, 0%-2.5% nodes within outage area).

longer and more likely to be "cut" after the partition with larger n, since connecting nodes might be located outside the partition.

Figure 2.6 and Figure 2.7 show that the *AP-TFP* in SPROUT is slightly greater than the one in social-aware overlay network. This is because the set of edges in SPROUT is the union of social-aware and Chord overlay edges. In other words, in the SPROUT overlay, besides the social-aware links, *n-social-hop friends* also can be connected by the Chord overlay links. Therefore, by design, it holds that *AP-TFP*(social-aware) $\leqslant$ *AP-TFP*(SPROUT). However, as shown in the next subsection, the same is not necessarily true for the *AP-RFP*.

2.6.2 After Partition Routing-connected Friends Probability (AP-RFP)

Although in Chord overlay network the *AP-TFP* is greater than zero when the fraction of nodes in the partition is greater than 3%, the *AP-RFP* is no larger than 0.1%, as shown in Figure 2.5. The reason is that Chord uses identifier-based routing. When a partition breaks a small number of nodes from a larger initial overlay, a large fraction of the structured links that maintain the Chord overlay are severed, and routing is significantly compromised. Note that, with compromised routing, the ability of new nodes to bootstrap and connect to the overlay is also compromised.

A social-aware overlay with unstructured social routing has the potential to improve routing among *n-social-hop friends* within the partition. In the social-aware overlay network, we use *geographical routing* to forward mes-

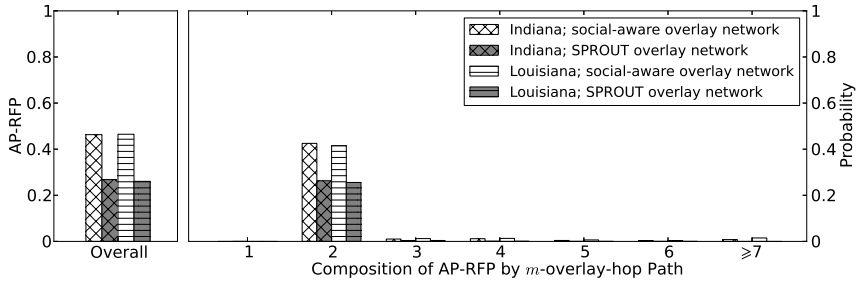
sages, where the source node forwards the message to a neighbor who is nearest to the target node.

Since social-aware and SPROUT overlays include the friendship from social network, *1-social-hop friends* can be reached in a straightforward manner. Figure 2.8 shows the *AP-RFP* on the social-aware and SPROUT overlay networks in the scenarios of both Indiana and Louisiana partitions (Foursquare dataset). Figure 2.8a, Figure 2.8b, and Figure 2.8c evaluate the *AP-RFP* for two nodes that before the cut are *2-social-hop, 3-social-hop* and *4-social-hop friends*, respectively. For each sub-figure in Figure 2.8, if two *n-social-hop friends* can be connected by the overlay routing, we also evaluate the composition of *overlay-hop Path* between them (shown on the right part of the figures). Figure 2.9 shows the *AP-RFP* in different partitions based on social-aware and SPROUT overlay networks (synthetic dataset).

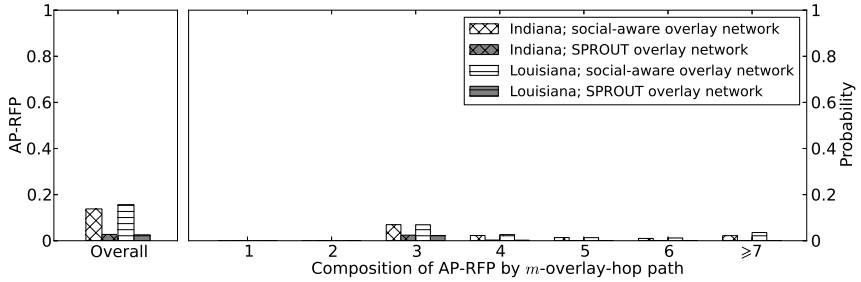
In a same partition, the *AP-RFP* of *shorter-social-hop friends* is higher than the one of *longer-social-hop friends* on both social-aware and SPROUT overlays. The reason is the same in the case of *AP-TFP*: a longer path between *n-social-hop friends* means there is a higher probability that an intermediate node might be blocked by the partition event.

From the topology viewpoint (*AP-TFP*), the nodes in the SPROUT are better connected than in the social-aware overlay network. However, *AP-RFP* is higher in social-aware compared to the SPROUT overlay network. This is because the social-aware overlay network uses geographical routing instead of structured routing. According to [15], if two nodes are close to each other, they are more likely to be friends. However, in SPROUT routing, a node routes the message to a neighbor that is 'Chordly' close to the target, i.e., its neighbor whose Chord ID is closest to the target node. As Chord ID is unrelated to geographical location, hence SPROUT routing is likely to need to route messages to a node ID that originally was outside the partition before the cut.

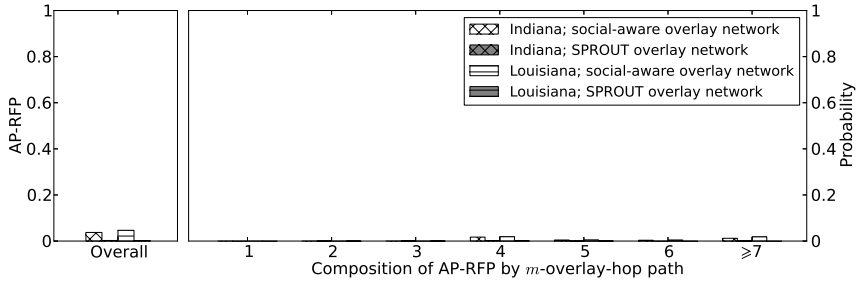
In our simulation, the average degree of the Foursquare network partitions (Indiana and Louisiana) is less than 4.1 (as shown in Section 2.5.1), while degree in synthetic network partitions is less than 2.1 (as shown in Figure 2.4). This is a consequence of the network sizes our experiments considered, which are relatively smaller than real social graphs. In practice, larger average degrees are expected within the partition. In this sense, the results are based on a conservative assumption: we expect in the real scenario that a partition can exhibit higher average degree. In addition, there are other routing protocols that might provide improved routability (as discussed in Section 2.7). As a result, in other practical scenarios *AP-RFP* could be higher than the results shown in Figure 2.8.



(a) *2-social-hop friends*



(b) *3-social-hop friends*



(c) *4-social-hop friends*

Figure 2.8: *AP-RFP* in Foursquare network. Left box accumulates across all *m-overlay-hop paths* in the right box. The maximum value of *AP-RFP* is its corresponding *AP-TFP* as shown in Figure 2.6.

2.7 Improvements

So far we have considered a baseline topology (friend-to-friend links) and routing algorithm (geographical routing) for the social-aware overlay for comparison with Chord and SPROUT. In this section, we discuss possible improvements to topology and routing of social-aware overlay, with respect to *AP-TFP* and *AP-RFP*.

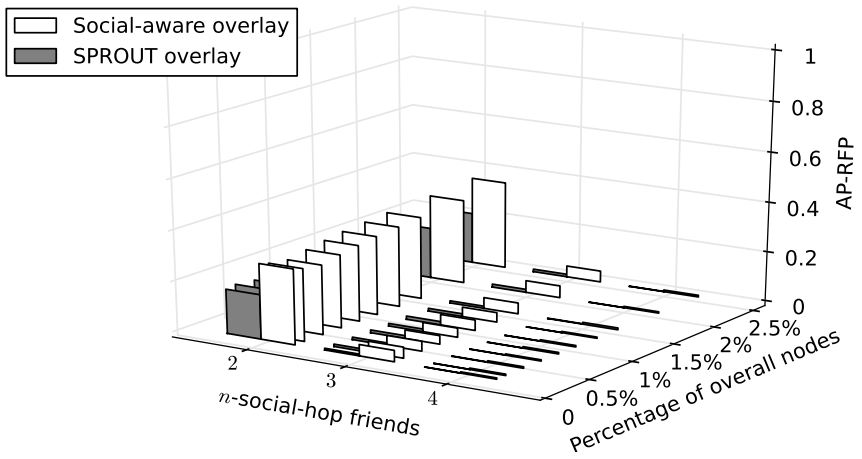
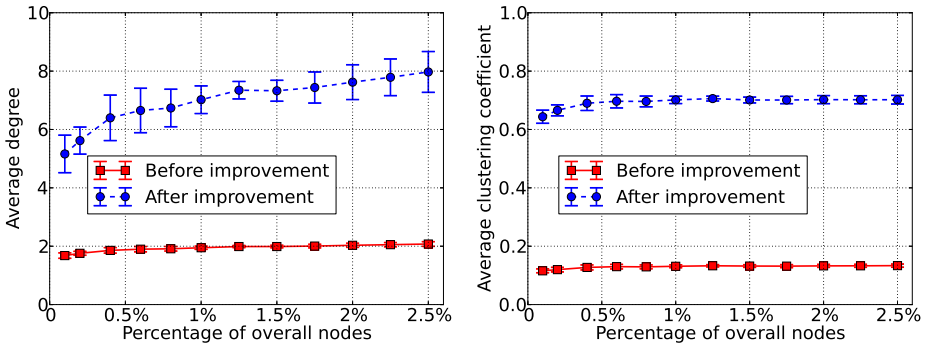


Figure 2.9: *AP-RFP* (synthetic dataset, 0%-2.5% nodes within outage area). The maximum value of *AP-RFP* is its corresponding *AP-TFP* as shown in Figure 2.7.

2.7.1 Adding friend-of-friend links

For social-aware overlay network, a possible improvement in the topology is for nodes to share friendship information with all their friends; i.e., the 2-*social-hop friends* can communicate with each other directly after sharing this information. This has the potential to increase routability, at the expense of a larger number of overlay links per node. Figure 2.10 shows the average degree and average clustering coefficient of the partition after the improvement, compared to the baseline. From Figure 2.10a and 2.10b, we observe that, the average degree and average clustering coefficient increase significantly when using such method.



(a) Degree (b) Clustering coefficient

Figure 2.10: Metrics of improved network in synthetic dataset.

In order to evaluate the improvement with respect to *AP-RFP*, we compare the routing performance on original overlay (before the improvement) with the improved overlay. In the case of Foursquare network (Indiana as the partition area), the *AP-RFP* of *3-social-hop friends* is 0.14 before the improvement, and increases to 0.6 after the improvement. In the case of synthetic network (the partition including 1% overall nodes), the *AP-RFP* of *3-social-hop friends* is 0.042 before the improvement, and increases to 0.27 after improvement.

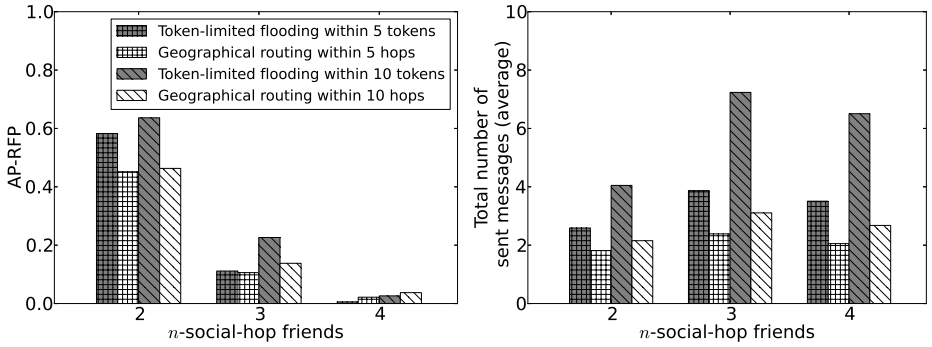
2.7.2 Comparison of Routing Algorithms in Social-aware Overlay Network

As mentioned in Section 2.1, in the social-aware overlay network, there are several routing algorithms that can be chosen. While we have focused on the geographical routing algorithm, it might not be the optimal approach. Here, we compare geographical routing with an alternative protocol—the limited-token flooding [65]. In the limited-token flooding, the token count indicates the total number of nodes to which a request may be forwarded. We use 5 and 10 as the number of tokens in our simulation, comparing with 5-hop and 10-hop in the geographical routing. This means a node might send less than such number of messages to discover its *n-hop social friend*. Note that, in geographical routing, a node will drop the message if there is no closer neighbor to the target node than itself. In limited-token flooding, a node will drop the message if it has no neighbor to forward the message.

Figure 2.11a shows the *AP-RFP* of *n-social-hop friends* (n = 2,3,4) in both geographical routing and limited-token flooding routing, while Figure 2.11b shows the average number of messages sent by all the nodes involved in a single route discovery process. Note that, the legend of Figure 2.11b is same as the one of Figure 2.11a. For *2-social-hop friends* and *3-social-hop friends*, limited-token flooding routing delivers improved routability, although it uses more messages to discover the target node. In the case of *4-social-hop friends*, 5 or 10 tokens is too small a number to route a 4-social-hop friend. Note that, 10 tokens is a somewhat conservative number, because token-limited flooding is only used in the process of discovering a path; we can increase this number to enhance the routability.

2.8 Summary

In this chapter, we evaluate the connection probability among social peers after network partition events. We perform the assessment considering Chord and SPROUT overlays, as well as the social-aware overlay proposed in this chapter. The results show that, for the partition area with less than 2.5% overall nodes, the Chord overlay is almost totally disconnected. In addition, for the *n-social-hop friends* ($2 \leqslant n$), although the *AP-TFP* on SPROUT



(a) *AP-RFP* (b) # of messages sent

Figure 2.11: Performances of routing algorithms on social-aware overlay; Foursquare dataset; Indiana as the outage area.

overlay is slightly higher than the one on social-aware overlay, the social-aware overlay achieves better *AP-RFP*. Moreover, we discuss possible improvements in topology and routing for the social-aware overlay and evaluate the impact on degree, clustering coefficient, and *AP-RFP*. Considering that the social-aware overlay network requires fewer links than SPROUT does (the former is a sub-graph of the latter), we contend that social-aware overlay network provides more routability while maintaining fewer links, comparing with the other considered overlays.

In this work, we have considered a static network (after the partition) and did not account for nodes constrained by NATs/firewalls, nor churn. In future, work we will consider the processes of nodes joining/leaving the network, and multiple devices per user. In particular, we will consider the challenges of bootstrapping nodes into the overlay after it has been partitioned, leveraging n-hop friends with public IP endpoints for reflection/forwarding (as in the STUN and TURN protocols).

Chapter 3

A Social Relation-based Bootstrapping Methods for Social-aware Overlay

The results presented in the previous chapter show that the social-aware overlay achieves better AP-RFP while maintaining fewer links than SPROUT overlay. In this chapter, we further demonstrate usability and feasibility of social-aware overlay in the network partition situation. In particular, we discuss how users bootstrap into the social-aware overlay even under the connectivity constraints imposed by firewalls and NATs.

The problem of bootstrapping in P2P overlay has largely been ignored [170]. Bootstrapping is an important functionality required by P2P overlay networks [34]. Nodes intending to participate in such an overlay network initially have to find at least one node that is already part of this network [84]. While structured P2P overlays based on distributed hash tables, define rules about how to bootstrap, unstructured P2P networks still need to use bootstrapping techniques until nodes are sufficiently connected [143]. Note that, we consider that the social-aware overlay is able to work in situations where a localized portion of network disconnects from the Internet. Therefore, a fully decentralized bootstrapping method need to be investigated [8]. The usual bootstrapping solutions, which typically require a centralized service [73], are not possible. In this chapter, we propose a bootstrapping method where overlay users are able to leverages social friends with public-IP address to joining the overlay. Afterwards, we analyze how a node can bootstrap into the social-aware overlay and reach its direct friends after a partition event, in a decentralized way.

3.1 Related Works

Structured P2P networks based on Distributed Hash Tables define rules about how to proceed the bootstrapping of nodes. In this section, we discuss the current P2P bootstrapping methods in unstructured overlays.

Research [70, 42] has shown that peers can use the locality properties of recent IP addresses to make intelligent guesses about other peers in the P2P system using an approach called random probing. In [51], authors proposed two mechanisms for increasing the performance of random address probing: (1) probing multiple ports per host and (2) hash-based filter-resistant port selection, making distributed bootstrapping feasible even from a practical point of view. Researchers in [115] presented a method of bootstrapping P2P overlay networks that involves multicasting P2P overlay join queries and responses, and hence nodes choose a bootstrapping node considers both the distance in hops. In [161], the authors survey the current weakness of P2P overlay bootstrapping, and propose a bootstrapping, considering NAT server and firewalls. Peer-to-peer solutions such as in [91, 5] use dedicated servers or overlay to assist nodes in bootstrapping. However, these solutions do not consider network partitions. Bootstrapping servers may be located outside the partition area, so that users who rely on such servers cannot join the overlay.

In our proposal, we leverage social friends with public address as Session Traversal Utilities for NAT (STUN) or Traversal Using Relays around NAT (TURN) servers to assist nodes to bootstrap into overlay.

3.2 Definitions and Considered Types of Nodes

In this section, we introduce the following definitions, in order to further analyze usability and feasibility of social-aware overlay, and to evaluate nodes bootstrapping situation under the overlay constraints, such as firewalls and NATs.

Definition 5. ***After Partition Topological Bootstrapping Ratio*** *(AP-TBR) refers to the likelihood that, after the partition, a node is able to bootstrap such that there are paths in the overlay topology to all of its* 1-social-hop *friends, under the constraints imposed by firewalls and NATs.*

Definition 6. ***After Partition Routing Bootstrapping Ratio*** *(AP-RBR) refers to the likelihood that, after the partition, a node is able to bootstrap such that there are routable paths in the overlay topology to all of its* 1-social-hop *friends, under the constraints imposed by firewalls and NATs.*

With our bootstrapping method, users leverage nodes with public IP addresses to join the social-aware overlay. Therefore, we consider two categories of nodes: nodes with private IP address (referred to as **private-IP nodes**) and nodes with public IP address (referred to as **public-IP nodes**). Furthermore, in order to be more realistic, within private-IP nodes, we take two sub-categories into consideration: nodes behind symmetric NAT (referred to as **symmetric-NAT nodes**) and nodes behind cone NAT (referred to as **cone-NAT nodes**).

3.3 Methodology

In order to assist private-IP nodes to bootstrap into the overlay, public-IP nodes may serve as STUN or TURN servers. STUN servers allow NATed nodes to connect directly in a peer-to-peer fashion, while TURN servers allow a relayed link through an intermediary. Note that, symmetric-NAT nodes always require an intermediary relay server and can only communicate with other private-IP nodes through a common public-IP TURN server node. In contrast, cone-NAT nodes can establish direct connection with other peers leveraging STUN servers. Certainly, before the communication, cone-NAT nodes need one or more connected STUN servers to discover each other. For all the public-IP nodes, although they still need to bootstrap into overlay, we assume they can reach all of their *1-social-hop friends* within outage areas as long as they are in their *m-social-hop friends*' boot-lists. Therefore, the *AP-TBR* and *AP-RBR* of public-IP nodes are always 1. In order to connect to STUN or TURN servers (public-IP nodes) and to discover other peers after a partition, each node maintains a list, created before the partition, named boot-list. Boot-list has q number of public nodes (boot-nodes) ($0 \leqslant q \leqslant n$), where n is a maximum number of boot-nodes in boot-list and controlled individually by overlay peers. Note that, boot-nodes in one's boot-list are one's nearest (shortest social-hops) public-IP nodes. After partition, according to the boot-list, each node in the outage area establishes links (boot-link) to its boot-nodes. For privacy concerns, boot-nodes may be limited to *m-social-hop friends*.

Figure 3.1 illustrates an example of bootstrapping scenario in our evaluation. Figure 3.1a shows an initial topology of social-aware overlay before partition. Note that, the links between nodes indicate social friendship. We assume that m is equal to 3 in Figure 3.1, which means that nodes can leverage public-IP nodes as boot-nodes as far as *3-social* hops. As a result, $c_1, c_2, c_3, c_4, c_5, c_7, s_1$ and s_2 have two public-IP nodes (i.e., p_1 and p_2) in their boot-lists; s_3 and c_6 have only one public-IP node (p_1) in their boot-list; and there are no public-IP nodes in the boot-lists of p_1 and p_2. Figure 3.1b shows the network after the partition, where all links (e.g., $\{s_1, c_7\}$,$\{s_1, c_2\}$,$\{p_1, s_2\}$

Table 3.1: Boot-lists of c_4, s_3 and c_6 before partition and after partition.

	Before Partition			After Partition		
	Nodes	**Hops**	**IP address**	**Nodes**	**Hops**	**IP address**
c_4	p_1	2	212.32.35.1	p_2	3	251.233.1.2
	p_2	3	251.233.1.2			
s_3	p_1	3	212.32.35.1	$\varnothing$		
c_6	p_1	3	212.32.35.1	$\varnothing$		

and $\{c_1, s_2\}$) between the outage area and the outside area are removed.[1] After partition, the boot-list is updated (by checking availability of public nodes) at each node. Table 3.1 shows boot-lists of a subset of nodes (c_4, s_3 and c_6), before partition and after partition, respectively. In Figure 3.1c, for nodes without any neighbor, we remove them (e.g., c_7) from the partition. This situation happens in the evaluation of Foursquare dataset. Foursquare is a tourist-orient application — people are more likely check in (record their location information) when they arrive at a new place (e.g., traveling resorts) or an interesting place (e.g., nearby restaurants). We do not consider these nodes in our evaluation. In Figure 3.1d, for each node, we add boot-links to its boot-nodes (e.g., $\{p_2, c_3\}$, $\{p_2, c_4\}$, $\{p_2, c_5\}$ and $\{p_2, s_2\}$) according to its boot-list. Given that symmetric-NAT nodes cannot connect to other private-IP nodes directly (they need a relay server), we remove links connecting symmetric-NAT nodes and other private-IP nodes (e. g., $\{c_4, s_3\}$) if no common public-IP nodes are in their boot-list, as shown in Figure 3.1e. Since p_2 could serve as an intermediary TURN server to forward messages among c_4, c_3 and s_2, we keep these links (e.g., $\{c_4, s_2\}$ and $\{c_3, s_2\}$). This is the topology of social-aware overlay that is Immediately After Partition, or *IAP* topology. However, over time, links may disappear. Notice, for instance, link $\{c_5, c_6\}$ that exists before the partition, cannot be re-established again. Over a period of time, the temporary links will be disconnected because their involved nodes (e.g., c_6) restart or links may fail (e.g., due to timeouts). Figure 3.1f shows the topology of social-aware overlay with no temporary links. We refer this topology as the *NTL* (No Temporary Links) topology. In this chapter, we evaluate both *AP-TBR* and *AP-RBR* in both *IAP* and *NTL* topologies, respectively. Note that, in the bootstrapping scenario, when a node is ready to bootstrap, we assume that this node has no links except boot-nodes (an example is shown in Figure 3.2).

To analyze the AP-TBR and AP-RBR, each simulation run is based on the same topology in the Foursquare dataset. In our evaluation, we perform 50 simulation runs and then average results. For each simulation, first, we obtain the outage area. For each node n_i in a partition area, after links

[1] To focus on analysis of outage area in the following, we use gray color for the outside area.

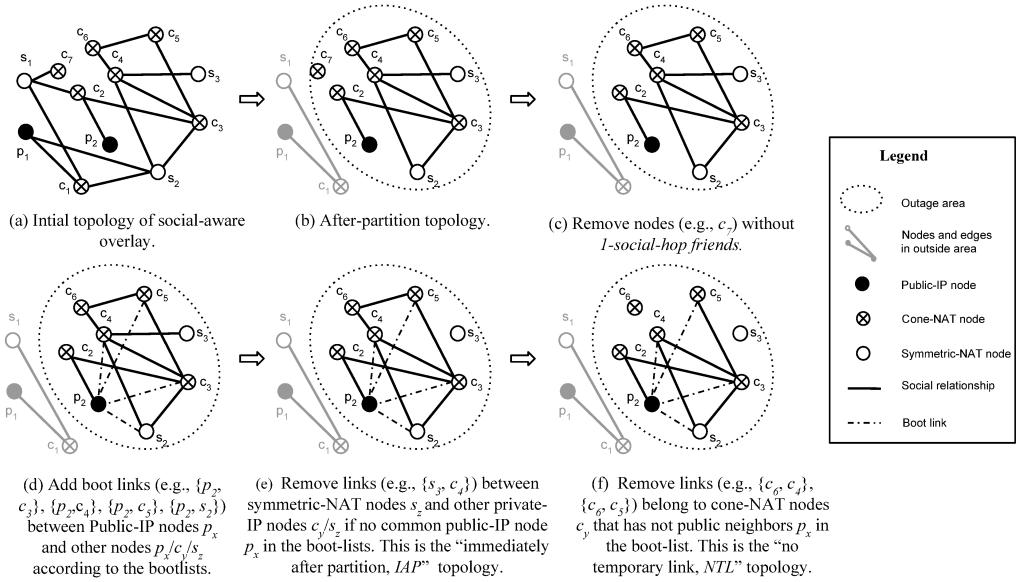


Figure 3.1: Different phases in dataset construction.

to boot-nodes are established (examples as shown in Figure 3.1e and 3.1f) and social links are removed (examples is shown in Figure 3.2), we count the number of n_i's *1-social-hop* friends that are topologically connected to n_i (topologically-connected neighbors, nt) as well as the number of n_i's *1-social-hop* friends that are reachable by n_i through the routing algorithm (routing-connected neighbors, nr). We evaluate n_i's *AP-TBR* (tbr) by:

$$tbr_{n_i} = \frac{nt_{n_i}}{n_{n_i}},$$

where nt_{n_i} is the number of n_i's topological connected neighbors, and n_{n_i} represents the total number of n_i's *1-social-hop* friends. In addition, n_i's *AP-RBR* (rbr) is represented by

$$rbr_{n_i} = \frac{nr_{n_i}}{n_{n_i}},$$

where nr_{n_i} is number of n_i's routing connected neighbors.

After averaging *AP-TBR* and *AP-RBR* of all nodes in a partition, we get the partition's *AP-TBR* as:

$$TBR = \frac{\sum_{i=1}^{x} tbr_{c_i} + \sum_{j=1}^{y} tbr_{s_j} + \sum_{k=1}^{z} tbr_{p_k}}{x+y+z},$$

and the partition's *AP-RBR*:

$$RBR = \frac{\sum_{i=1}^{x} rbr_{c_i} + \sum_{j=1}^{y} rbr_{s_j} + \sum_{k=1}^{z} rbr_{p_k}}{x+y+z},$$

where x, y and z represent the number of cone-NAT nodes (c_i), symmetric-NAT nodes (s_j), and public-IP nodes (p_k) in the partition area, respectively.

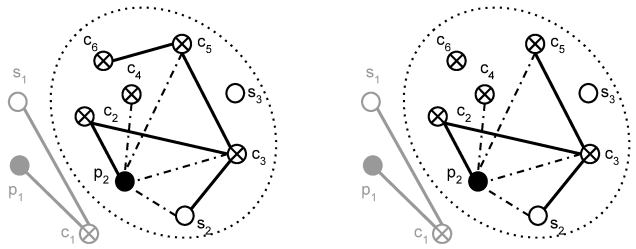
Figure 3.2 shows an example of how to calculate *AP-TBR* and *AP-RBR* of a node (e.g., c_4) in both *IAP* topology (as shown in Figure 3.2a) and *NTL* topology (as shown in Figure 3.2b), respectively. Before the bootstrapping evaluation, we remove links (if existed) between c_4 and c_4's *1-social-hop* friends with private IP address (c_4 can reach the public-IP friends anyway). We evaluate c_4's *AP-TBR* (*tbr*) by:

$$tbr_{c_4} = \frac{nt_{c_4}}{n_{c_4}},$$

where nt_{c_4} is number of c_4's topological connected neighbors, and n_{c_4} represents number of c_4's *1-social-hop* friends. Note that, c_4's *1-social-hop* friends are c_6, s_3, c_3 and s_2. c_4 is topological connected to c_6, c_3 and s_2 in *IAT* topology, while c_4 is topological connected to c_3, s_2 in *NTL* topology. Therefore, c_4's *AP-TBR* are 0.75 and 0.5 in *IAP* and *NTL* topologies, respectively, for the example graph in Figure 3.1. In addition, c_4's *AP-RBR* (*rbr*) is represented by

$$rbr_{c_4} = \frac{nr_{c_4}}{n_{c_4}},$$

where nr_{c_4} is number of c_4's routing connected neighbors.



(a) *IAT* topology (b) *NTL* topology

Figure 3.2: Example of evaluation of the *AP-TBR* and *AP-RBR* of c_4 in *IAT* topology and *NTL* topology.

Considering node c_4 in Figure 3.2a with its boot-link $\{c_4, p_2\}$, it can request a message to be overlay-routed to node c_6 via c_5. Using p_2 or another public node as a STUN server, the cone-NAT nodes c_4 and c_6 can then bootstrap a link. However, in Figure 3.2b because the link $\{c_5, c_6\}$ has disappeared (e.g., due to timeout), c_4 is not able to bootstrap a link to c_6. As a result, IAP topology provides better topological connectivity.

3.4 Evaluation

In our evaluation, we consider a set of baseline parameters, then perform a sensitivity analysis by varying each parameter. We use limited-token flood-

ing routing algorithm to discover peers, since this routing algorithm has higher *AP-RFP*. The baseline of token number is 50. Note that, we only vary this token number in routing bootstrapping analysis, given that routing algorithm is not related to topological analysis. According to [60], and considering increasing prevalence of IPv6, the baseline of the public-IP nodes portion accounts for 20% (out of overall nodes). The baseline of cone-NAT nodes proportion is 92% (out of all private-IP nodes) according to [72]. For nodes in our baseline evaluation, the length of their boot-list is 50, and the maximum hops for selecting boot-nodes is $m = 5$. However, in practice, a user could tune these numbers in order to limit resource utilization (by decreasing these numbers), or achieve higher connectivity (by increasing these numbers). Table 3.2 shows the baseline parameters we considered in our evaluation.

Parameters	**Value**
Tokens in limited-token flooding	50
Proportion of Public-IP nodes	20%
Proportion of cone-NAT nodes	92%
Total items in boot-list	50
Maximum hops of boot-nodes in boot-list	5

Table 3.2: Baseline parameters.

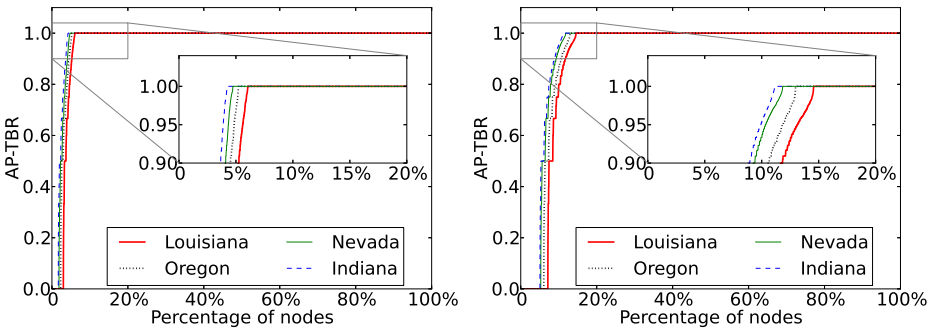
We assess our bootstrapping method on the Foursquare dataset described in previous chapter (Section 2.4.1). In particular, we first evaluate *AP-TBR* and *AP-RBR*, respectively. Then we tune the baseline network parameters to perform sensitivity analyses.

3.4.1 Evaluation of AP-TBR / AP-RBR

In this subsection, we evaluate the *AP-TBR* and *AP-RBR* based on baseline parameters, in Foursquare dataset, in both *IAP* topology and *NTL* topology, respectively. In the bootstrapping analysis, besides the Louisiana and Indiana, we also consider the Oregon and Nevada as the partition area. Therefore, we consider four outage areas (i.e., Louisiana 0.6%, Oregon 0.8%, Nevada 0.9% and Indiana 1.2%) in our evaluation. Note that, the percentage behind each outage area indicates portion of nodes in such outage area.

Figure 3.3 shows Cumulative Distribution Function (CDF) of *AP-TBR* based on baseline parameters, while Figure 3.4 shows CDF of *AP-RBR* based on baseline parameters. In particular, Figures 3.3a and 3.4a show the bootstrapping ratio distribution on *IAP* topology, while Figures 3.3b and 3.4b show the bootstrapping ratio distribution on *NTL* topology. An observation from Figures 3.3 and 3.4 is that, *IAP* topology has more nodes with 100% *AP-TBR* / *AP-RBR* and less nodes with 0% *AP-TBR* / *AP-*

RBR than *NTL* topology. For example, as shown in Figure 3.3a, in Indiana partition, around 2% of nodes cannot bootstrap into overlay (their *AP-TBR* is 0); around 96% of nodes can bootstrap into overlay and communicate with all of their neighbors (their *AP-TBR* is 100%). In Figure 3.3b, in Indiana partition, around 5% of nodes' *AP-TBR* is 0; around 90% of nodes' *AP-TBR* is 100%. This is because *IAP* topology provides better connectivity than *NTL* topology as we discussed in Section 3.3. Another observation from Figures 3.3 and 3.4 is that, for a same partition area, *AP-TBR* is always larger than *AP-RBR*. In other words, there are more nodes with 100% bootstrapping ratio and less nodes with 0% bootstrapping ratio in *AP-TBR* than *AP-RBR*. Note that, the maximum value of AP-RBR shown in Figure 3.4 is its corresponding AP-TBR as shown in Figure 3.3.

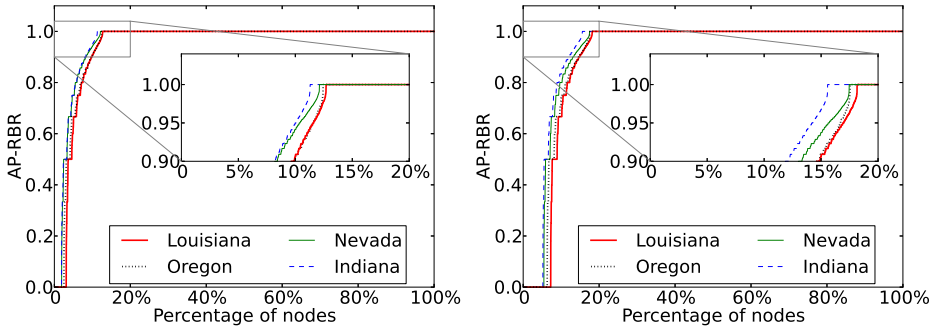


(a) *IAP* topology. *AP-TBR* in Louisiana, Nevada, Oregon and Indiana are 96.2%, 96.9%, 97.3% and 97.6%, respectively.

(b) *NTL* topology. *AP-TBR* in Louisiana, Nevada, Oregon and Indiana are 91.9%, 92.1%, 92.5% and 93.6%, respectively.

Figure 3.3: Distribution of *AP-TBR*; baseline parameters; the y-axis indicates *AP-TBR* as a function of percentage of nodes (shown on x-axis).

Figure 3.5 shows the distribution of *AP-TBR* among nodes with different degrees in Indiana partition, while Figure 3.6 shows the distribution of *AP-RBR* among nodes with different degrees in Indiana partition. Figure 3.5 shows a trend where nodes with larger degree achieve higher *AP-TBR*. This is because the large degree of a node indicates large number of 1-*social-hop friends* neighbors connected to this node. Therefore, boot-nodes (in boot-list) have more probability to be socially close (short hops) to the nodes with large degree. In this sense, these boot-nodes have higher probability to be located near the nodes with large degree. As a result, after partitions, the nodes with large degree could reach more boot-nodes. Other distributions of *AP-TBR* in Nevada, Oregon and Louisiana have shown the same trend, with the maximum difference of 2.1% at *IAP* topology and 3.8% at *NTL* topology. Figure 3.6 shows an opposite trend compared to Figure 3.5. Figure 3.6 shows a trend where nodes with larger degree have lower *AP-RBR*. Note that, we



(a) *IAP* topology. *AP-RBR* in Louisiana, Nevada, Oregon and Indiana are 94.8%, 95.1%, 95.7% and 96.1%, respectively.

(b) *NTL* topology. *AP-RBR* in Louisiana, Nevada, Oregon and Indiana are 90.8%, 90.9%, 91.3% and 92.7%, respectively.

Figure 3.4: Distribution of *AP-RBR*; baseline parameters; the y-axis indicates *AP-RBR* as a function of percentage of nodes (shown on x-axis).

use limited-token flooding as the routing algorithm in our evaluation, and the token is 50. Nodes with large degree need larger tokens to discover their neighbors. Other distributions of *AP-TBR* in partition Nevada, Oregon and Louisiana has shown the same trend, with the maximum difference 2.2% at *IAP* topology and 3.2% at *NTL* topology.

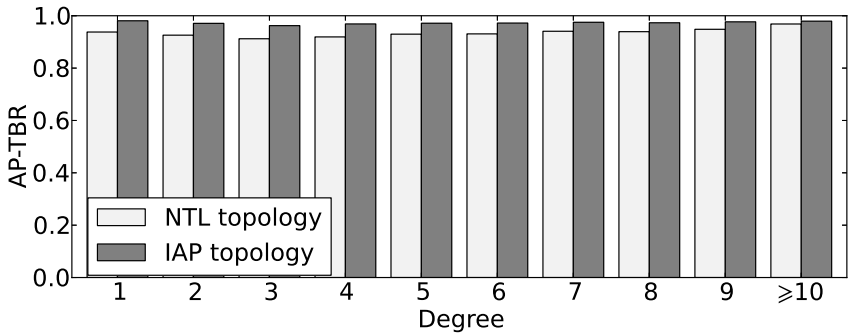


Figure 3.5: *AP-TBR* in different degree; baseline parameters; Indiana is the partition area.

3.4.2 Sensitivity analyses

By varying baseline network parameters, Figures 3.7, 3.8 and 3.9 shows the changes of *AP-TBR*, while Figures 3.10, 3.11, 3.12 and 3.13 shows the changes of *AP-RBR*. In particular, Figures 3.7, 3.8 and 3.9 show the different values of proportion of Public-IP nodes, proportion of cone-NAT nodes and length of boot-list, in four partition areas, and in both *IAP* and *NTL*

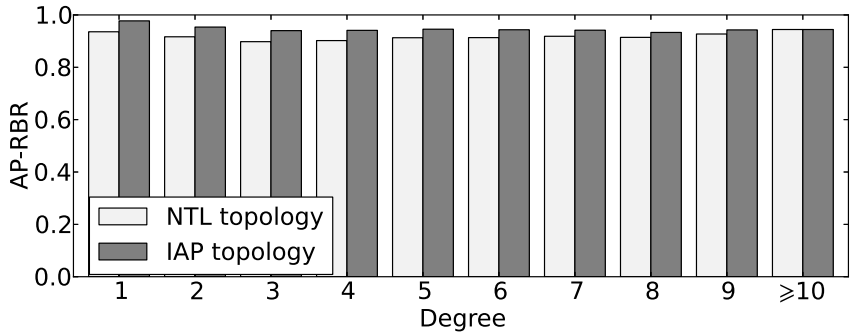


Figure 3.6: *AP-RBR* in different degree; baseline parameters; Indiana is the partition area.

topologies, respectively. Figures 3.10, 3.11, 3.12 and 3.13 show the different values of proportion of Public-IP nodes, proportion of cone-NAT nodes, length of boot-list and token in four partition areas, and in both *IAP* and *NTL* topologies, respectively. Note that, in *AP-TBR* analysis, we do not consider tokens of routing algorithm.

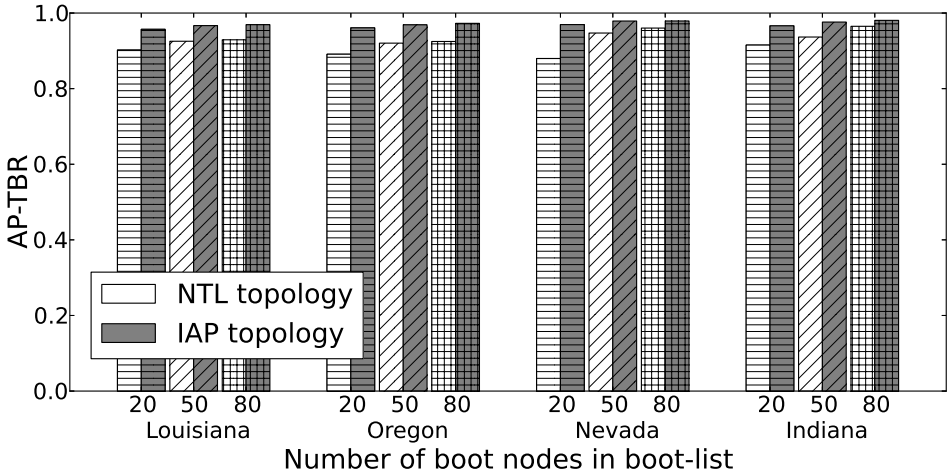


Figure 3.7: *Length of boot-list. AP-TBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

As shown in Figures 3.7 and 3.10, due to the increases of length of boot-list, the *AP-TBR* and *AP-RBR* increase. Increasing the length of boot-list indicates that nodes could choose larger number of boot-nodes, which means it is more likely that more boot-nodes are within the outage area after partition event. Figures 3.8 and 3.11 show the *AP-TBR* and *AP-RBR* increase when the proportion of public-IP nodes increases. This

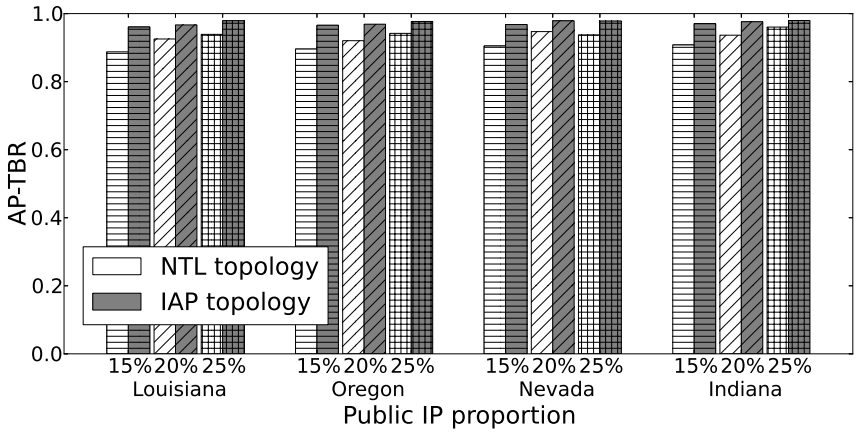


Figure 3.8: *Proportion of public-IP. AP-TBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

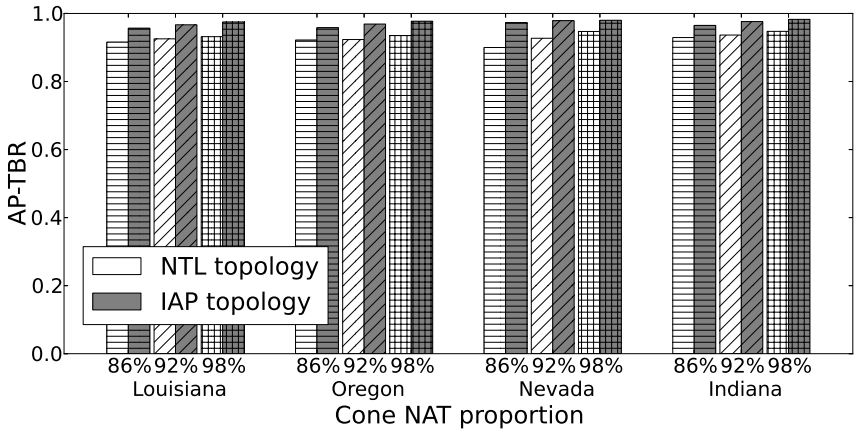


Figure 3.9: *Proportion of cone-NAT. AP-TBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

is because there are more public-IP nodes available to be chosen by other nodes. In this sense, the boot-nodes in a boot-list of a node are more likely near to that node geographically. Therefore, after partition, it is more likely that more boot-nodes are within outage area. In addition, as we discussed in Section 3.3, public-IP nodes can always be discovered by other peers (public-IP node's bootstrapping ratio is always 1). As a result, more public-IP nodes in a network indicates higher bootstrapping ratio. As shown in Figures 3.9 and 3.12, due to the increases of proportion of cone-NAT nodes, the *AP-TBR* and *AP-RBR* increase. This is because cone-NAT nodes do not need a relay server to communicate with other cone-NAT nodes. These nodes have fewer

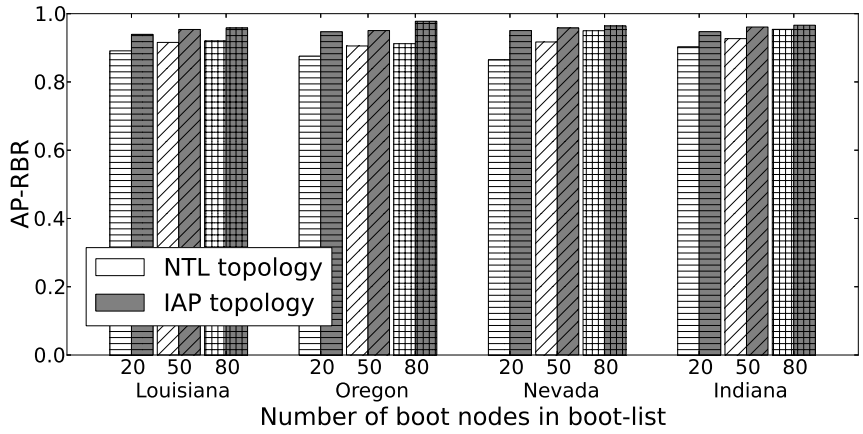


Figure 3.10: *Length of boot-list. AP-RBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

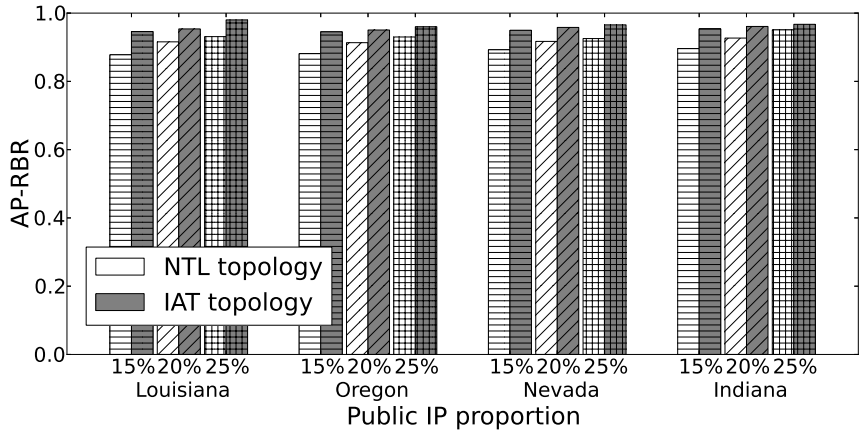


Figure 3.11: *Proportion of public-IP. AP-RBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

connectivity constraints. Figure 3.13 shows the *AP-RBR* increases while the token increases. Increasing token indicates more delivered nodes could be reached in the routing discovery process. Therefore, increasing token could discover more nodes in the limited-token routing algorithm. Notice, the maximum values of *AP-RBR* shown in Figure 3.10, 3.11, and 3.12, are their corresponding *AP-TBR* as shown in Figure 3.7, 3.8, 3.9.

Another observation from Figures 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 and 3.13, is that the *AP-TBR* and *AP-RBR* of four areas in increasing order is: Louisiana, Oregon, Nevada, Indiana. Note that, the number of nodes of these four areas is also in increasing order. Since more boot-nodes are within

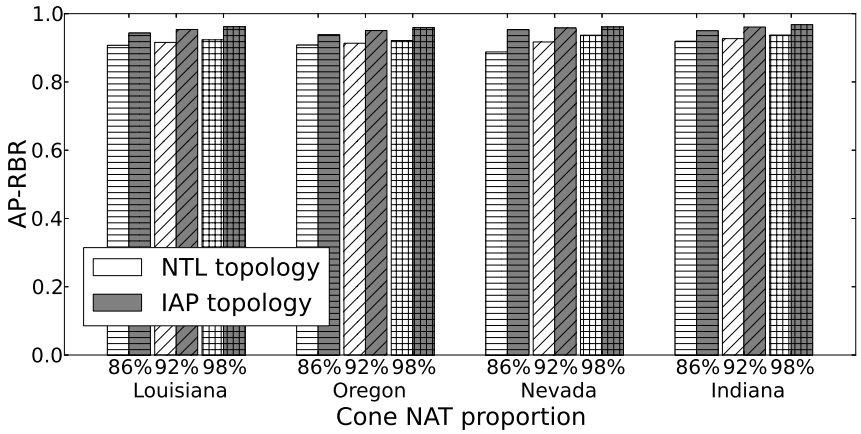


Figure 3.12: *Proportion of cone-NAT. AP-RBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

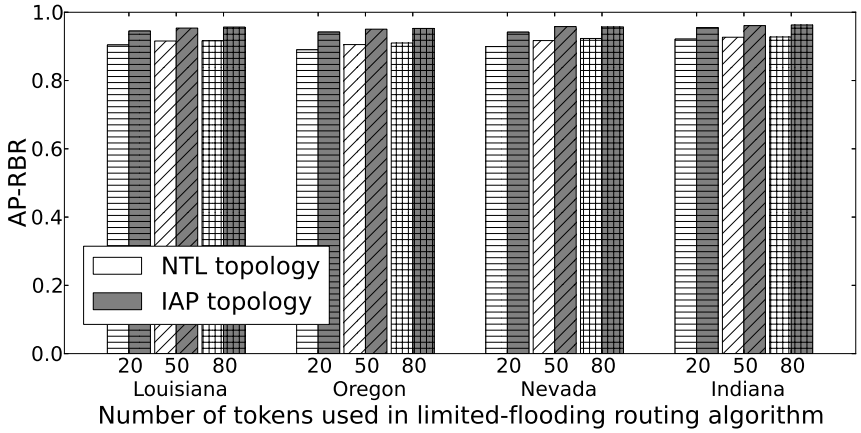


Figure 3.13: *Tokens in limited-token flooding. AP-RBR* in different partition, in different parameters, and in both *IAP* and *NTL* topologies. Louisiana, Oregon, Nevada and Indiana are partition area.

the outage area when number of nodes in partition area is large, the larger size of partition area also achieves higher *AP-TBR*.

3.5 Discussion

We point out that there are incentives in using social-aware overlay, even before the network outage: the social-based overlay network is useful in normal circumstances, providing a motivation for users to bootstrap into and maintain connections over time, prior to a partition event. For instance, the

open-source SocialVPN [91] and commercial P2P VPNs such as Hamachi [3] demonstrate that these systems have usefulness in personal computer environments, by establishing end-to-end secure virtual networks among users to allow them to share data, play games, and access remote desktops privately among peers. In a mobile computing environment, the incentive for users to maintain their overlay VPN connectivity is that it allows devices to serve data (e.g., photos, streaming media), query data, and receive notifications (e.g., private updates) from social peers. Furthermore, in addition to overlay links to devices of social peers, an unstructured overlay can accommodate ad-hoc links for connections established through short-lived relationships, e.g., based on geographical proximity. For instance, in a recent event that happened in Hong Kong, over 100,000 people used Firechat to communicate over ad-hoc links [1]; these transient links may be created after a partition happens, in contrast to the more persistent links that social peers can establish and maintain prior to partition.

3.6 Summary

In this chapter, we assessed the topological connectivity and message routability among social peers after network partition events. We performed an evaluation considering Chord and SPROUT overlays, as well as the social-aware overlay proposed in this chapter. The results show that, for the partition area with less than 2.5% overall nodes, the Chord overlay is almost totally disconnected. In addition, for *n-social-hop friends* ($2 \leqslant n$), although the *AP-TFP* on SPROUT overlay is slightly higher than the one on social-aware overlay, the social-aware overlay achieves better *AP-RFP*. Moreover, we also discussed another routing algorithm (i.e., limited-token flooding) that provides better *AP-RFP* for the social-aware overlay.

In addition, we further demonstrated usability and feasibility of social-aware overlay in a partition event. In particular, we analyzed how a node can bootstrap into the social-aware overlay and reach its direct friends. Our results show that, the social-aware overlay achieves more than 96% and 91% *AP-TBR* in different partitions, in *IAP* and *NTL* topology respectively, while achieve more than 94% and 90% *AP-RBR* in different partitions, in *IAP* and *NTL* topology respectively.

Chapter 4

SAND: a Social-Aware, Network-Failure Resilient, and Decentralized Microblogging System

Microblogging services (such as Twitter, Weibo) have emerged as popular communication mechanisms in recent years [35, 9]. For example, Twitter — the most successful microblogging service [86, 25] — has attracted more than 316 million monthly active users as of Q2 2015[1]. Nowadays, microblogging services play important roles in many aspects of our social life, such as obtaining interesting news, and keeping in touch with friends [169, 99]. However, the current popular microblogging services are based on a client-server architecture, and are vulnerable to central server failure and/or censorship [63, 85]. In recent years, network failures happened where a localized portion of a network is partitioned from the rest of the Internet, due to natural hazards [44], mis-configurations [122], Distributed Denial of Service (DDoS) attacks on links [92], and national censorships [45]. For example, on the evening of January 27, 2011, Egypt – a country with a population of 80 million, including 23 million Internet users – was partitioned from the Internet. The connectivity of the network inside the country was recovered after five days [48]. The people in Egypt were not able to use Twitter service because that Twitter servers are located outside of Egypt. Moreover, owing to the news media property [98] of microblogging services, under these

[1] `http://techcrunch.com/2015/07/28/twitter-monthly-active-users-crawl-to-316m/`

extremely situations (e.g., censorships, and natural hazards) that result in network partitions, people are more willing to rely on this service, in order to spread information [109, 37, 140]. Therefore, researchers are called to design microblogging systems that are resilient to network failures [160]. An intuitive solution to central server failures is to leverage decentralized Peer-to-Peer (P2P) networks [38, 83]. In the past few years, several decentralized microblogging solutions have been proposed [81, 130, 26, 66]. However, these solutions have the following issues: (i) they were not designed for (and hence cannot handle) scenarios where massive correlated failures occur; or (ii) their delivery rates were not significantly high; or (iii) they did not evaluated their working mechanisms on partitioned networks based on a real dataset, and with a real publisher-subscriber distribution.

We point out the following challenges to build a decentralized microblogging system:

- The system should support users with large numbers of subscribers (e.g., news media, celebrities), as well as users with few subscribers (e.g., a typical Internet user). For example, *katyperry* (a profile for the singer Katy Perry) in Twitter has over 76 million subscribers while the average number of subscribers in Twitter is only 208[2]. The design should be able to support users with different social patterns.

- The system should be able to work in situations where massive (e.g., 98%) nodes fail. Authors in [50] have already demonstrated that the routability of approaches based on Distributed Hash Table (DHT) (e.g., Chord[142]) is severely hampered in a massive nodes failure scenario. Therefore, the design should be able to support node bootstrapping and message routing without relying on DHT, in partitioned networks.

- End-to-end communication should be trusted, so as to avoid spam messages or misbehaviors from malicious users (e.g., a user with fake profile).

- The system should be able to achieve a desirable delivery rate. Publishers should find a way to deliver their messages to their subscribers.

- The system should support offline nodes that are inside a partitioned network. Since nodes inside the partitioned network may leave and rejoin the network (e.g., devices restarted), or be censored by authority, the system needs to tolerate offline nodes even inside partitions.

- The system should allow the communication between users even under the constraints imposed by firewalls or Network Address Translation (NAT) [55].

[2] `http://www.jeffbullas.com/`

In this chapter, we propose SAND, a social-aware, network failure-resilience, and decentralized microblogging system. SAND is thought as a possible alternative solution to centralized microblogging services (e.g., Twitter), and is able to provide the primary functions even in the situation of centralized microblogging services unavailable (i.e., network partitionsa). SAND takes advantages of an available social P2P virtual private networking (SocialVPN [91]). By utilizing SocialVPN, SAND is built upon an overlay where users have private IP connections to their social friends. Consequently, SAND enables trusted social communication on partitioned networks (even through NATs and firewalls). It makes more difficult to censor, invade and disrupt, which consequently creates a network-failure resilient microblogging system. Since the social relationship is unstructured, SAND is resilient to network partitions, as well as offline nodes inside the partition area, without relying on DHT. In SAND, *publishers* disseminate messages to their *subscribers* through one or more *forwarders*, with acceptable network overheads. Messages are stored in forwarders in order to ensure the delivery rate even when some of nodes in partitioned areas are offline. In order to study tradeoffs among different properties and delivery rates, we propose three variants of SAND: SAND based on Biased Random Walk algorithm (SAND-BR), SAND based on Popularity Flooding algorithm (SAND-PF), and SAND based on Super Node algorithm (SAND-SN). In order to avoid overheads on a single node with substantial number of subscribers, the subscribers cannot establish direct links to their publishers, as long as they are friends to each other. Note that, the network costs of uploading, downloading and storing messages are important. These costs are modeled in our evaluation by considering the number of sent messages, number of received messages, and number of stored copies of messages, respectively.

Figure 4.1 gives a birds-eye view of the basic concept of SAND. In Figure 4.1, the physical network level is at the bottom. It illustrates a physical distribution of endpoint devices and the links among them. Moreover, due to network failures, the devices inside the partitioned area cannot access the centralized server of a microblogging service. In the top part of Figure 4.1, we represent the social relations among people who own the devices inside the partitioned area. People are able to communicate with their friends directly by leveraging SocialVPN. In SAND, instead of sending updates to central servers, a publisher sends messages to his or her subscribers in a P2P fashion. Note that, Alice is a publisher; Bob and Charlie are subscribers of Alice; other intermediary nodes may act as forwarders to help Alice to deliver messages.

Contribution: In this chapter, we propose SAND, a social-Aware, network failure-resilient, and decentralized microblogging system. To the best of our knowledge, SAND is the first to provide a high delivery rate (e.g., 100% in SAND-SN), with acceptable network overheads, in situations where users cannot access to the central servers of microblogging services. More-

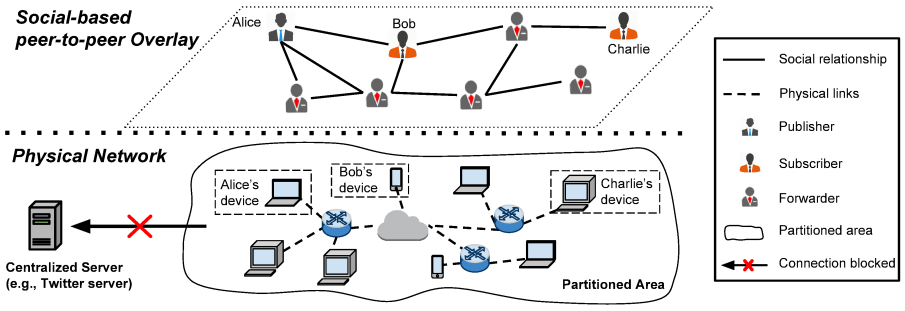


Figure 4.1: Overview of SAND's message dissemination mechanism based on social-based P2P overlay. Overlay links indicate social relationship. For example, since Alice and Bob are friends, they can communicate with each other in a P2P fashion through the SocialVPN overlay. Alice is a publisher, and Bob and Charlie are subscribers of Alice. After a network partition, without central servers, Alice disseminates messages to her subscribers (i.e., Bob and Charlie) through one or more forwarders.

over, we are the first to evaluate the performance of proposed decentralized microblogging system (i.e., SAND) on a ground-truth dataset with real social relationship, as well as real publisher-subscriber distribution.

4.1 Related Work

In this section, we discuss the following topics in the state of the art: network partitioning, decentralized online social networks, structured decentralized microblogging systems, unstructured decentralized microblogging systems, and P2P virtual private networks.

4.1.1 Network partitioning

Recently, researchers proposed several methodologies to analyze the situation of blocked Internet access, caused by country-wide censorships or natural disasters. In [45], Dainotti et al. utilized three types of data (Border Gateway Protocol updates, active trace route probing, and Internet background radiation traffic data) to analyze events that happened in two national Internet censorship episodes: in Libya and Egypt, in early 2011. The authors in [44] leveraged the malware pollution to analyze outage of Internet during two natural disasters — the recent earthquakes in New Zealand and Japan. These related works analyze the changes of network flow during the Internet outage. However, they did not propose a method to recover the connectivity at the application layer (e.g., for a microblogging application). In contrast, SAND focuses on the messages delivery at the application layer,

inside the partitioned network. In order to provide people (inside partition area) access to blocked servers, authors in [121] proposed Pisces, a decentralized protocol for anonymous communications that leverages users' social links to build circuits for onion routing [52]. In addition, a user discovers peers by using random walks [94] in the social network graph. However, unlike SAND, Pisces does not consider message dissemination in a situation where large numbers of nodes fail (e.g., a partition).

4.1.2 Decentralized online social networking

Authors in [40] introduced the concept of Virtual Private Social Networks (VPSNs), and proposed FaceVPSN, a decentralized system which is able to enforce privacy protection in Facebook. FaceVPSN mitigates the problem of lack of privacy in Facebook and hides information from users outside the VPSN. In [28, 43], authors proposed Distributed Online Social Networking (DOSN) Systems designed to provide OSN capabilities (e.g., the ability to post news, follow friends) in a decentralized way. PeerSoN [28] is a fully decentralized online social network. Peers use a DHT as a lookup service to locate each other's endpoints, then exchange encrypted information directly with each other. Safebook [43], based on DHT, focuses on anonymity by adding an additional layer of indirection to message requests by routing them through multiple layers of friends and friends of friends. However, the links in these networks are friend-to-friend links. They do not consider forwarding messages from a publisher to subscribers who are multiple-social-hops away from the publisher. Differently from the work in the state of the art, SAND focuses on disseminating messages from publishers to subscribers who might be not direct friends of each others.

4.1.3 Structured decentralized microblogging

In [131, 164, 69], authors proposed structured distributed microblogging systems. In particular, in Megaphone [131], subscribers join a multicast tree by using Pastry, a structured overlay based on DHT. In Cuckoo [164], peers discover each other's endpoints and send follow requests directly to each other through DHT. Publishers are then able to send updates directly to a subset of subscribers, depending on bandwidth availability, while the remaining subscribers use a gossip protocol to propagate updates among themselves. To handle churn, the Cuckoo approach relies on a centralized backend that stores all updates and follows requests from all users in case a publisher is not online. Twister [69] is comprised of three independent overlay networks: a Bitcoin protocol based overlay provides distributed user registration and authentication. DHT-based overlay provides key/value storage for user resources and tracker location for the third network. Bittorrent-based overlay is a collection of possibly disjoint "swarms" of subscribers,

which can be used for efficient near-instant notification delivery to many users. However, Cuckoo [164] does not mention replication mechanism to ensure content availability considering offline users. Moreover, these previous works [28, 43, 131, 164, 69] are based on DHT, which has already been demonstrated in [50] that routability of approaches based on Distributed Hash Table (DHT) (e.g., Chord[142]) is severely hampered in a massive nodes failure scenario. In SAND, we leverage unstructured social relationship to disseminate messages from publishers to subscribers.

4.1.4 Unstructured decentralized microblogging

FETHR [135] aims at a fully decentralized HTTP-based microblogging service. Peers subscribe to each other by exchanging canonical URLs and use the HTTP GET and POST methods to pull or push updates to each other. FETHR uses gossip-based dissemination for users with a high number of subscribers. However, depending on URLs to push and pull messages is problematic when the DNS severs are not available. In addition, FETHR do not consider the networking constrains imposed by NATs or firewalls. Moreover, a partition event may cripple the relationship among subscribers, so that the message dissemination is hampered. In SAND-SN, instead of communicating by URLs, publishers and their subscribers communicate with intermediary forwarders by public IP address directly. MoP-2-MoP [137] is a decentralized privacy-preserving microblogging infrastructure based on a distributed peer-to-peer network of mobile users. User exchange messages based on local point-to-point communication links over a delay-tolerant opportunistic network. However, in order to disseminate messages to distant subscribers, publishers in this approach, need a central server that could be easily censored by authorities. Firechat[3] is a message dissemination application which has been used in network partition event[4]. It delivers messages only according to the proximity among users. In SAND, publishers are able to send their messages precisely to their subscribers. Litter [141] is a P2P-based microblogging system that leverages the private IP connectivity of social-based VPN. It utilizes both IP multi-casting and Random Walks to ensure that publishers are able to delivery messages with varying degree of scope (i.e., friends, friends of friends, and/or the public). Considering the limited delivery rate of Random Walk, we propose SAND that provides improved message dissemination mechanisms. In addition, Litter evaluates its performance on a whole graph instead of a partitioned area. Therefore, the evaluation cannot reflect the performance during partition events. SAND is a alternative solution for current microblogging systems (e.g., Twitter, Weibo) and provides the primary functions in a situation where central servers fail (a portion of network partitioned from the Internet). In SAND,

[3] `http://opengarden.com/firechat/`
[4] `http://www.bbc.com/news/blogs-trending-29411159/`

System	Type	Social Aware	No DHT	Network Resilient	Delivery rate**
PeerSoN	Online social network	✗	✗	✗	0%
Safebook	Online social network	✓	✗	✗	0%
Megaphone	Microblogging	✗	✗	✗	0%
Cuckoo	Microblogging	✓	✗	✗	0%
FETHR	Microblogging	✗	✓	✗	*n/a*
Litter	Microblogging	✓	✓	✓	84.9%*
FireChat	Message dissemination	✗	✓	✓	*n/a*
SAND-BR	**Microblogging**	✓	✓	✓	**87.8%***
SAND-PF	**Microblogging**	✓	✓	✓	**86.0%***
SAND-SN	**Microblogging**	✓	✓	✓	**100%***

Table 4.1: Comparison of SAND with the related distributed systems (*results are gotten based on the evaluation setup shown in Section 4.4; **delivery rate is evaluated in 2% partition).

we perform an extensive simulation-based analysis to assess the delivery rate, as well as network overheads (i.e., transferred messages, stored copies of messages), in partitioned areas. Moreover, the evaluation in Litter is based on a synthetic dataset with friend-to-friend relationship, while does not consider a ground-truth publisher-subscriber distribution. In SAND, our simulation is based on a dataset of real data crawled from Twitter, that includes the ground-truth publisher-subscriber distribution. We summarize the related decentralized system in comparison with SAND in Table 4.1.

4.1.5 Peer-to-peer virtual private network (P2PVPNs)

Recently, P2P virtual private networks (P2PVPNs) such as Hamachi[5], SocialVPN [91] have become popular decentralized alternatives to centralized VPNs. However, in Hamachi, centralized Simple Traversal of UDP through NATs (STUN)-like servers are used to enable NAT traversal and establish direct P2P connections among users. In SocialVPN, NAT traversal is not centralized because it uses existing nodes in the overlay to perform UDP hole punching for direct P2P connectivity. In addition, SocialVPN leverages existing social infrastructures to enable VPNs. SAND is built upon an SocialVPN overlay where users have private IP connections to their social friends, which consequently enables trusted social communication on a partitioned network (even through NATs and firewalls). As shown in Figure 4.1, in SAND, friends are able to communicate directly.

[5] `https://secure.logmein.com/products/hamachi/`

4.2 Our Solution for Microblogging: SAND

In this section, we present SAND — a social-Aware, network failure-resilient, decentralized microblogging system. For ease of exposition, we first introduce an overview of SAND (Section 4.2.1). Then, we describe the message dissemination mechanism (i.e., Random Walk algorithm) of Litter (Section 4.2.2). In addition, in order to increase the message delivery rate, we introduce three versions of SAND: SAND based on *Biased Random Walk algorithm* (SAND-BR), SAND based on *Popularity Flooding algorithm* (SAND-PF) and SAND based on *Super Node algorithm* (SAND-SN) (Section 4.2.3). Then we compare the proposed mechanisms, and discuss their advantages and disadvantages (Section 4.2.5). Finally, we discuss implementation issues (Section 4.2.6).

4.2.1 Overview

SAND is a alternative solution to centralized microblogging services (e.g., Twitter), in the situation where users cannot access to the central servers of the microblogging services. In particular, before a partition, nodes are able to exchange self-signed public key certificates, as well as detailed contact information to their direct friends, through central servers (e.g., Twitter's server). Once central servers fail, social friends can connect to each other to form a social-aware overlay [50] by leveraging the information gotten from central servers before the partition. SAND takes advantages of SocialVPN [91], and Litter [141] as its building blocks.

SAND is built upon a SocialVPN overlay where users have private IP connections to their social friends, and consequently enables trusted social communication on a network (even a partitioned network) through the constraints imposed by NATs and firewalls. By leveraging SocialVPN, social friends are able to communicate with each other. Note that, social information are obtained from the current online social networks (e.g., Facebook) before partition events. However, in a microblogging service, a message publisher not only needs to disseminate messages to his social neighbors (the neighbors who follow him), but also needs to disseminate messages to his subscribers who may be multiple social hops away. Then, the challenge of designing SAND is how a publisher sends his messages to his social distant subscribers with acceptable network overheads. SAND is inspired from a previous solution, Litter (which uses a Random Walk algorithm) to design the message dissemination mechanisms. In our chapter, in order to study and evaluate tradeoffs among different properties and delivery rates, we propose three variants of SAND: SAND-BR, SAND-PF, and SAND-SN. In the following, we first describe the concept of Random Walk algorithm used in

Litter, then introduce the improved message dissemination algorithms used in SAND, respectively.

4.2.2 Basic dissemination mechanism in Litter

In Litter, a publisher disseminates messages using Random Walk algorithm (i.e., a publisher randomly pushes his messages to one or more forwarders in the network), while the subscribers pull messages by simply flooding to request their social neighbors.

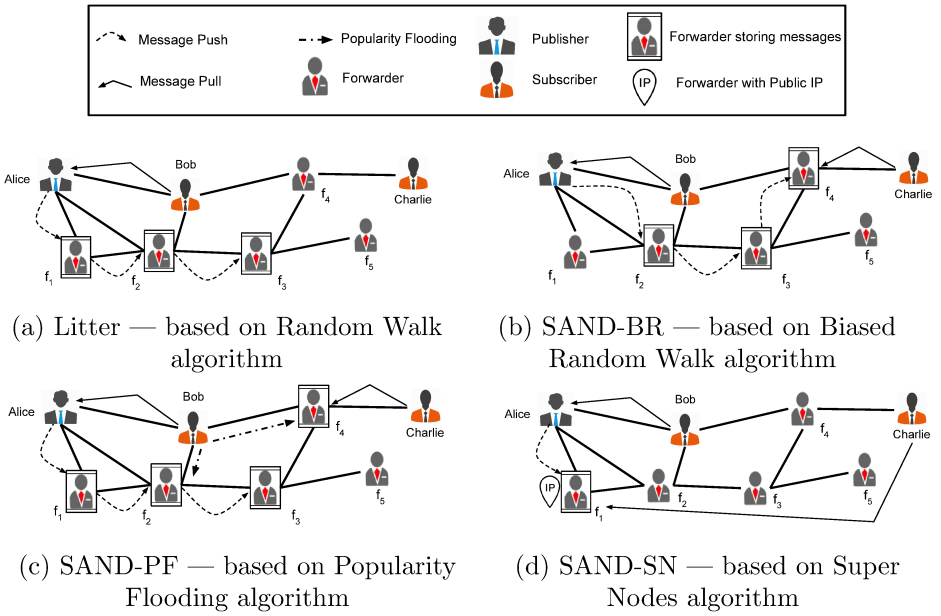
In Random Walk, in order to disseminate messages to subscribers, a publisher pushes messages to the network hop by hop. In particular, a node (a publisher/forwarder) randomly selects one of his neighbors to forward messages. Note that, in Random Walk, a publisher's messages may be stored at nodes multiple social hops away from the publisher. The delivering scope of a message is controlled by a TTL (Time To Live): if a publisher wants to disseminate messages to one of the direct friends, then the TTL is set to 1. In order to enhance the delivery rate, TTL could be set to a large value (e.g., 50, 100) depending on the user's preference. However, selecting a higher TTL increases the chances that distant subscribers will be able to receive the updates of the publisher, while resulting in large network overheads (e.g., transferred messages, stored messages). The TTL serves as the replication factor because a messages is stored at each node the message reaches. Note that, if a node does not have a neighbor to forward a message, it will drop the message even the TTL is not 0.

Figure 4.2a shows an example of how Alice disseminates messages to her subscribers in Litter (TTL is set to 3). Alice has three neighbors: f_1, f_2 and *Bob*. She randomly chooses one of her neighbors (i.e., f_1) to forward a message. Afterwards, f_1 randomly chooses one neighbor of f_1 (excluding Alice) to forward the message, and the message's TTL reduced by 1. After this iteration (the TTL drains out), the message forwarding and storing path is $\{Alice, f_1, f_2, f_3\}$. Therefore, in this case, Charlie cannot retrieve the message from his social neighbor. Note that the message will not traverse a node where it has already passed.

In Litter, nodes in overlay are fully equal, and the overlay is resilient to nodes failures. However, the delivery rate in Litter under a partition event cab be improved. In order to enhance the delivery rate, in this chapter, we propose following versions of SAND: *SAND-BR*, *SAND-PF*, and *SAND-SN*.

4.2.3 Improved dissemination mechanism in SAND

In this section, we describe the message dissemination mechanisms in three variants of SAND: SAND-BR (Section 4.2.3, SAND-PF (Section 4.2.3), and SAND-SN (Section 4.2.3).



(a) Litter — based on Random Walk algorithm

(b) SAND-BR — based on Biased Random Walk algorithm

(c) SAND-PF — based on Popularity Flooding algorithm

(d) SAND-SN — based on Super Nodes algorithm

Figure 4.2: Message dissemination mechanisms. Alice is a publisher; Bob and Charlie are subscribers of Alice.

SAND-BR (based on Biased Random Walk)

A node with large degree has higher probability to be reached by other nodes [120] in the social graph. Therefore, we first propose a Cluster Walk algorithm: a node always forwards a message to a neighbor with largest degree. However, although Cluster walk achieves a desirable delivery rate, network overheads are higher for the set of nodes with relatively large degree. We then propose a tradeoff method — SAND-BR — based on Biased Random Walk algorithm, in which a node has higher probability to forward a message to a neighbor with large degree. Figure 4.3 shows how Alice selects a neighbor to forward messages in Random Walk (Litter), Cluster Walk and Biased Random Walk (SAND-BR), respectively (topology shown in Figure 4.2b). In Random Walk, all neighbors have equal chance to receive messages from Alice. In Cluster Walk, due to f_2 with largest degree, Alice only forwards messages to f_2. In Biased Random Walk, f_2 has highest probability to receive messages from Alice. Neighbors f_1 and *Bob*, nevertheless, still could receive messages from Alice. Note that, in Biased Random Walk, the biased level (i.e., the probability that the message is sent to f_2) can be tuned, in order to trade off the delivery rate and the network overheads.

Figure 4.2b shows an example of how Alice disseminates messages to her subscribers in SAND-BR (TTL is set to 3). Different from Random Walk in Litter (shown in Figure 4.2a), Alice forwards her message to f_2 in SAND-BR. Therefore, the message forwarding and storing path is $\{Alice, f_2, f_3, f_4\}$.

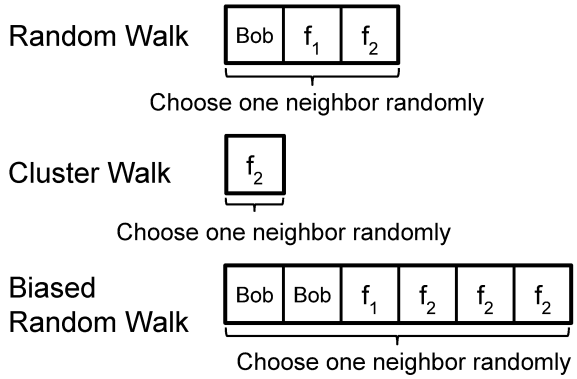


Figure 4.3: Forwarding mechanism in Random Walk, Cluster Walk and Biased Random Walk algorithms, respectively.

Consequently, both Bob and Charlie can retrieve the message from Alice. Biased Random Walk algorithm is a tradeoff approach between Random Walk and Cluster Walk. It provides a high delivery rate, while keeping a relative uniformity of network overheads among nodes in a partitioned network. Note that, in SAND-BR, the forwarding and storing paths of a publisher's messages may not be same.

The procedure of message dissemination in SAND-BR is detailed in Algorithm 2. If the TTL of a message is 0, the node will store this message and stop forwarding (line 2 in Algorithm 2). Otherwise, the node chooses a forwarder from his forwarder list that contains the neighbors that have not stored the message yet (lines 5 and 6 in Algorithm 2). Once the node selects a forwarder from the list (by biased random selecting), the node sends the message to the forwarder. If the forwarder is able to receive this message, it will return a confirmation message to the node (lines 8-10 in Algorithm 2). Note that, a forwarder may not be available for two reasons: (i) the forwarder is located in outside of network partition area; (ii) the forwarder is offline.

SAND-PF (based on Popularity Flooding)

In this section, we introduce SAND-PF based on Popularity Flooding algorithm. Same with Litter and SAND-BR, a subscriber floods requests to his neighbors to retrieve messages. In order to enhance the delivery rate, once a subscriber receives a message, based on the popularity level [17] of the message (assess the characteristics of the message, as well as the publisher of the message), the subscriber will decide whether to flood the message to his neighbors.

In order to identify the popularity of a message, the authors in [53, 78] discuss the following properties: *message content relevance features* (e.g.,

Algorithm 2 Publishers or forwarders disseminate messages in SAND-BR.

Input: *passedList* is a list containing all the forwarders that have the copies of the messages; *ttl* is to control the scope of the message; *message* is the message need to be forwarded.

```
1: procedure pushMessage(passedList,ttl,message)
2:     if ttl = 0 then
3:         return
4:     end if
5:     forwardersList ← publisher.neighbors()
6:     forwardersList.remove(passedList)
7:     while len(forwardersList)> 0 do
8:         forwarder ← biasedchoosing(forwardersList)
9:         forwardersList.remove(forwarder)
10:        flag ← sendMessage(forwarder,message)
11:        if flag = True then
12:            ttl ← ttl − 1
13:            passedList.append(forwarder)
14:            break
15:        end if
16:    end while
17: end procedure
```

semantic analysis [80], time distance to the last message), *microblogging service specific features* (e.g., whether including URL, number of repost times), and *publisher authority features* (e.g., the number of subscriber of the message publisher, whether the message publisher is verified). In view of the decentralized characteristic of SAND, SAND-PF, we consider the following factors to decide whether a subscriber to flood a message:

- *Message content*: semantic analysis.
- *Publisher*: the number of subscribers; verified or not.
- *Surrounding situation*: the proportion of subscriber's neighbors who have the copies of the message.

Algorithm 3 shows the procedure of how to decide if a message is popular. A subscriber will flood a message, if the message satisfies the following requests: (i) the message is popular (by semantic analyzing, line 2 in Algorithm 3); (ii) enough subscribers follows this message (lines 3 and 4 in Algorithm 3); and (iii) the proportion of subscriber's neighbors who receive the message is low.

With a view to the limitation of the simulation (the messages in the simulation do not include semantic content, as well as timestamps), we only consider the following metrics in our simulation: the number of subscribers (who published the message), and the proportion of neighbors (who have

Algorithm 3 Subscriber disseminates messages in SAND-PF.

Input: $receivedNeighbors$ represents the neighbors who have the message's copies; $message$ is the message to be flooded.

Note: $threshold1$ controls whether to flood messages according to contents of the messages; $threshold2$ controls whether to flood messages according to publishers' information; $threshold3$ controls whether to flood messages according to surrounding situation.

```
1: procedure floodMessage(receivedNeighbors, message)
2:     flag1 ← semanticAnalyzing(message) > threshold1
3:     subscribers ← message.owner().subscribers()
4:     flag2 ← len(subscribers) > threshold2
5:     portion ← len(receivedNeighbors)/len(neighbors)
6:     flag3 ← portion < threshold3
7:     if flag1 & flag2 & flag3 = True then
8:         sendMessage(message)
9:     end if
10: end procedure
```

the copies of message). In particular, once a subscriber s receives a message posted by a publisher p, if p's number of subscribers is larger than a threshold, and the proportion of s neighbors who have the copies of the message is less than a threshold, s will flood the message to all his neighbors (the values of the thresholds are discussed in Section 4.4.1).

Figure 4.2c shows an example of how Alice disseminates messages to her subscribers in SAND-PF (TTL is set to 3). The message forwarding and storing path is same as the one in Random Walk algorithm, $\{Alice, f_1, f_2, f_3\}$. Once Bob's overlay node receives the message from Alice, it decides that this message is meaningful. Moreover, it realizes that Alice has many subscribers, and the proportion of his neighbors who have the copies of this message is low. Under this circumstances, SAND-PF floods this message to all of Bob's neighbors. As a result, a copy of the message is also stored at f_4, where Charlie, another one of Alice's subscriber, is able to retrieve the message.

To avoid spam messages, a node may set a filter to block a set of popularity-based flooded message. Moreover, only subscribers assess messages to decide whether to flood, according to the characteristics of the messages. The reason for this behavior is: in SAND, forwarders only forward and store messages, in order to reduce the resources usage (i.e., computing).

SAND-SN (based on Super Node)

In addition to the methods based on walk-based algorithms (i.e., Litter, SAND-BR, and SAND-PF), we make use of *super node* concept [166] in P2P network and propose SAND-SN based on Super Node algorithm, which

achieves better message delivery rate, while reducing the network overheads. Super nodes indicate nodes with Public IP address, which can be communicated directly by the nodes with private IP address. Figure 4.2d shows an example of how Alice disseminates messages to her subscribers in SAND-SN. Before the partition, each publisher (e.g., Alice) maintains a list, named Super Nodes List (SNL), that contains a set of nodes with public IP address within a certain social hop count (e.g., within 2 social hops). Each of Alice's subscribers has a local copy of Alice's SNL. After network partitions, according to her SNL, Alice sends messages to the public-IP nodes where Alice's subscribers could retrieve the messages. Alice forwards her messages to f_1. Given that f_1 has public IP address, other subscribers (i.e., Charlie) of Alice are able to connect to f_1 directly to retrieve the message from Alice.

One situation needs to be cautiously considered in SAND-SN is offline status of super nodes. In SAND-BR and SAND-PF, we do not consider failed nodes or offline nodes, since all nodes work together with equal roles. However, super nodes may be unavailable even inside partition networks for three reasons: (i) super nodes themselves may be offline; (ii) the overheads on some of the super nodes can become significant, so that super nodes cannot serve requests from publishers and subscribers; (iii) nodes with public IP address are more vulnerable to censorship. Therefore, as a consequence, the following situation might happen: a publisher could only reach a portion of super nodes in his SNL.

As shown in Figure 4.4, Alice is a publisher and Bob is one of her subscribers. Note that, in Figure 4.4, we only consider Bob as Alice's only existing subscriber; s_1, s_2, s_3 and s_4 are four super nodes in Alice's SNL. Although the ID of the last message Alice posts is 42, due to the offline status of s_1, s_2, s_3 and s_4, several messages are absent from these super nodes. For example, the messages with the IDs ranging from 37 to 41 stored in s_1 are missed. The reason of this behavior is that, s_1 is offline when Alice posts messages with the IDs ranging from 37 to 41. To mitigate a "message missing" problem caused by nodes' offline status, in SAND-SN, we propose a push mechanism and a pull mechanism for publisher and subscriber, respectively.

Publisher: When pushing messages, in order to guarantee enough super nodes to receive messages, a publisher considers a threshold, indicating the proportion of super nodes that are online and could receive messages. The publisher will stop sending messages to the super node as long as the proportion of super nodes who receive messages is beyond the threshold. Algorithm 4 shows how a publisher pushes a message to his super nodes. First, the publisher randomly selects a super node from his SNL (lines 3 and 4 in Algorithm 4). Then the publisher sends his message to the super node. If the super node is available, he will send a confirmation message to the publisher (line 5 in Algorithm 4). If the portion of super nodes who

have the message's copy is larger than a threshold, the publisher will stop pushing messages (line 9 in Algorithm 4).

Subscriber: When a subscriber receives a message whose ID is not consistent with previous one (which means the subscriber has missed some messages), the subscriber will traverse all the super nodes (in the SNL of a node who generates that message) to request the missed messages. Algorithm 5 shows how a subscriber retrieve messages if the subscriber misses one or more messages. First, the subscriber sends a request message to a super node (the super node is randomly selected from the SNL of the publisher who sends that message) to request recent messages generated by the publisher (lines 4-7 in Algorithm 5). Once the subscriber receives the messages from the super node, he updates his messages list to check whether the missed message is retrieved (lines 8 and 9 in Algorithm 5). Note that, the subscriber may find more missed messages in this procedure. If the subscriber still misses one more messages, he will send a request message to another super node. Otherwise, if he receives all the messages, he will stop sending messages to super nodes. In Figure 4.4, after retrieving messages from s_1, Bob realizes that some of Alice's messages (i.e., ID from 37 to 42) are absent. Then Bob retrieves these missed messages from s_2, s_3 and s_4, respectively.

Algorithm 4 A Publisher disseminates messages to forwarders in SAND-SN.

Input: *snList* is the Super Node List of the publisher; *message* is the message need to send to super nodes.

Note: If the portion of super nodes who receive messages larger than the *threshold*, the publisher stop disseminating messages.

```
1: procedure pushMessage(snList,message)
2:     count ← 0
3:     shuffle(snList)
4:     for each sn ∈ snList do
5:         flag ← sendMessage(sn, message)
6:         if flag = True then
7:             count ← count + 1
8:         end if
9:         if count/len(snList) > threshold then
10:            break
11:        end if
12:    end for
13: end procedure
```

4.2.4 Message field

In this section, we discuss the fields of a forwarding message (sent from a publisher to a forwarder), a requesting message (sent from a subscriber to a

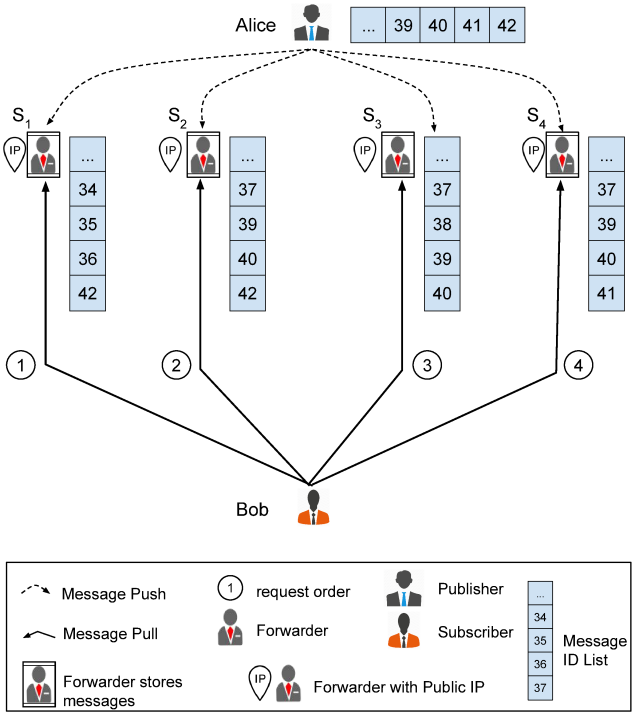


Figure 4.4: SAND-SN. Forwarders s_1, s_2, s_3, and s_4 are four super nodes that store and forward Alice's messages. Each super node maintains a list that contains the messages from Alice, ordered by message ID. Message ID is an integer that increments by one for each message published by a publisher. Assume Bob successfully pulls the messages with the IDs that are less than 34.

forwarder), and a delivering message (sent from a forwarder to a subscriber), respectively.

Forwarding message: A forwarding message is a message sent from a publisher/forwarder to a forwarder. A forwarding message is generated by publishers and forwarded by forwarders. It contains the following information: a *Message ID*, a *Publisher ID*, a *Message Delivery Type*, a *TTL*, a *Timestamp* , a *Passed Forwarders List*, a *Message Playload*, and a *Signature*. Figure 4.5 shows the details of forwarding message field.

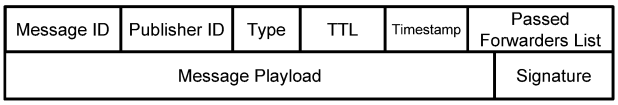


Figure 4.5: A forwarding message fields.

Algorithm 5 A Subscriber retrieves messages from forwarders in SAND-SN.

Note: *message* is the received message.

```
1: procedure pullMessage( )
2:     missedMessageList ← MissedMessage(message)
3:     if len(missedMessageList) > 0 then
4:         publisher ← message.owner()
5:         snList ← publisher.snl()
6:         shuffle(snList)
7:         for each sn ∈ snList do
8:             messagesList ← retrieveMessages(sn,publisher)
9:             missedMessageList.update(messagesList)
10:            if len(missedMessageList) = 0 then
11:                break
12:            end if
13:        end for
14:    end if
15: end procedure
```

The Message ID is an integer that increments by one for each message, and then allows subscriber to track missed messages created a publisher. The Publisher ID helps Forwarders and Receivers keeping track of the source who posted the message. The Message Delivery Type indicates the type of the dissemination algorithms (i.e., Random Walk, Biased Random Walk, Popularity Flooding, and Super Node). The TTL field controls the scope of delivering the message. The Timestamp tracks the creation time of the message. The Passed Forwarders List contains all forwarders that have the copies of the message, in order to avoid transferring loop. The Message Playload is the actual content of the post. Currently it is limited to a maximum of 140 characters. The Signature field is created by hashing the publisher ID, the timestamp, the post ID, and the message, then signed with the publisher's private key. Therefore the signature ensures the integrity of the message and allows the verification of its publisher. Since SAND is a alternative solution for current microblogging services, subscribers are able to keep publishers' public keys before networking partitions, through the official microblogging server.

Requesting message: A requesting message is a request message sent from subscriber to forwarders, which contains the following information: a *Subscriber ID*, a *Publishers List*, a *Timestamp*, and a *Signature*. Figure 4.7 shows the details of requesting message field.

Figure 4.6: A requesting message fields.

The Subscriber ID helps Forwarders tracking the source who sends request message. The Publishers List includes a set of publishers that the subscriber follows. The Timestamp tracks the creation time of the message. The signature field is created by hashing the Subscriber ID, the Publishers List and the Timestamp, then signed with the subscriber's private key. Therefore the signature ensures the integrity of the Publishers list and allows the verification of the subscriber. In SAND-BR and SAND-PF, when a subscriber retrieve messages, he sends requesting messages to all his neighbors (direct friends). A node keeps his friends' public key before partitions. In SAND-SN, a requesting message sent from a subscriber to super nodes. Therefore, a nodes in SAND exchanges the public keys with the super nodes in the SNL of publishers the node follows.

Delivering message: A delivering message is a message sent from a forwarder to a subscriber, and it is generated by a forwarder who stores messages that the subscriber requests. The message contains the following information: a *Forwarder ID*, a *Subscriber ID*, a *Timestamp*, a *Messages Dictionary* and a *Signature*.

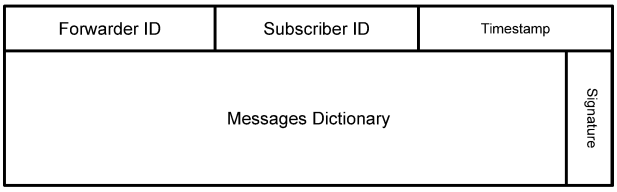


Figure 4.7: A delivering message fields.

The Forwarder ID is a identity of a node who stores the messages requested by subscriber. The Subscriber ID is a identity of a node who requests the messages. The Timestamp tracks the creation time of the delivering message. The Messages Dictionary is a set of messages (from different publishers) that the subscriber requests. The Signature field is created by hashing the Forwarder ID, Subscriber ID, the Timestamp, and Messages Dictionary, then signed with the forwarder's private key. Therefore the signature ensures the integrity of the Message Dictionary and allows the verification of the forwarder.

4.2.5 Comparision of proposed mechanisms

Comparing with Litter (i.e., random walk), SAND-BR, SAND-PF and SAND-SN provide higher delivery rate in network partition events, where large numbers of nodes fail. In this section, we further discuss the advantages and disadvantages of proposed mechanisms.

SAND-BR achieves higher delivery rate by disseminating messages (with higher probability) to neighbors with larger degree. As a result, in a network,

part of the network overheads (i.e., message sending, message receiving and message storing) are assembled on these nodes who may become network bottlenecks. In other words, the distribution of network overheads is not uniform. Because of the heavy network overheads, nodes with large degree may not be willing to support the system. However, the following explanations are able to mitigate this issue: (1) since nodes with more neighbors might post more messages and have more followers in a microblogging system [98], they need more people to help them to disseminate messages. In return, these nodes need to afford more network overheads; (2) the bias level (i.e., the probability that a message sent to node with larger degree) can be configured.

SAND-PF obtains higher delivery rate by disseminating messages to more nodes based on the popularity of messages. As a result, total network overheads increase. This is a tradeoff between delivery rate and networking overheads. Note that SAND-BR and SAND-PF are able to be used together, in order to achieve a desirable delivery rate and network overheads.

SAND-SN obtains 100% delivery rate (based on the evaluation setup shown in Section 4.4.2) by leveraging nodes with public IP addresses. In SAND-SN, public-IP nodes become relay servers that store and distribute messages. The total network overheads are lower than the one in SAND-BR and SAND-PF. However, the disadvantage of SAND-SN is that the public-IP nodes are vulnerable to censorship. Authorities are potentially able to obstruct the system by blocking the public IP addresses of these nodes. Therefore, SAND-SN is suitable to work in the situation of network partition caused by accidents, such as earthquake and hurricane [44].

4.2.6 Implementation

By leveraging SocialVPN[6] and based on the prototype of Litter[7], we implement the following functionalities in SAND: object serialization, data encryption, data transmission, data storage, and web interface. Note that, we use SQLite for the data storage. Our implementation is written in Python. Figure 4.8 shows the user interface of SAND.

To communicate, we created a service that listens on a user-specified UDP port for incoming messages or requests. To retrieve messages, the request payload is encapsulated in a UDP datagram that is sent to the multi-cast IP address (social neighbors) through SocialVPN, so that the listening service is able to receive the request message. Note that, SocialVPN maintains the friendship list/SNL for users.

[6]The IPOP Project is available at: `https://github.com/ipop-project/`
[7]Litter is available at: `https://github.com/pstjuste/litter/`

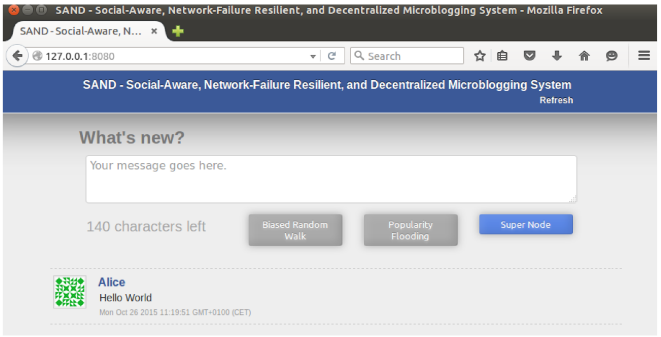


Figure 4.8: Interface of SAND.

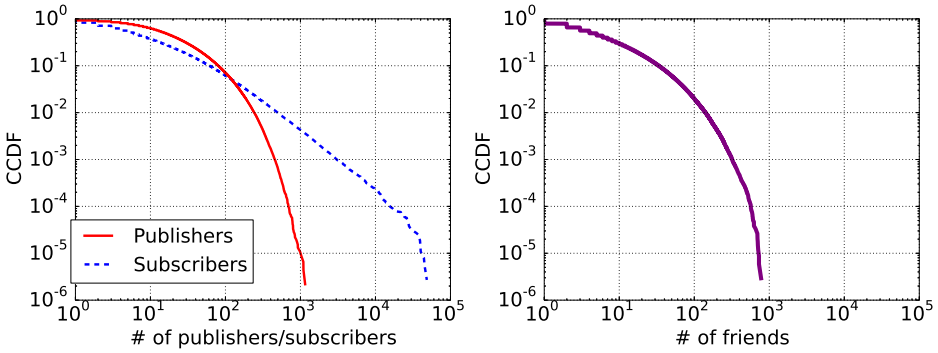
4.3 Dataset

We are the first to simulate and analyze the performance of the proposed decentralized microblogging system on a dataset with real publisher-subscriber distribution, in a partitioned network. In addition, we also run our evaluation based on a synthetic dataset. This section describes the datasets used in our evaluation. In particular, in Section 4.3.1 we describe how we extract the friendship from the publisher-subscriber relationship in the Twitter dataset; and how we construct the synthetic dataset. Section 4.3.2 describes how we assign location information for nodes and generate partitions with different remaining nodes. Finally, in Section 4.3.3, we describe how we select the partition areas that use in the evaluation of SAND.

4.3.1 Initial graph

In order to simulate the decentralized microblogging behavior in a partitioned area, the initial number of nodes in the dataset must be large. Furthermore, we not only need the information about friendship of users in order to simulate SocialVPN links, but also need the information about publisher-subscriber distribution in order to simulate the message dissemination pattern. We obtained the Twitter dataset from [46]. The raw data consist of 456,626 users and 14,855,842 directed edges. To extract friendship among users, we convert the directed edges to undirected friendship using the following principle: if there are bi-directional edges between two users, we can assume they are friends to each other; otherwise, they are just a publisher and a subscriber. Note that, since the Twitter dataset does not provide temporal information about online and offline nodes, we assume that all the nodes are online users in our simulation. Nonetheless, we discuss the issue of "offline nodes" in Section 4.5.

Figure 4.9a displays the distribution of the number of publishers as the solid line and that of subscribers as the dotted line. The curve of Figure 4.9a follows the observation described in [98].



(a) Publishers/subscribers (b) Friends

Figure 4.9: Analysis of social relations. The y-axis represents complementary cumulative distribution function (CCDF).

Figure 4.9b displays the distribution of the number of friends (after converting undirected graph to directed graph). The curve of Figure 4.9b follows the observation described in [147].

Considering that this Twitter dataset only includes 456,626 nodes, it could not show a comprehensive behavior of a system that has similar scale as Twitter (billions of users). Therefore, we also use a synthetic dataset in our evaluation since it is able to provide more flexibility (i.e., different number of total nodes in a network). In particular, we evaluate the delivery rate of SAND-BR, SAND-PF and SAND-SN, in networks with different number of nodes, based on the synthetic datasets.

In the following, we describe how we set up a synthetic dataset for use in our evaluation. First, we generate a social network (including nodes and their friendships) based on a modified Nearest Neighbor Model [134]. This model follows the observation that two people sharing a common friend are more likely to become friends. Second, we add subscribers for each node in the social network. The assumption is based on [98]: if node a is socially close to node b, there is a higher probability that a subscribes to b, or b subscribes to a. Note that we create the synthetic datasets using metrics that proportionally approximate those found in the Twitter dataset.

4.3.2 Partition graph

In order to partition a network, for each node in the network we get from Section 4.3.1, we use *Loc-Generation* algorithm (introduced in [50]) to assign location information[8]. Then based on the location of each node, we cut a certain area to simulate network partition events. *Loc-Generation* algorithm is based on the following observation [15]: the shorter is the distance among

[8]The location information is based on the Cartesian coordinate system.

two nodes, the higher is the probability for them to be friends. Algorithm 6 shows the procedure to determine geo-location information for each node.

Algorithm 6 Assign geo-location information to each node.

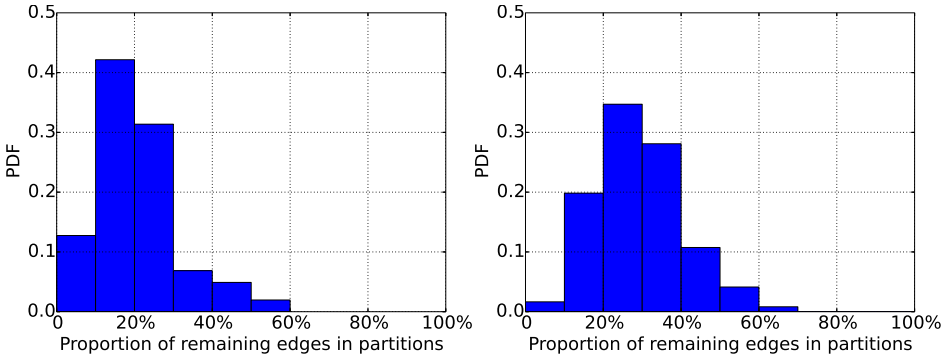
```
1: procedure GENELOCATION(nodesList) Input: nodesList a set of nodes need
   to be assigned location information.
2:     nodesWithLocation ← ∅
3:     for each node ∈ nodeList do
4:         neighborsWithLocation ← ∅
5:         neighbors ← node.neighbors()
6:         for each neighbor ∈ neighbors do
7:             if neighbor ∈ nodesWithLocation then
8:                 neighborsWithLocation.add(neighbor)
9:             end if
10:        end for
11:        if len(neighborsWithLocation) > 0 then
12:            neighborWithLocation ←
                   randomSelect(neighborsWithLocation)
13:            node.setLocationBased(neighborWithLocation)
14:        else
15:            node.setLocationRandomly()
16:        end if
17:        nodesWithLocation.add(node)
18:    end for
19:    return nodeList
20: end procedure
```

For each node i in the *nodesList* ($1 \leq i \leq N$; where N is the total number of nodes in the network), the algorithm assigns location information to it. If at least one of i's friend has been assigned location information already, the algorithm assigns the location information to i based on one of i's random friends, *neighborWithLocation*. The geo-location information is assigned according to the observation [15] that i is more likely to be closed to *neighborWithLocation* (line 11 in Algorithm 6). Otherwise, if all of i's friends have not yet been assigned location information, the algorithm will assign i the geographical location according to a uniform distribution in the network (line 14 in Algorithm 6).

Nodes are distributed in a Cartesian coordinate system. The outage area that includes w nodes is defined as: the smallest circular area that has the center in the origin and includes w nodes. We get two subnetworks from the whole network after performing partition: the outage area (partitioned area), and the outside area (area with the central servers of microblogging services). The size of the partition can be controlled by adjusting the number of nodes in the partition. Therefore, we can evaluate the delivery rate on different sizes of partitions.

4.3.3 Selecting the partitions

Authors in [147] show that, if a country partitioned from the whole social network graph, on average 84.2% percent of edges will remain within the country. However, due to the limitations of the synthetic algorithm (i.e., *Loc-Generation*), after partitioning, the range of proportions of remaining edges in partitions are from around 10% to around 50%. Figure 4.10 shows the distribution of the proportion of remaining edges in partitions. In particular, partitions include 2% nodes in Figure 4.10a, while partitions include 4% nodes in Figure 4.10b. As shown in Figure4.10, the average proportion of remaining edges in 4% partition is larger than the one in 2% partitions. This is because the larger the number of nodes in the partition, the higher the possibility for path to exist between nodes and their neighbors [50].



(a) Partition with 2% nodes (b) Partition with 4% nodes

Figure 4.10: Proportion of remaining edges in partitions. The y-axis represents probability distribution function (PDF). The results are obtained from 100 simulation runs.

Since the location information is generated by *Loc-Generation* algorithm, the remaining nodes and their friendships of the partitions (generated based on the description in Section 4.3.2) is different in different simulation runs. To be more realistic, we select partitions with more than 40% remaining links in the following evaluation. Note that, this is still a conservative number, based on the discussion on the previous paragraph.

4.4 Evaluation

In this section, we perform a simulation-based evaluation to assess the delivery rate and the network overheads (i.e., transferred messages, stored messages) of SAND. In particular, we describe the experimental setup in Section 4.4.1. Then we report the results in Section 4.4.2. The evaluation is

analyzed with the NetworkX graphing library[9], which is a Python package for examining complex networks.

4.4.1 Experimental setup

In our evaluation, we first analyze the *delivery rate* of SAND in different size of networks, and in different size of partitions. In addition, we analyze the detailed probability distribution of *delivery rate*, *stored messages* and transferred messages (i.e., *sent messages* and *received messages*) in a simulation run (partition size is 2%). In the following, we define the metrics reported in our evaluation.

- *Delivery Rate*: For a publisher i, it posts x_i messages ($0 < x_i \leqslant 5$). For a message, we assume q ($0 < q \leqslant m$) subscribers receive it (m subscribers follow the publisher). Note that messages may be disseminated by different paths, so that the subscribers who receive the messages may be different in different message dissemination turns. Therefore, the publisher's delivery rate r_i is:

$$r_i = \frac{\sum_{j=1}^{x} q_j}{m_i x_i}$$

  In addition, assume a network have n publishers. The delivery rate of network R is:

$$R = \frac{\sum_{i=1}^{n} r_i}{n}$$

- *Stored Messages*: When a forwarder receives a message from a publisher or another forwarder, the forwarder first stores the message, then forwards the message based on the remaining TTL. In our simulation, we analyze the probability distribution of messages stored on each node in partitioned networks. Note that the number of stored messages reflects the number of used TTL in SAND-BR. While in SAND-SN, this number indicates the number of super node in publishers' SNL after partitions.

- *Sent Messages*: In our evaluation, we analyze the number of sent messages and received messages respectively. In particular, we assess the probability distribution of sent messages on each node in a partitioned network in a simulation run.

- *Received Messages*: We assess the probability distribution of received messages on each node in a partitioned network in a simulation run. Note that the number of received messages of a node may be different from the number of sent messages of the node. The reason for this

[9]Networkx is available at: `http://networkx.github.io/`

behavior is: the node may send a message to a forwarder/super node who is located outside of a partition. In this case, the node cannot receive the reply messages from the outside forwarder/super node.

Authors in [98] show that nodes with large number of subscribers, or following large numbers of publishers, have higher probability to post more message. In the evaluation, each node, as a publisher, sends 1-5 messages according to the number of his subscriber and number of publishers he follows. For Litter, SAND-BR and SAND-PF, TTL is set to 100, in the the Twitter-dataset evaluation. In the synthetic-dataset evaluation, TTL changes according to the total number of nodes in the network. In SAND-PF, the threshold of neighbors with the copies of a messages is 30%, while the threshold of the publisher's (who posts the messages) number of subscribers is 20. According to [60], and considering increasing prevalence of IPv6, the portion of public-IP nodes accounts for 20% (out of overall nodes), in SAND-SN. In addition, the number of super nodes in SNL is 20, and the maximum hops when a node selects super nodes in SNL is 3. Table 4.2 shows the input value we considered in our evaluation.

Parameters	**Value**
Range of generated messages per publisher	1-5
TTL (Litter, SAND-BR and SAND-PF)	100
Threshold of neighbors with message copies (SAND-PF)	30%
Threshold of number of subscribers (SAND-PF)	20
Proportion of Public-IP nodes (SAND-SN)	20%
Total items in SNL (SAND-SN)	20
Maximum hops of super nodes in SNL (SAND-SN)	3

Table 4.2: Input Parameters.

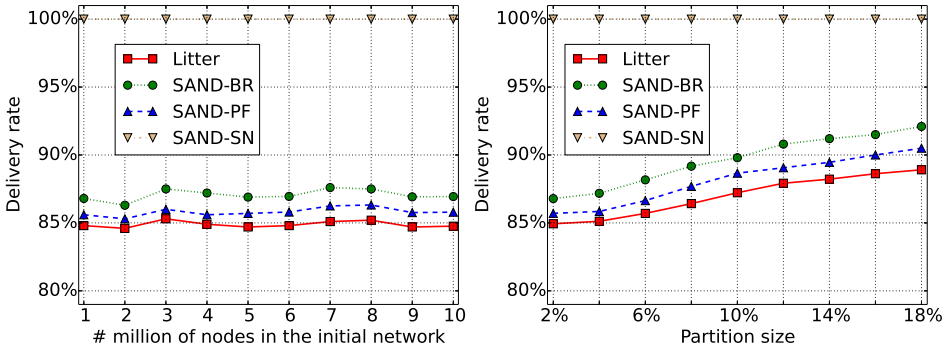
4.4.2 Results

In this section, we first show the delivery rate in the different size of networks (proportions of partition are same). Then, we show the delivery rate in different size of partitions (the initial number of nodes in networks are same). In addition, because network costs are important factors that need to be considered, we then report the distribution of delivery rate, as well as network overheads (i.e., stored messages, sent messages and received messages) in the partition with 2% nodes. Note that, for the following metrics, we report the average of the results obtained for the 50 simulation runs.

Delivery rate in different size of networks

In this section, we show the delivery rates of Litter, SAND-BR, SAND-PF and SAND-SN in networks with different initial nodes (the proportions of partition are same). Since we analyze the delivery rates in the situation where the number of nodes in the initial network changes, this evaluation runs on top of the synthetic dataset.

Figure 4.11a shows changes in average delivery rate for different size of networks. Note that, the proportions of partitions are 2% in this evaluation. When the number of nodes in the initial network increases, the delivery rates of Litter, SAND-BR, and SAND-PF fluctuate around 84%, 86% and 87%, respectively. This is because these walk-based mechanisms are fully distributed, and they do not rely on any central server. The delivery rate of SAND-SN is 100%. The reason of this behavior is that, publishers are always able to find public-IP forwarders who could distribute messages to subscribers.



(a) Delivery rate in different size of networks (synthetic dataset).

(b) Delivery rate in different size of partitions (Twitter dataset).

Figure 4.11: Analysis of delivery rate in different size of networks, and in different size of partitions.

Since the hardware limitations of our clusters used for this evaluation, we cannot run this evaluation on a network with larger number of nodes. In this evaluation, censored area includes 2% nodes of total network nodes. Therefore, there are 200,000 nodes in the censored area when the initial network includes ten million nodes. While in a realistic scenario, Egypt has 519,000 Twitter users[10]. Although the number of censored nodes in the simulation is smaller than the number of Egyptian Twitter users, it is on the same order of magnitudes as for the Egyptian Twitter users.

[10] `http://www.arabsocialmediareport.com/Twitter/LineChart.aspx`

Note that, since the following evaluations do not need to change the number of nodes in the initial network, all these evaluations runs on top of the Twitter dataset.

Delivery rate in different size of partitions

Figure 4.11b shows changes of average delivery rate, considering partitions with different numbers of nodes. In Litter, SAND-BR, and SAND-PF, the delivery rates slightly increase when the size partitions increases. The reason for this behavior is: as shown in Section 4.3.3, a partition with larger size, has higher probability to maintain large proportion of remaining edges. As a result, publishers have higher probability to find delivery paths to forward messages to subscribers in large size partitions.

Moreover, the average delivery rates in SAND-BR and SAND-PF are higher than the one in Litter. The reasons for this behavior are: (i) in SAND-BR, a node with higher degree has higher probability to be a forwarder to store and transfer messages, and a large-degree forwarder connects more nodes, therefore subscribers have higher probability to reach a large-degree forwarder in order to retrieve the messages. (ii) In SAND-PF, a subscriber will diffuse a message if the message is 'popular'. As a result, the subscribers of the message have higher probability to get this message. Note that, as we discussed in Section 4.3.3, the proportion of remaining edges in partition network in our simulation is lower than the one in reality. Therefore, the results shown in Figure 4.11b are still conservative. We believe in a real usage scenario, the delivery rate could be even better.

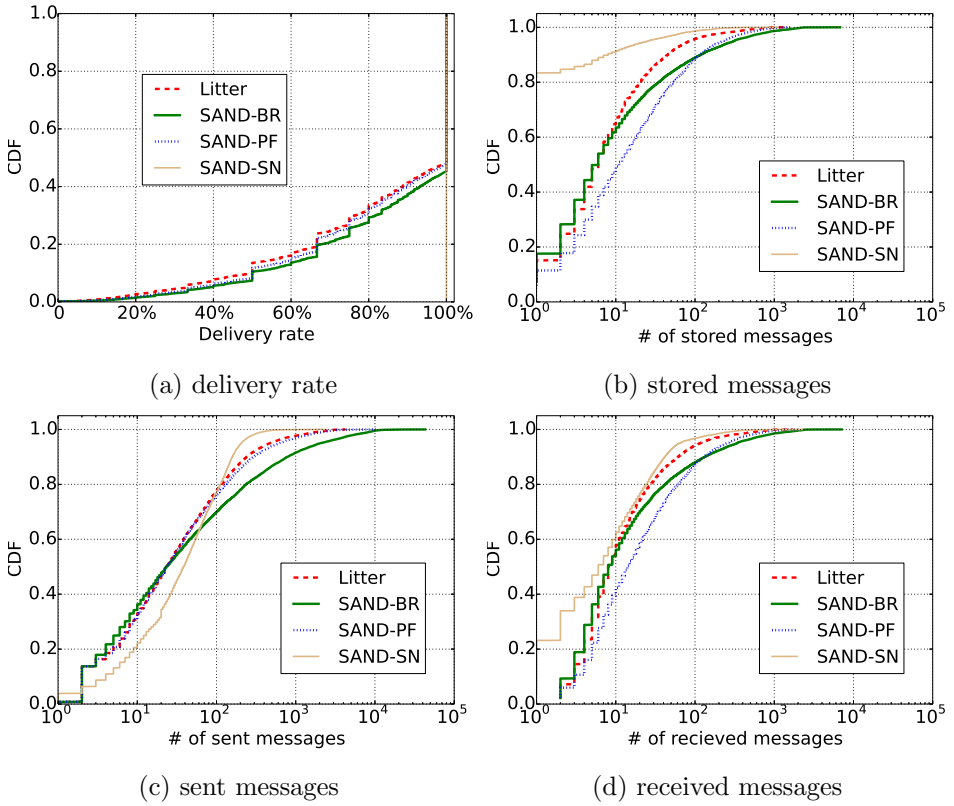
In SAND-SN, all the nodes in partition networks achieve 100% delivery rate. The reason for this behavior is that, all publishers are able to find at least a forwarder with Public IP address to forward messages to their subscribers. Note that, the delivery rate in practice may not be this high for two reasons: (i) proportion of public IP address is different from country to country (from area to area). A publisher is not able to find a public IP node within a short social hop in a partition area; (ii) A public IP forwarder is not available even inside a partition area due to censorship, or excessive forwarding requests.

Detailed evaluation of a partition with 2% nodes

In this subsection, we report the distribution of delivery rate, the number of stored messages, the number of sent messages, and the number of received messages, respectively, in a partition with 2% nodes.

Distribution of delivery rate: Figure 4.12a shows the delivery rate distribution in Litter, SAND-BR, SAND-PF, and SAND-SN, respectively. The Delivery rate of Random Walk, Biased Random Walk, Popularity Flooding and Super Node are: 84.9%, 87.8%, 86.0% and 100%, respectively.

In particular, for around 60% of the nodes, the delivery rates achieves 100% in Litter, SAND-BR, and SAND-PF, while all the nodes in SAND-SN have 100% delivery rate.



(a) delivery rate

(b) stored messages

(c) sent messages

(d) received messages

Figure 4.12: Detailed evaluation of a partition with 2% nodes. The Partition includes 10,000 nodes. The y-axis represents cumulative distribution function (CDF). In Figure 4.12b, Figure 4.12c and Figure 4.12d, the x-axis is in log scale.

There are noticeable discontinuities in the lines that represent Litter, SAND-BR and SAND-PF. They occur at $x = 1/2$, $x = 2/3$, $x = 3/4$, and $x = 4/5$. The reason for this behavior is that the publishers who have 2, 3, 4 or 5 subscribers, although unable to deliver their messages to all of their subscribers, are able to deliver the message to most of their subscribers (only one subscriber cannot be delivered).

Distribution of number of stored messages: Figure 4.12b shows the distributions of number of stored messages on nodes in a partitioned network in a simulation run, in Litter, SAND-BR, SAND-PF, and SAND-SN, respectively. Specifically, the number of stored messages in SAND-BR and SAND-PF is larger than the one in Litter. The reason for this behavior is: in SAND-BR, a node always has higher probability to send a message to a

forwarder with large degree, which indicates nodes have higher probability to find a forwarder before TTL exhausted. As a result, more nodes hold the copies of messages in SAND-BR. Note that, the nodes with a small degree in SAND-BR have smaller probability to be a forwarder. As shown in Figure 4.12b, in SAND-BR, there are more number of nodes with less than seven messages copies, and more number of nodes with large number of messages copies (more than 2000). Note that large number of messages on a node may result in node resource exhaustion and failure, and thus needs to be mitigated.

In SAND-PF, after a message dissemination process from publishers to forwarders, once a subscriber receives a message, the subscriber may also flood the message to his neighbors. Therefore, as shown in Figure 4.12b, the number of messages in Popularity Flooding is always higher than the one in Random Walk algorithm.

In SAND-SN, only forwarders (super nodes) in publishers' SNL store messages. Although a publisher sends his messages to several forwarders (super nodes), this sending process is only in one hop. In other words, a forwarder does not send messages to another forwarder. As a result, the number of stored message is low. In addition, only the nodes with public IP address can be the forwarders. In our simulation, the public IP proportion is set to 20%. Therefore, as shown in Figure 4.12b, more than 80% nodes do not store any message (few public IP node cannot be reached by publishers). Note that, the length of SNL, the Proportion of public IP address, and the size of partitions are other factors that affect the number of stored messages in SAND-SN. The length of SNL in our simulation is set to 20. The number of stored messages will increase if the length of SNL becomes larger.

Distribution of number of sent and received messages: In the transferred messages analysis, we assess sent messages and received messages separately, in a simulation run. Figure 4.12c shows the distribution of the number of sent messages in Litter, SAND-BR, SAND-PF, and SAND-SN, respectively. In SAND-SN, there is a noticeable change at $x = 20$. The reason for this behavior is that, the length of SNL in our simulation is 20. For the nodes who do not follow publishers, they only send 20 messages to their super nodes to publish messages.

Figure 4.12d shows the distribution of the number of received messages in Litter, SAND-BR, SAND-PF, and SAND-SN, respectively. Nodes send/receive less messages in SAND-SN compared to Litter, SAND-BR, and SAND-PF. The reason of this behavior is that, for the node with private IP address in SAND-SN, they cannot serve as forwarders. As a result, the private-IP nodes send/receive messages when they are publishers (i.e., send post to forwarders, receive confirmation messages from forwarders) or subscribers (i.e., send request to forwarders, retrieve messages from forwarders). However, in Litter, SAND-BR, and SAND-PF, every node can be a forwarder if their neighbors send them forwarding requests.

Nodes in SAND-BR and SAND-PF send/receive more messages compared to Litter. The motivations for this behavior are: (i) in SAND-BR, a message has higher probability to use more TTL when it is forwarded in a partitioned network (as shown in Figure 4.12b, TTL serves as the replication factor because a messages is stored at each node the message reaches). (ii) In SAND-PF, after a message dissemination from publishers to subscribers, subscribers also may flood the message to their neighbors according to the popular level of the message.

4.5 Discussion

In this section, we first discuss a possible solution for how a node bootstraps into the overlay after network partitions in SAND. Next, we discuss the issue of offline nodes. Finally, we point out the incentives for end users to adopt SAND after network partition.

Bootstrapping: Bootstrapping needs to be considered when network partition occurs in SAND. Although this is not within the scope of this chapter, we have an approach that is able to assist nodes in partition network to bootstrap into the overlay even under the constraints imposed by firewalls and NATs. In order to assist nodes with private IP address to bootstrap into the overlay, nodes with public IP address may serve as STUN or Traversal Using Relays around NAT (TURN) servers. STUN servers allow NATed nodes to connect directly in a P2P fashion, while TURN servers allow a relayed link through an intermediary. In order to connect to STUN or TURN servers (public-IP nodes) and to discover other peers after a partition, each node maintains a boot list, created before the partition, which contains information of several public nodes. Note that, public nodes in ones boot-list are ones nearest (shortest social-hops) nodes with public IP address. If network partition happens, according to the boot-list, each node in the outage area is able to establish links to public node in its boot-list.

Offline Nodes: SAND disseminates and store messages based on a social-aware overlay [50] where people are able to communicate with their social friends directly. Therefore, in the case of offline forwarders (inside partitioned area), nodes could still communicate with other online forwarders. In other words, if the first choice is offline, a node can select an alternative choice. Furthermore, given that forwarders are able to store messages, even a follower offline during a message dissemination, the follower still could retrieve the message once it becomes online (as long as the follower has at least one direct friend who is a forwarder). Note that, 'offline' is not a permanent status. When a node becomes online, it will re-establish all the connections with other online social friends. Authors in [62] leverage the concept of 'personal overlay' to connect all devices that belongs to one person. A personal overlay represents a node (a person) in our system. Although one individual

device (e.g., smartphone) may be offline at some point, there can be other online devices (e.g., laptop, iPad) belonging to the same user who is online and connects to our system. Moreover, people tend to own more and more personal online devices [60], and the online time of devices tends to increase, considering that data plans become increasingly accessible to end users. As a result, SAND is resilient to offline nodes even the nodes are inside the partitioned area.

Incentives: In Tor [52] and VPN gate [125], many people provide their PCs and laptops to help other people. Therefore, we believe many people will provide their computing devices to help others store and forward messages. For the nodes who contribute to SAND, in return, they also benefit from the SAND since they can send out messages to their subscribers. Note that, a policy a node may have is to store up to 'N' messages and then have a replacement policy based on timers - after an expiration period if messages get to use too much space. In other words, users may not need to contribute a lot to the service, and the whole set of users can benefit from it.

4.6 Summary

In this chapter, we propose three different variants of SAND: SAND-BR, SAND-PF, and SAND-SN. We assess SAND through a simulation-based analysis considering a Twitter dataset as well as a synthetic dataset. Our simulation results show that SAND is feasible and efficient, also in case of network partitions. Compared with the related work, our system does not rely on DHT, and provides a desirable delivery rate with acceptable network overheads.

Chapter 5

SEnD: a Social Network Friendship Enhanced Decentralized System to Circumvent Censorships

The Internet has evolved to play a significant role in providing a medium for spreading news and organizing political activities [127]. Politicians, activists and citizens use the Internet to coordinate their activities and share their ideas in ways that are not otherwise possible via traditional media [47]. For instance, during the Arab Spring in 2011, several mainstream websites (e.g., Facebook, Twitter, YouTube) had a great influence on distributing important information and news [159]. Therefore, free access to Internet content and services has been regarded as a serious threat by dictatorships [29, 88]. In general, users may be subject to censorship by government or private enterprise entities in control of the network infrastructure that underlies (a subset of) the Internet [100, 96]. In particular, there are situations where a government may implement censorship to block traffic from/to other countries [22]. For example, as of March 2016, it has been estimated that the so-called "Great Firewall of China" blocks (or partially blocks) 7 out of the top 10 non-Chinese websites (70%), and in total, 135 out of 927 Alexa Top Domains (14.6%)[1]. In response, researchers are called to design a communication service to connect censored people (i.e., people inside the censored area) with censored Internet websites [89]. A typical approach to circumvent censorship is to use relay servers, such as VPN services [125], or Tor nodes [52], in order to provide indirect access to blocked Internet services.

[1] `https://en.greatfire.org/`

However, censors are able to block many of these systems by simply discovering and black-listing the IP addresses of relay servers [132, 162]. Moreover, censors monitor, filter, trace, and block the Internet traffic by leveraging several techniques, such as Domain Name System (DNS) hijacking [133], active probing attack [107], insider attacks [149] and Deep Packet Inspection (DPI) [49].

To overcome these censorship techniques, in this chapter, we propose SEnD, a Social network friendship Enhanced Decentralized censorship circumvention system. By leveraging IPOP (IP-over-P2P) tunnels [90], SEnD builds upon an overlay where users have private IP channels to their social friends [91], enabling virtual private network communication among social peers (even through NATs and firewalls). Figure 5.1 gives a birds-eye view of the basic concept of SEnD. In Figure 5.1, the physical network level is at the top, which illustrates a physical distribution of endpoint devices and the links among them. Moreover, due to censorship, the devices inside the censored area cannot access to the blocked services. At the bottom part of Figure 1, we describe the social relations among people who own the devices inside and outside the censored area. People are able to communicate with their friends through peer-to-peer VPN links by leveraging IPOP. In SEnD, we define the following three types of nodes: *requester*, *assister* and *supporter*. A requester represents a node who is located inside a censored area, and wants to access to blocked services. A supporter represents a node who is located outside a censored area, and is willing to help requesters to bypass the censorship (i.e., has intention to serve as a proxy node). Note that, it may occur that no supporters are available in the direct neighbors of a requester. In this case, the requester is able to discover a remote supporter by sending query messages to assisters who are the common friends of the requester and the supporter. Considering that people are more willing to help peers that they are familiar with [57], a requester can only discover a supporter within two social hops.

Note that, in SEnD, requesters try to discover and establish links with supporters on demand, as a function of their usage patterns. In Figure 5.1, r_2 is a requester and he utilizes one of his direct friends (i.e., s_2) as a relay server to access to blocked servers. In addition, r_1 is another requester. In this case, r_1 is able to discover remote supporters by sending query messages to assisters (e.g., a_4). Note that, as shown in Figure 5.1, a_4 is one of direct friends of r_1. He only delivers the discovering messages to possible supporters (e.g., s_1), and does not forward Internet traffic between the requester and the supporter. However, none of her friends are located in the uncensored area. In this case, r_1 sends a query message to a assister (i.e., a_4) to request a 2-hop supporter (i.e., s_1). If s_1 is available, r_1 will establish a direct link with s_1 to bypass censorship. We argue in Section 5.1 that this distributed and social-based censorship bypassing mechanism makes the system resilient

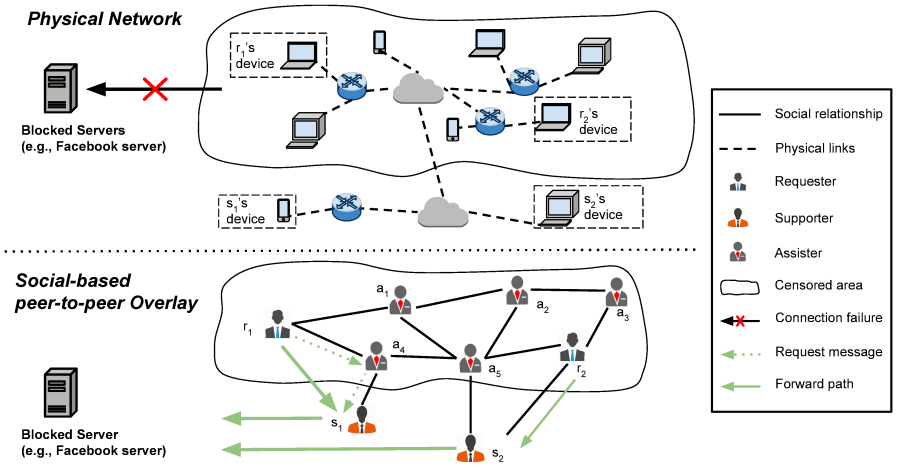


Figure 5.1: Overview of SEnD. Overlay links indicate social relationship. For example, because r_1 and a_4 are friends, they can communicate with each other in a P2P fashion through the social overlay.

to censorship techniques. In order to achieve load balance, requesters use random walk routing to discover and select supporters through assisters.

Contribution: To the best of our knowledge, SEnD is the first decentralized social-based censorship circumvention system where a requester inside censored area discovers a proxy node (i.e., supporter) in an on-demand way. SEnD can deliver improved resiliency to current censorship techniques by:

- Constraining the scope of advertisement/discovery of services to a 2-hop social network radius (i.e., friends or friends-of-friends).
- Tunneling traffic in a peer-to-peer fashion with encrypted, encapsulated messages without passing through a centralized gateway.
- Leveraging trusted social relationship to discover/establish connections between requesters and supporters.
- Allowing traffic proxies to use dynamically-allocated NATed endpoints that are difficult to systematically predict or probe.

To evaluate the connectivity between requesters and supporters, we perform simulations based on synthetic network graphs, using parameters to drive the simulation that are derived from outcomes of a user survey. Moreover, we carry on experiments based on a prototype implementation to evaluate ICMP and page load latency overheads of the proposed system.

5.1 Background

In this section, we discuss the backgrounds of this chapter. In particular, we first introduce how we construct a social-based overlay based on peer-to-peer virtual private links. Then we discuss the current censorship techniques used by censors, and how SEnD is resilient to these techniques.

5.1.1 Peer-to-peer virtual private network (P2PVPNs)

Recently, P2P virtual private networks (P2PVPNs) such as Hamachi[2], IPOP [91, 90] have become popular decentralized alternatives to centralized VPNs. In Hamachi, centralized Simple Traversal of UDP through NATs (STUN)-like servers are used to enable NAT traversal and establish direct P2P connections among users [64]. However, censors are able to black-listing the IPs of these servers to block users from using Hamachi [145]. In IPOP, NAT traversal is not centralized because it uses existing nodes in the overlay to perform UDP hole punching for direct P2P connectivity [150]. In addition, IPOP leverages existing social infrastructures to enable VPNs. SEnD is built upon an IPOP overlay where users have private IP connections to their social friends, which consequently enables trusted social communication (even through NATs and firewalls). As shown in Figure 5.1, with SEnD, friends are able to communicate directly.

5.1.2 Censorship techniques

In order to control the content of the Internet, dictatorships monitor, filter, trace, and block the Internet traffic by leveraging several technologies, such as IP address blocking, Domain Name System (DNS) hijacking, active probing attacks, insider attacks and Deep Packet Inspection (DPI). In this section, we first introduce these technologies, and hence discuss how SEnD is resilient to these censorship techniques.

IP address blocking: To bypass a censorship, an intuitive and simple method is to leverage a server-based virtual private network (VPN). Specifically, we list two different cases based on centralized servers: (1) users first connect to centralized proxy servers, send traffic to these rely servers directly[3]. (2) Instead of being proxy servers, centralized servers act as dictionary servers. These centralized dictionary servers assign proxies for each user who requests it [125]. These VPN servers behind public IP addresses can be easily targeted by censors. However, in SWIFT, proxies (i.e., supporters) reside behind NAT and dynamically create P2P link with assisters or requesters. In perspective of censorship authorities, these traffics are UDP encapsulated HTTP streams [146], which make censorship

[2] `https://secure.logmein.com/products/hamachi/`
[3] `http://www.evergreenvpn.com`

authority more difficult to detect. Although admitting that supporters are behind NAT, they are mapped with public IPs of NAT routers. If a censor blocks the IP address of a supporter, the connected requesters could easily discover other supporters through the overlay in a decentralized way; the supporter's public IP can be easily changed by requesting ISPs of supporters (e.g., restarting the online devices if the devices acquired public IP addresses based on Dynamic Host Configuration Protocol [76]).

DNS hijacking: DNS hijacking is the practice of subverting the resolution of DNS queries. Instead of returning a real IP address to a user, the malicious DNS server may respond a fake one, or just simply does not respond. Note that, DNS hijacking only affects areas where malicious DNS servers serve. In other words, DNS hijacking only happens in censored area. In SEnD, requesters establish links with supporters without translating of DNS servers. They negotiate with each other on top of the social-based overlay based on socket addresses. Moreover, supporters are able to reach censored services without effects of DNS hijacking, considering following two reasons: (1) a malicious DNS server controlled by a censored only could affect limited areas (censored areas); (2) supporters and blocked services are in the uncensored areas.

Active probing attacks: In active probing attack, a censor-controled user issues its own connections to a suspected circumvention server. If the server responds using a prohibited protocol, then the censor takes a blocking action, such as IP address black-listing. If the circumvention server does not incorporate access control mechanisms or techniques to distinguish the censor's probes from normal user connections, the censor can reliably identify and block it [58]. Note that, in order to be connected by users, the circumvention servers mentioned above are with public IP addresses. Howerver, in SEnD, supporters are social peers. According to [60], most of users surf the Internet with private IP address. Therefore, supporters are behind Network Address Translation (NAT), which makes harder for them to be actively probed. This is because that, NATs will typically drop "unsolicited" messages - incoming messages from computers they have not sent outgoing messages to.

Insider attacks: Another particularly powerful approach to enumerating relay servers is the insider attacks — the censor colludes with corrupt users (relays) to discover and shut down relays (users); the censor can further amplify the attack by deploying a large number of Sybils to accelerate the discovery of relays/users. In SEnD, supporters are social friends (or friends of friends) of censored users. Supporters only establish connections to requesters they trust. Therefore, SEnD is able to mitigate the insider attack effectively.

Deep Packet Inspection (DPI): DPI technology is increasingly used by censoring countries to filter the free flow of information. SEnD is based on a social overlay where a requester could easily discover a supporter, and

hence establish a link to the supporter. Traffics over SEnD are encapsulated with Datagram Transport Layer Security (DTLS), which makes SEnD resilient against DPI. Moreover, in this chapter, we mainly focus on that requesters discovering supporters in a decentralized and on-demand way. Note that, SEnD is able to adopt approaches (e.g., TRIST [39], GoHop [155]) that are more resilient to DPI attack (or other attacks aiming at analyzing pattern of traffics) as the data transmission channel.

5.2 Related work

In this section, we discuss related works in the topics of anonymity system and Censorship circumvention system respectively.

Anonymity system: Tor [52] is an anonymous communication system. A client builds a circuit by choosing several proxy servers. Tor system relies on a set of directory servers to distribute the information of proxy servers to clients. However, the centralized directory servers can be blocked by censorship. In addition, proxy servers themselves might be malicious. To overcome these limitations, researchers proposed several Tor extensions [156] bases on user reputation [154], or social trust [171, 121] to assist clients to choose proxy servers. These Tor-based approaches mainly address the problem of anonymous communication. They require sending packets through a sequence of relays on dilated paths for strong anonymity protection. However, it causes network overheads, in particular increased end-to-end latency, because Tor extensions provide anonymity to all hops of circuit. In contrast, SEnD aims to provide censorship circumvention to users. As we discuss in Section 5.5.2, since communications in SEnD are relayed by a single proxy server (although a requester might need multiple social hops to discover the proxy server), the latency of SEnD is smaller than the one in Tor-based approaches. In addition, clients in these Tor-based systems choose proxy servers only considering the trust level. In SEnD, we take another dimension (i.e., current network overheads on available proxy servers) into consideration.

Censorship circumvention system: VPN Gate [125] uses techniques such as innocent IP mixing and collaborative spy detection to enhance the resistance against censorship. It shows that it achieves effective censorship resistant against China's Great Firewall (GFW). But VPN Gate still may have a substantial portion of VPN servers blocked by the censorship, since it uses central servers to disseminate information of proxy servers. Authors in [77] propose CacheBrowser, a client-side censorship circumvention application that is able to access censored content stored in CDN servers. It allows users to bypass censors without involvement of third party volunteers. CacheBrowser, however, cannot access contents that are not cached in CDN servers. Collage [30, 31] leverages the available user generated contented

server (e.g., Flickr and Youtube) to embed hidden messages covertly into traffic, making it difficult for a censor to block the contents. However, servers storing hidden information also could be censored by authorities. Actually, the servers (i.e., Facebook, Youtube, Twitter and Flickr) mentioned in [30] are all blocked in China. This approach is limited in the bandwidth they can support, but might be useful as rendezvous schemes. FreeWave [79] works by modulating a client's Internet traffic inside the acoustic signals that are carried over VoIP connections. It does not mention how a client bootstraps to initiate communication with proxy servers.

In Kaleidoscope [139], proxy servers disseminate their addresses over a social network. Note that, targets of the dissemination may not be the one who requests proxy servers. As a result, this non-target dissemination mechanism results in exposing proxy servers to censors. In SEnD, a requester searches proxy servers on demand, based on its requests to Internet services. In other words, a node in SEnD only requests a supporter when it is needed, dynamically, which makes it harder to expose proxy servers to authorities. In addition, Kaleidoscope does not consider network overheads when a requester selects a proxy server, which results in a problem that network overheads are not uniformly distributed (i.e., massive number of connections are assembled on a set of proxies servers). In SEnD, we have the following two approaches to mitigate this drawback: (1) a requester uses random walk routing algorithm to discover/select a supporter; (2) a supporter can set a number of maximum connections from his friends (or friends of friends). Moreover, Kaleidoscope does not evaluate the connectivity between requesters and supporters.

5.3 Our solution for censorship circumvention: SEnD

In this section, we present SEnD — a decentralized social-based censorship circumvention system. For ease of exposition, we first introduce the overview of SEnD (Section 5.3.1). Then, we describe random walk routing algorithm that requesters use for discovering available supporters (Section 5.3.2). Finally, we discuss the implementation details of requester, assister, and supporter, respectively (Section 5.3.3).

5.3.1 Overview

In SEnD, nodes cooperate to bypass censorship over a social-based overlay network. The overlay nodes are categorized into three types: *requester*, *supporter* and *assister*. Specifically, a *requester* is a user who resides in a censored area and wants to access blocked services. A *supporter* node is a user outside censored network, and is willing to help requesters to bypass

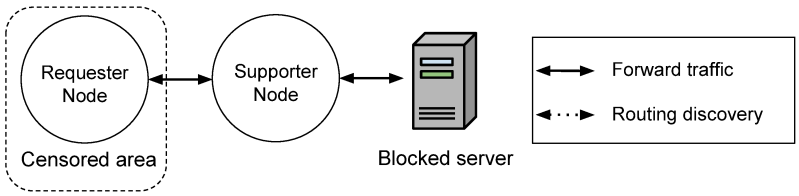
censorship. A supporter receives a requester's traffic from their direct link and injects the traffic to the public network. In other words, supporters serve as proxy servers for requesters. Note that, if a requester is not able to reach available supporters in his direct neighbors, the requester node will need a *assister* node to discover a supporter in 2-social hops. Therefore, assisters are overlay nodes that only deliver discovery message to assist requesters to discover supporters. Once a requester discovers a supporter, they will establish a direct link to each other. Assisters do not transfer Internet traffic between requesters and supporters. Note that, in SEnD, requesters only can discover supporters within 2-social hops. This is because that supporters are more willing to help their close friends, instead of strangers.

Figure 5.2 shows different situations of bypassing a censorship. In particular, Figure 5.2a describes the requester access blocked server through the supporter who is his direct friend. Figure 5.2b and Figure 5.2c show a situation where the requester has no direct supporters. In this case, the requester first needs to use a routing algorithm to discover a supporter based on social overlay (routing discovery channel), and hence establishes a link (forward traffic channel) with the supporter directly. In Figure 5.2b, the assister who helps the requester to discover a supporter is located in the uncensored area. Figure 5.2c shows both requester and assister are in the censored area. Although each node functions differently depending on varying circumstances, in SEnD, nodes are able to function as any type of nodes simultaneously.

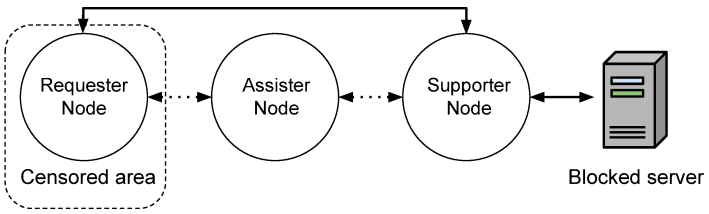
In SEnD, in order to limit the number of connected requesters, supporters are able to set a 1-hop/2-hop Maximum Number of Affordable Connections (MNAC). To be more specific, 1-hop MNAC represents the maximum connections a supporter could accept from requesters who are his 1-hop friends. 2-hop MNAC is the maximum connections from 2-hop requesters. In addition, in the following, we use 1-hop/2-hop Number of Available Connections (NAC) to represent a supporter's number of current connections from 1-hop/2-hop requesters. Note that, NAC of a supporter is no more than his MNAC.

5.3.2 Route discovery protocol

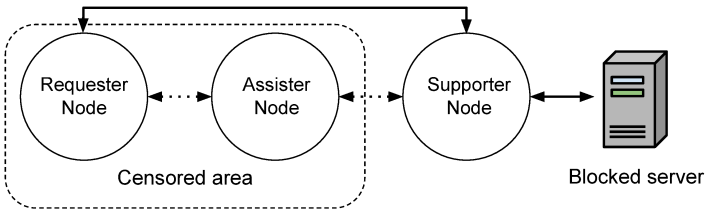
In this section, we introduce the route discovery mechanism of SEnD. As we discussed in Section 5.3.1, a requester needs protocols to discover a distant supporter if no directly-connected supporter is available. Because of the decentralized and unstructured manner of social affinity network, the overlay topology cannot be mapped to a tree-like structure, hence routing protocols that rely on structured identifiers (e.g., IP addresses) cannot directly apply. Instead, to discover a supporter, unstructured policies based on flooding or random walks need to be considered. We employ a random walk based discovery algorithm to find a route, through the social graph, to a supporter.



(a) The supporter is the requester's 1-hop friends; no assister involved.



(b) The supporter is the requester's 2-hop friends; the assister is in the uncensored area.



(c) The supporter is the requester's 2-hop friends; the assister is in the censored area.

Figure 5.2: Federation of requester, supporter, assister nodes that across overlay network to bypass a censorship.

In particular, a requester uses random walk to discover a supporter, and then establishes a direct link with the discovered supporter. Note that, in general, route discovery can in principle be used to find a multi-hop, source-routed path to a supported who is n-hops away from the requester. However, considering the requirements of our system – the low likelihood that users are willing to provide proxying, or to query for proxies beyond the scope of a friend-of-a-friend, we focus on 2-hop discovery as a special case.

Algorithm 7 shows the procedure of discovering an available supporter. In order to balance the load (i.e., number of connections on nodes) in a network, a requester use random walk to discover its 1-hop/2-hop supporter(s). In particular, a requester first randomly selects a direct neighbor. If the selected neighbor is in uncensored area, and his NAC is larger than 0, the requester will choose this neighbor as his supporter (line 6 in Algorithm 7). If all direct neighbors of the requester can not be a supporter (neighbors are in the censored area, or neighbors' NAC are zero), the requester will leverage a assister (one of his neighbors) to send a discovery message to his

2-hop neighbors. To prevent cycling, the message records the identifier of requester and does not sends packet to whom already visited (line 13 in Algorithm 7). Once a supporter accepts a request from a 1-hop (or 2-hop) requester, the 1-hop (or 2-hop) NAC of the supporter will minus one (line 7 and line 18 in Algorithm 7).

Algorithm 7 Discovery Phase in Routing Algorithm.

```
1: procedure DiscoverSupporter(G,censoredList,unCensoredList)
       Input: G is the initial network that includes all the nodes and their relationships; censoredList is a set of nodes in the censored area; uncensoredList is a set of nodes in the uncensored area.
2:     neighbors ← requester.neighbors()
3:     random.shuffle(neighbors)
4:     for each neighbor ∈ neighbors do
5:         nac1hop ← neighbor.availConn1Hop()
6:         if neighbor.isUncensored() & nac1hop > 0 then
7:             neighbor.availConn1Hop() ← nac1hop - 1
8:             return neighbor
9:         end if
10:    end for
11:    for each neighbor ∈ neighbors do
12:        2HopNeighbors ← neighbor.neighbors()
13:        2HopNeighbors.remove(node)
14:        random.shuffle(2HopNeighbors)
15:        for each 2HopNeighbor ∈ 2HopNeighbors do
16:            nac2hop ← 2HopNeighbor.availConn2Hop()
17:            if 2HopNeighbor.isUncensored() & 2HopNeighbor.availConn2Hop() > 0 then
18:                2HopNeighbor.availConn2Hop() ← nac2hop -1
19:                return 2HopNeighbor
20:            end if
21:        end for
22:    end for
23:    return ∅
24: end procedure
```

Upon reaching an available supporter, the supporter sends back an ack message to the requester along with the path that dilvers the request message. After the requester receives the ack message from the supporter, he will establish a direct link to the supporter. The requester uses this link to forward the blocked Internet traffic. Note that, this link does not represent that the requester and the supporter are direct friends.

5.3.3 SEnD Implementation

In this subsection, we detail the implementation of SEnD. In particular, we introduce the functionalities of a requester node, a assister node, and a supporter node, respectively.

Requester

Requesters' traffic destined to censored services are routed through the overlay network to their supporters, while traffic to uncensored services is not sent via the overlay. As shown in Figure 5.3, a requester has two network interfaces: *peth*0 and *veth*0. *peth*0 is connected to the default gateway and leads traffics to the Internet, while *veth*0 is a virtual network interface connected to the SEnD overlay. The overlay uses SHA-1 as unique identifiers for every node.

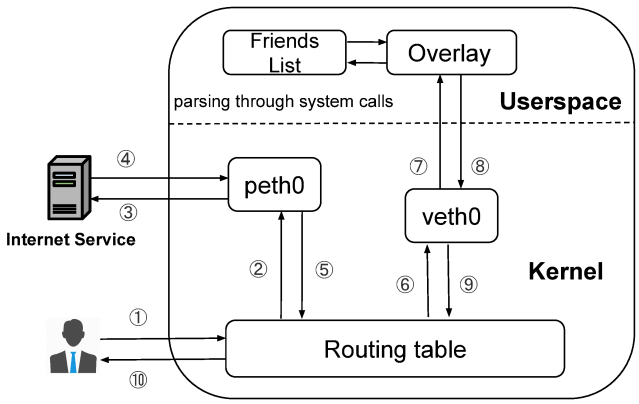


Figure 5.3: Requester node structure.

Since requesters are in a censored area, the "uncensored" traffic can be directly sent to the Internet through *peth*0 (action (1), (2) and (3) in Figure 5.3). In contrast, "censored" traffic is blocked by censorship (requesters can not receive the response from the Internet). To distinguish these two cases, SEnD adds the IP address or domain name of the service to the blocked service list in the O/S routing table. Configuring the routing table entries can be done in different ways: this can be potentially automated by parsing the reply (or lack of reply) from the censorship agent (action (4) and (5) in Figure 5.3), or, alternatively, a SEnD user can also configure a blocked service list manually, based on their knowledge of services that are blocked. Such configuration can be done for DNS names, or IP addresses – resolving DNS names requires an extra step, which is to request the supporter in an uncensored domain to resolve the name on behalf of the requester.

Then, all subsequent traffic destined to the IP addresses in the blocked service list are streamed to the overlay network to supporters through *veth*0

(action (6) and (7) in Figure 5.3). Note that, before forwarding traffic to a supporter, a requester first need to discover an available supporter by using a route discovery algorithm (as discussed in Section 5.3.2), and hence establish a direct link with the supporter. Streamed packets are encapsulated and encrypted by the overlay and sent to supporters through dynamically-established P2P tunnels.

Assister

In SEnD, assisters only route discovery messages from requesters to supporters, and do not forward the Internet traffic. Figure 5.4 shows the structure of a assister node. Once a assister receives a route discovery request (action (1) in Figure 5.4) from his neighbor (a requester), it sends the message to one of its neighbors (action (2), (3) and (4) in Figure 5.4) to inquire whether this neighbor could act as a supporter for the requester. The assister sends the information of a supporter (if available) back to the requester. Subsequently, the requester establishes a link with the supporter (if available), and they use this link to forward the Internet traffic.

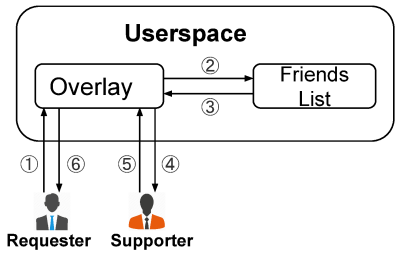


Figure 5.4: Assister node structure; only userspace are shown.

Supporter

Figure 5.5 shows the structure of a supporter node in SEnD. Once a requester receives traffic from the overlay, it decrypts, decapsulates, and places the traffic into physical network. In particular, upon receiving the packet, the supporter decapsulates SEnD header and place the packet into physical network (action (1), (2) and (3) in Figure 5.5). This process can be done by inspecting the transport address of TCP/UDP stream and open a socket with given transport address. Note that, the IP address of the packets is a private address of overlay. Thus, we need to change IP address to *peth0* of the supporter node along with its checksum. After the packet is put to the peth0, the packet traverses the public, uncensored network, reaching destination server and replies with served contents (action (4), (5) and (6) in Figure 5.5). This takes a reverse path from supporter node to requester node (action (7) and (8) in Figure 5.5).

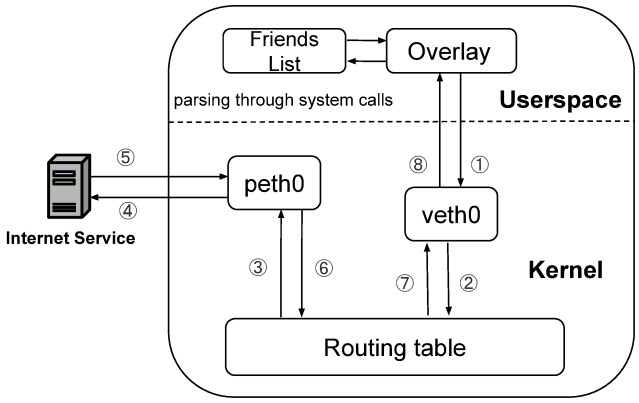


Figure 5.5: Supporter node structure.

Establishing links

The links connecting the requester, assister, and supporter nodes described earlier are P2P tunnels established through the aid of a messaging protocol service (e.g., XMPP) to exchange (possibly NATed) IP:port endpoints discovered through STUN services, as described in [90]. Applying censorship actions to the process of establishing links is difficult, due to a number of reasons. First, a variety of distributed messaging/chat services can be used to bootstrap links; in principle, the social overlay itself can be used to deliver these messages. Second, a variety of STUN services can also be used to discover endpoints; STUN servers are stateless and used for many P2P applications to discover NAT-mapped endpoints – there is no information associated with a STUN request that can be associated with the intent to create such an overlay link. Third, links are encrypted and use dynamically-allocated NATed endpoints at each peer; deep-packet inspection and traffic probing/identification based on IP:port endpoints is no longer practical.

5.4 Simulation setup

In this section, we describe the simulation setup used in our evaluation. The goal of the simulation is to evaluate the extent to which SEnD users can successfully discover supporter nodes that can acts as proxies for censored services. The setup is based on a graph simulation framework driven by synthetic graphs that are constructed based on parameters derived from social network characteristics and by results from a questionnaire that the authors have used to gather information from social network users. Specifically, in Section 5.4.1, we describe the questionnaires we carry out both inside and outside China. In Section 5.4.2, we describe how we construct

synthetic networks based on the parameters gathered from the questionnaires. Section 5.4.3 discusses the baseline metrics for the simulation.

5.4.1 questionnaires

We have conducted a survey of a sample of social network users within China (representing requesters potentially subject to censorship to a subset of Internet services) and outside China (representing potential supporters). The purpose of this study is to help us estimate parameters used to drive the simulation-based experiments – a large-scale longitudinal user study is beyond the scope of this chapter, but we wanted to gain insights into realistic parameters that are able to be used in our evaluation.

questionnaire inside China

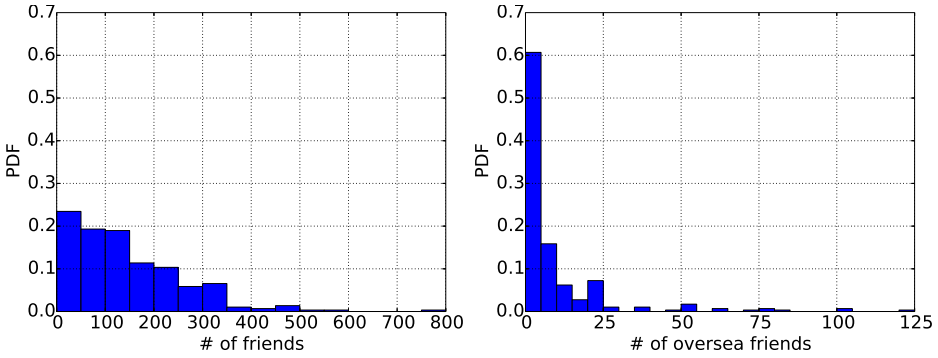
We first distribute the questionnaires inside China, in order to estimate the following three parameters: (1) the proportion of people who have demand for access to blocked services; (2) the average number of friends in a popular social network[4]; (3) the average number of friends whose current location are outside of China, which we refer to as "number of overseas friends". The questionnaire was administered in two ways: (1) we randomly selected passers-by in a supermarket and a shopping mall in Jinan, Shandong Province, China, and kindly requested them to answer the questionnaire in exchange of a small gift; (2) with the help of lecturers in the universities located in Jinan, Shanghai, and Qingdao, we recruited university students and kindly requested them to fill in the questions on an online survey website in exchange of a small gift. We distribute the questionnaires among 320 people (104 males, 216 females), and these participants are in the age range of 18 to 60.

The outcome of the questionnaires shows that there is a significant interest in accessing blocked services - 90.1% of the participants expressed demand for accessing blocked services (e.g., Facebook, Instagram). This result indicates the significance of proposing a censorship circumvention system that is resilient to the current censorship techniques.

Figure 5.6 shows the distributions of number of friends and number of overseas friends. These follow power-law distributions. The average number of friends in Wechat is 141, while the average number of overseas friends is 9. This indicates that 93.6% friendship of the participants are linked to peers inside China. As a point of comparison, the authors in [147] show that, if a country partitioned from the whole social network graph, on average 84.2% percent of edges will remain within the country. Although there is a difference between these two numbers, the survey results are not far from

[4]The questionnaires are based on Wechat because Wechat is the most popular/successful social network in China. Wechat is available at `http://www.wechat.com`

the values observed in related work; furthermore, a larger percentage of within-Country friends/edges is a conservative approach, since it leads to a smaller pool of potential supporters.



(a) # of friends (b) # of oversea friends

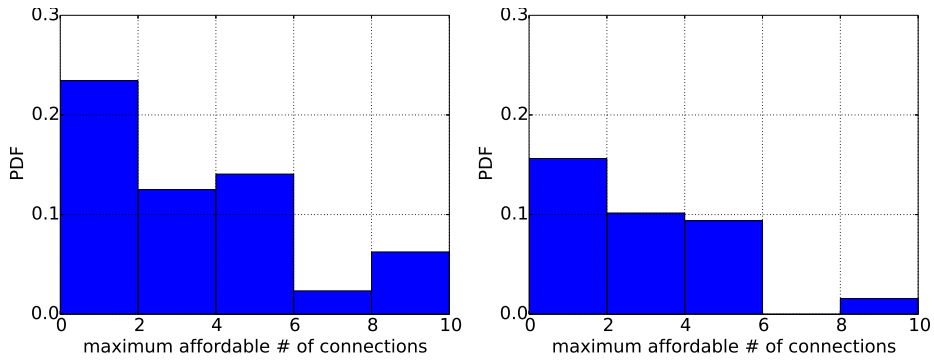
Figure 5.6: Distribution of number of friends/oversea friends. The y-axis represents probability distribution function (PDF).

questionnaire outside China

We also distributed questionnaires to 128 Chinese people who live outside China (e.g., Italy, U.K., U.S.A, Japan and Australia). In this questionnaire, we inquire the 1-hop and the 2-hop Maximum Number of Affordable Connections (MNAC) of these participants. For example, if Alice's 1-hop MNAC is 0, it means Alice does not want to be a supporter to her direct neighbors.

The results show that 93.8% participants are willing to help their direct friends to bypass the censorship, while only 50% participants are willing to help their friends of friends to do so. One important factor that influences the performance of SEnD is MNAC of overseas users. MNAC determines the probability that a censored node is able to discover an available friend (or friend of friends). Normally, a overseas user cannot accept unlimited connections, otherwise it will dry out his Internet bandwidth. Therefore, for the users whose answers are accepting unlimited connections, we assume their MNAC is 10. Figure 5.7 shows the distribution of 1-hop/2-hop MNAC. The average MNAC of 1-hop friends is 5.8, while the average MNAC of 2-hop friends is 2.5.

In the following section, we use metrics (distribution of number of friends, distributions of number of overseas friends, distribution of 1-hop MNAC, and distribution of 2-hop MNAC) extraced from the questionnaire responses to construct synthetic networks.



(a) 1-hop MNAC (b) 2-hop MNAC

Figure 5.7: Distribution of MNAC of participants who are willing to act as proxy servers. The y-axis represents probability distribution function (PDF).

5.4.2 Dataset

We simulate and analyze the connectivity of the proposed censorship circumvention system based on a synthetic network with realistic parameters. This section describes how we construct the synthetic datasets used in our simulation. In particular, in Section 5.4.2, we first describe how we construct an initial network. Section 5.4.2 describes how we generate a censored area and an uncensored area. Finally, in Section 5.4.2, we describe how we trim excess supporters for nodes inside censored area. Note that, the dataset construction and the simulation are operated with the NetworkX graphing library[5], which is a Python package for examining complex networks.

Initial network

We generate a social network (including nodes and their friendships) with n nodes based on a modified Nearest Neighbor Model (MNNM) [134]. This model follows the observation that two people sharing a common friend are more likely to become friends. The average degree of the generated networks follows the parameters we described in Section 5.4.1.

Partitioned network

In order to get a censored area and an uncensored area, we partition the network into two parts. For each node in the network we get from Section 5.4.2, we use *Loc-Generation* algorithm (introduced in [50]) to assign location information. The location information is based on the Cartesian coordinate system. Then based on the location of each node, we cut a certain area

[5] `http://networkx.github.io/`

to simulate a censorship region. *Loc-Generation* algorithm is based on the following observation [15]: the shorter is the distance among two nodes, the higher is the probability for them to be friends.

The censored area that includes w nodes is defined as: the smallest circular area that has the center in the origin and includes w nodes. We get two subnetworks from the whole network after performing partition: the censored area, and the uncensored area. The size of the censored area can be controlled by adjusting the number of nodes in the area.

Trim excess supporters

Section 5.4.1 discussed that, 93.6% friendship of Chinese people is linked to peers inside China. However, due to the limitations of the synthetic algorithms, after partitioning, the range of proportions of remaining edges in censored area are from around 40% to around 70%. Note that, proportion of censored area is 18%, which similar to the percentage of people in China over the entire world population. Therefore, in this section, we propose a *Supporter-Trimming* Algorithm to remove the excess oversea friends. Algorithm 8 shows the details.

Algorithm 8 Trim Excess Supporter.

1: **procedure** SUPPORTER-TRIMMING(G,*censoredList*,*unCensoredList*)
Input: G is the initial network that includes all the nodes and their relationships; *censoredList* is a set of nodes in the censored area; *uncensoredList* is a set of nodes in the uncensored area.
2: **for each** $node \in censoredList$ **do**
3: $NumberOverseasFriends \leftarrow$ generateOverseasNumber()
4: $neighbors \leftarrow node$.neighbors($G$)
5: $censoredNeighbors$, $uncensoredNeighbors \leftarrow$ classify($neighbors, CensoredList$, $UncensoredList$)
6: **if** len($uncensoredNeighbors$) > $NumberOverseasFriends$ **then**
7: $tuningTimes \leftarrow$ len($uncensoredNeighbors$) - $NumberOverseasFriends$
8: $uncensoredNeighbors$.sort()
9: **while** $tuningTimes > 0$ **do**
10: $uncensoredNeighbor \leftarrow uncensoredNeighbors$.pop()
11: G.removeEdge($node$, $uncensoredNeighbor$)
12: $censoredNode \leftarrow node$.getNeighbor($censoredList$)
13: G.addEdgeBasedNearestNeighborModel($node$)
14: $tuningTimes \leftarrow tuningTimes$ - 1
15: **end while**
16: **end if**
17: **end for**
18: return G, *censoredList*, *unCensoredList*
19: **end procedure**

For each node in the censored area (i.e., *censoredList*), the algorithm trims its overseas friends. Based on the distribution of number of overseas friends gathered from the questionnaire (we discussed in Section 5.4.1), algorithm first generates a reference number of overseas friends to a node (we use "reference number" to represent the number generated by the algorithm based on the realistic distribution) (line 3 in Algorithm 8). If the node's current number of overseas friends is larger than the reference number, algorithm will adjust this number to the reference one. In particular, the algorithm removes links between the node and some of its neighbors in the uncensored area (line 11 in Algorithm 8). Then, based on the modified Nearest Neighbor Model, algorithm adds neighbors (inside censored area) to the node (line 13 in Algorithm 8). Afterwards, we generate a network with censored and uncensored areas based on the realistic metrics.

5.4.3 Selection of baseline parameters

In this section, we discuss the baseline parameters in our simulation. As we discussed in Section 5.4.1 and Section 5.4.1, the average number of overseas friends is 9; the average 1-hop MNAC is 5.8; the average 2-hop MNAC is 2.5. In addition, we consider the baseline of the proportion of censored area is 18%, that is very close to the one of Chinese population over the whole worlds. Moreover, since the hardware limitations of our clusters used for this evaluation, we cannot run this evaluation on a network with nodes similar with the whole world population. Therefore, we set the baseline of number of initial nodes to 10 million. Nevertheless, in the following evaluation, we will demonstrate SEnD is scalable and is able to deploy in a network with larger number of initial nodes.

5.5 Evaluation

In this section, we first carry out an extensive simulation in order to analyze the connectivity between requesters and supporters. Then we assess the latency of the proposed system through an experiment-based evaluation.

5.5.1 Simulations

In our simulation, we perform a sensitivity analysis of successful connection rate (SCR) by varying each parameter. The SCR of a network is defined here as the number of *connected censored nodes* divided by the number of *total censored nodes*. We denote *connected censored nodes* as requesters who are able to discover 1- or 2-hop supporters whose NAC is greater than 0. Since the dataset does not provide temporal information about online and offline nodes, we assume that all the nodes are online users in our simulation. Nevertheless, we leverage the concept of "personal overlay" [62] to

mitigate the side effect of offline nodes (Section 5.6). Instead of online and offline nodes, we consider active requesters and inactive requesters in our simulation, given that not all the requesters use the SEnD service simultaneously (even when they are online). For the following metrics, we report the average of the results obtained across 100 simulation runs. In the graph reported in this chapter, error bar indicates confidence interval (95%).

Impact of the proportion of concurrent active requesters

Figure 5.8 shows how the observed SCR varies as a function of the proportion of active requesters. An active requesters is a user in the censored region who actively seeks to access censored content. Other metrics of the network in this evaluation are shown in the caption of Figure 5.8. When the proportion of active requesters increases, the SCR of the system decreases. For example, when the system has less than 60% active requesters, almost all the active requesters are able to discover supporters to bypass censorship. On the other hand, when the system has 80% active requesters, only around 80% of them can use the system. This is because that, the total number of connections that overseas supporters are willing to provide is fixed. When the portion of concurrent active requesters is less than 60%, the number of requesters is less than the total number of MNAC provided by supporters. Considering that SEnD is based on the social-based overlay, almost all requesters are able to discover a supporters under this circumstance. When the portion of concurrent active requesters is larger than 60%, some of these active requesters cannot discover available supporters. As a result, larger portion of active requesters (i.e., larger than 60%) results in less SCR.

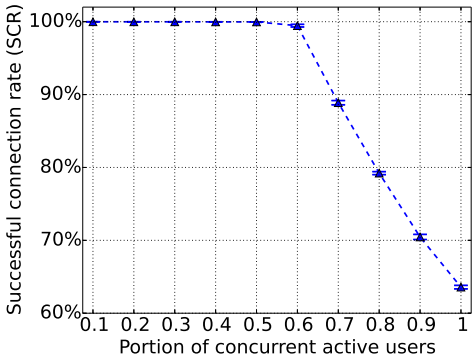


Figure 5.8: SCR in different proportion of active requesters. Average number of overseas friends is 9; average MNAC of 1-hop friends is 5.8; average MNAC of 2-hop friends is 2.5; proportion of censored area is 18%; number of nodes in initial network is 10 million.

Note that a proportion of concurrent active users of a system/service normally is less than 60%. For example, percentage of monthly Facebook

users that use it daily is 66.1% (updated March 2016[6]). Considering that the total registered users is larger than the number of monthly active users, and the number of daily active users is larger than the number of concurrent active users, it is expected that the proportion of concurrent active users of Facebook (i.e., concurrent active users over the total registered users) should be smaller than 66.1%. Another example is Steam, a digital distribution platform offering digital rights management, multi-player gaming and social networking services. The peak number of concurrent Steam users in 24th May, 2016, is around 11 million[7], which is 8.8% of the number of total registered Steam Users (125 million). Therefore, considering the result from Figure 5.8, with SEnD, an active requester is highly likely to discover an available supporter to bypass censorship. Nevertheless, we will discuss an approach to increase the number of supporters in Section 5.6.

In the following evaluation, we set the baseline of portion of concurrent active user to 80%. This is because that: (1) we carry on the following evaluations to show the SCR with different parameters. If the portion of concurrent active user is less than 60%, the the SCR is always near to 100%. In order to show the SCR evidently when other parameters change, we set the portion of concurrent active user to 80%. (2) In some extremely situations (e.g., dictators limit their citizen accessing the Internet contents drastically after political events), it may happens that large number of requesters rely on SEnD system simultaneously.

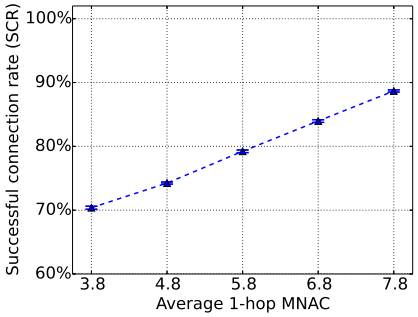
Different MNAC

Figure 5.9 shows the changes of SCR, as a function of varying MNAC of supporters. In particular, Figure 5.9a shows how 1-hop MNAC effects SCR, while Figure 5.9b shows how 2-hop MNAC effects SCR. Note that, other metrics used in this evaluation are shown in the caption of Figure 5.9.

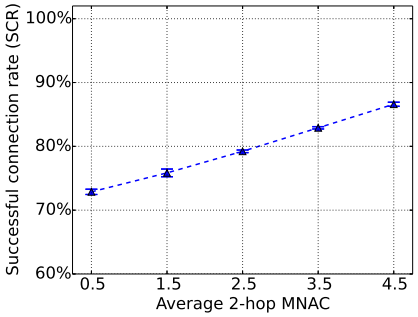
As Figure 5.9 shows, along with the increase of average 1- or 2-hop MNAC, the SCR increases linearly. This is because that higher 1- or 2-hop MNAC indicates larger number of connections that could be provided to requesters inside a censored area. More requesters discovering available supporters leads higher SCR.

In addition, the slope of the line in Figure 5.9a is steeper than the one in Figure 5.9b, in the situation where the intervals of x-axis and y-axis are same in both figures. This is because that requesters are more easily to find direct supporters (comparing to 2-social-hop supporters), and hence consume 1-hop NAC of the supporters. Therefore, larger 1-hop MNAC leads higher SCR than 2-hop MNAC does; while smaller 1-hop MNAC leads lower SCR than 2-hop MNAC does. In other words, 1-hop MNAC has more influence on SCR than 2-hop MNAC.

[6] `https://zephoria.com/top-15-valuable-facebook-statistics/`
[7] `http://store.steampowered.com/stats/`



(a) Different MNAC of 1-hop supporters; average 2-hop MNAC is 2.5.

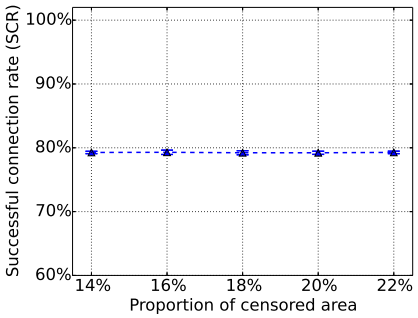


(b) Different MNAC of 2-hop supporters; average 1-hop MNAC is 5.8.

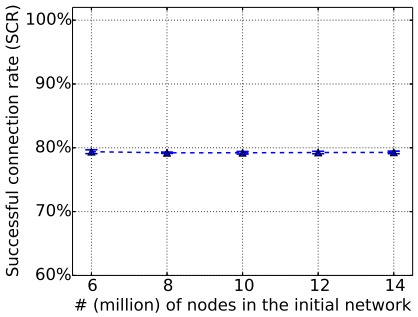
Figure 5.9: Different average MNAC of supporters. Average number of overseas friends is 9; proportion of censored area is 18%; number of nodes in initial network is 10 million; proportion of concurrent active users is 80%.

Different proportion of nodes in censored area

Figure 5.10a shows changes of SCR, considering different sizes of censored areas. Note that, other parameters used in this evaluation are shown in the caption of Figure 5.10.



(a) Different portion of censored area; total 10 million nodes in initial network.



(b) Different number of nodes in initial networks; proportion of censored area is 18%

Figure 5.10: Different number of nodes in censored area / initial networks. Average number of overseas friends is 9; proportion of censored area is 18%; number of nodes in initial network is 10 million; proportion of concurrent active users is 80%.

As we discussed in Section 5.4.3, the baseline of proportion of censored area is 18%. The SCR of SEnD fluctuates around 79% when the proportion of censored area ranges from 14% to 22%. In addition to the results shown in Figure 5.10a, we also carry out an simulation where the propor-

tion of censored area is 1% (other parameters are same), which is a similar number to the Iran's population over the entire world population. Iran is another country where the people access the Internet controlled by censors. The result shows that, with 1% proportion of censored area, the successful connection rate is also around 79%. The larger portion of censored area indicates larger number of requesters who need to bypass censorship. It also indicates larger number of supporters located in uncensored area connecting to the requesters (since the average number of oversea friends are fixed), which results in larger number of NAC. When the portion of censored area increases, the additional 1- or 2-hop NAC provided by additional supporters are consumed by additional requesters. SCR of SEnD is independent from the size of censored area, if the number of overseas friends and 1- or 2-hop MNAC are fixed. As a result, the SCR of SEnD is stable at around 79% in different partition sizes.

Different number of nodes in initial network

In this section, we discuss the SCR of SEnD in networks with different initial nodes. Figure 5.10b shows changes of SCR in different network sizes. Note that, other parameters used in this evaluation are shown in the caption of Figure 5.10. When the number of nodes in the initial network increases, the SCR of SEnD fluctuates around 79%. The reason of this behavior is that, in SEnD, a requester uses random walk to discover a supporter within 2-social hops. The random walk mechanism is fully distributed, and does not rely on any central server. Therefore, even in a network with large number of nodes, the SCR of the network is stable, if other parameters (i.e., average number of oversea friends, 1- or 2-hop MNAC) does not change.

Since the hardware limitations of our clusters used for this evaluation, we cannot run this evaluation on a network with larger number of nodes. However, as the curve shown in Figure 5.10b, we believe SEnD is scalable, and is able to work in a network with even more number of nodes.

5.5.2 Experiments with a prototype

Having evaluated the ability to discover supporter nodes, we now carry out experiments to analyze a prototype that implements the core overlay virtual networking capability of SEnD. In particular, we run the requester node in Alibaba's ECS cloud (located in eastern China)[8], and the supporter node in Cloudlab (located outside China)[9]. The experiment assumes the requester has discovered and linked with the available supporter, and evaluates the ICMP latency and page load latency in accessing a censored resource. Note that, we report the results obtained across 50 experimental runs.

[8] `https://www.aliyun.com/product/ecs`
[9] `https://www.cloudlab.us`

ICMP latency

In order to analyze ICMP latencies of SEnD, we measure the following three latencies:

- *req*2*sup* is the ICMP latency between the requester and the supporter.
- *sup*2*ser* is the ICMP latency between the supporter and a blocked service (i.e., `http://www.google.com`).
- *req*2*ser* is the end-to-end ICMP latency from the requester to the blocked service. Note that, this latency includes the time when the ICMP packets traverse the SEnD overlay.

We report the minimum value, maximum value, median value and average value of *req*2*sup*, *sup*2*ser* and *req*2*ser* in Table 5.1.

	Min	Max	Mediam	Average
*req*2*sup*	189 ms	190 ms	190 ms	189.79 ms
*sup*2*ser*	70.9 ms	71.6 ms	71.2 ms	71.23 ms
*req*2*ser*	278 ms	281 ms	279 ms	278.89 ms

Table 5.1: Results of ICMP latency.

As we can see in Table 5.1, the *req*2*ser* is larger than the sum of *req*2*sup* and *sup*2*ser*. This is because when packets traverse the overlay to the public network or the other way around, there is overhead associated with overlay processing (17.87 ms on average, based on data in Table 5.1).

The overall latency that a requester access the blocked service with SEnD is 278.89 ms (average value). According to [33], the average latency between tor nodes are around 200 ms. In addition, in order to achieve anonymity, Tor defaults to circuits three nodes (i.e., an entry node, a middle node, and an exit node) between a requester and a web server. Therefore, the latency of Tor is expected to be significantly larger than the one of SEnD.

Page load latency

In addition to the latency of ICMP, we also test the page load latency of SEnD in this Section. We use the Google main page (i.e., `http://www.google.com`); and the size of the downloaded page is around 177 KB. Table 5.2 shows the minimum value, maximum value, median value and average value of page load latency from the requester to the server with SEnD.

The page load latency shown in Table 5.2 is not significantly small compared to regular browsing in U.S. This is because the server is in US and the client resides at opposite site of globe (i.e., China), and the traffic is relayed

	Min	Max	Mediam	Average
$req2ser$	3.345 s	3.851 s	3.392 s	3.447 s

Table 5.2: Results of page load latency.

by an overlay node in Utah (Cloudlab). Note that, SeND is a censorship circumvention tool that aims to provide availability of blocked services to the people in censored areas. Therefore, such page load latency is in a reasonable range.

5.6 Discussion

In this section, we first discuss the issue of offline nodes. Then, we point out the incentives for supporters to contribute in SEnD. Finally, we describe a possible method to increase available proxy servers.

Offline nodes: Users in SEnD discover peers and establish link to bypass a censorship based on a social-aware overlay [50] where people are able to communicate with their social friends directly. Social affinity topology churns dynamically. However, in the case of offline neighbors, nodes could still communicate with other online neighbors. Note that, 'offline' is not a permanent status. When a node becomes online, it will re-establish all the connections with other online social friends. In addition, authors in [62] leverage the concept of 'personal overlay' to connect all devices that belongs to one person. A personal overlay represents a node (a person) in our system. Although one individual device (e.g., smartphone) may be offline at some point, there can be other online devices (e.g., laptop, iPad) belonging to the same user who is online and connects to our system. Moreover, people tend to own more and more personal online devices [60], and the online time of devices tends to increase, considering that data plans become increasingly accessible to end users.

Incentives: In Tor [52] and VPN gate [125], many people provide their PCs and laptops to share their Internet bandwidth to help other strangers. SEnD builds upon an overlay where users have private IP connections to their social friends. Therefore, we believe people are more willing to provide their computing devices to help their social friends in SEnD. Another supportive evidence is that, 93.8% (50%) people in uncensored area are willing to help their friends (friends of friends) who locate in censored area (as the results from the questionnaires). Note that, a policy a supporter have is to accept up to N connections and then have a replacement policy based on timers - after an expiration period if a connection get to use too much bandwidth. In other words, users may not need to contribute a lot to the service, and the whole set of their friends can benefit from it.

Available proxy servers: With SEnD, users outside censored area could make use of their own online devices (e.g., laptop, PC, smartphone) to act as proxy servers. Nevertheless, users (either inside censored area or outside censored area) are able to leverage cloud servers (e.g., Amazon EC2, Alibaba ECS), that locate in uncensored area and are not blocked by censors, to be proxy servers. Therefore, a user even inside censored area, also could act as a supporter if he set up a cloud proxy server. As a result, number of available supporters could be larger than the results shown in the Section 5.5.

5.7 Summary

In this chapter, we propose SEnD, a fully-distributed censorship circumvention system builds upon an overlay where users have private IP connections to their social friends. In order to assess the effectiveness of SEnD, we perform extensive experiments and simulations. The results show that, with SEnD, users are always able to discover proxy servers in a fully distributed way, while keeping short latency to access blocked services, comparing Tor-based anonymous systems.

Chapter 6

Conclusions

Network failures and censorships result in web services/applications failures in partitioned area. Considering the prevalence of Internet-based applications, it is indeed a huge disturbance for affected people. In this dissertation, we presented effective techniques to improve network resiliency against failures and censorships. In this chapter, we summarize our contribution in Section 6.1, and discuss the possible future works in Section 6.2.

6.1 Summary of Contribution

This dissertation presented our contribution, which can be divided in two main parts.

Social-aware overlay and its bootstrapping method: we present a social-aware overlay, where users have private communication channels to their social friends, enabling virtual private network communication among social peers (even through NATs and firewalls). Our evaluation shows that, the connectivity (i.e., message routability) of social-aware overlay is better than other structured P2P overlays (i.e., Chord, SPROUT). In addition, the social-ware overlay maintains a smaller number of links. Moreover, we present a bootstrapping methods for the overlay users to bootstrap into the network. In particular, by leveraging close neighbors with Public IP address, a nodes could easily bootstrap and join the overlay.

Decentralized applications based on social-aware overlay: we present two representative applications based on social-aware overlay. First, we present SAND: a social-aware, network-failure resilient, and decentralized microblogging system. SAND is designed for (and hence is able to handle)

scenarios where massive correlated failures occur. Moreover, the delivery rates of SAND is significantly high (i.e., in SAND-SN, peers are able to effectively follow each others updates with 100% delivery rate). Then, we present SeND, a social network friendship enhanced decentralized system to circumvent censorships. With SEnD, users in an uncensored area can act as proxy servers for their social friends in a censored area, allowing them to bypass the censorship. We assessed the effectiveness of SEnD through extensive simulations based on a synthetic dataset, as well as experiments based on a prototype implementation.

6.2 Future Works

The work presented in this dissertation shows an initial effort. In the following we outline some directions for future works.

Comprehensive comparison of social-aware overlay and structured P2P overlays: In Chapter 2, we compared the topological connectivity and message routability of social-aware overlay with structured P2P overlays (i.e., Chord and SPROUT). However, this is a preliminary comparison. For example, we performed these evaluations on the small portion of network (less than 2.5% in synthetic dataset). One interesting future direction is to evaluate the connectivities of such overlays on the different size of networks (e.g., from 5% to 20%), in order to simulate the "large" countries (e.g., China, U.S.) are partitioned from the Internet. Another interesting direction is to consider other structured P2P overlays (e.g., Kademlia [116]) that are more resilient to node failures.

Comparison of bootstrapping methods for social-aware overlay: In Chapter 3, we introduced a methodology of how a node leverages his/her social friends to bootstrap into social-aware overlay. One future work could focus on the following direction: comparing the performance (i.e., bootstrapping rate) of proposed method with other available unstructured P2P bootstrapping methods.

Implementation of SAND and SeND: Based on social-aware overlay, in Chapter 4 and 5, we proposed a decentralized microblogging system, and a censorship circumvention system, respectively. In these two chapters, we introduced the methodology of these two systems, and analyzed the feasibility of the proposed systems based on extensive simulations. Although we have prototypes for these two system, it is still necessary to have the implementations with full functionality described in this thesis.

Bibliography

[1] Firechat. http://www.bbc.com/news/blogs-trending-29411159. [Last Accessed: 2014-11-28].

[2] Foursquare. https://foursquare.com/. [Last Accessed: 2014-05-27].

[3] Hamachi. https://secure.logmein.com/products/hamachi/. [Last Accessed: 2014-03-15].

[4] Networkx. http://networkx.github.io/. [Last Accessed: 2014-03-15].

[5] P2PVPN. https://p2pvpn.org/. [Last Accessed: 2015-05-04].

[6] Twitter. http://www.twitter.com/. [Last Accessed: 2015-09-28].

[7] Weibo. http://www.weibo.com/. [Last Accessed: 2016-09-28].

[8] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the tenth International Conference on Information and Knowledge Management, CIKM '2011*, pages 310–317. ACM, 2001.

[9] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media, LSM '2011*, pages 30–38. Association for Computational Linguistics, 2011.

[10] Philip E Agre. Real-time politics: The internet and the political process. *The Information Society*, 18(5):311–331, 2002.

[11] Ilhem Allagui and Johanne Kuebler. The arab spring & the role of icts— introduction. *International Journal of Communication*, 5:8, 2011.

[12] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in iran: A first look. In *The 3rd USENIX Workshop on Free and Open Communications on the Internet, FOCI '2013*, 2013.

[13] H Asthana and Ingemar J Cox. Pac'npost: a framework for a microblogging social network in an unstructured p2p network. In *Proceedings of the 21st International Conference on World Wide Web, WWW '2012*, pages 455–456. ACM, 2012.

[14] H Asthana and Ingemar J Cox. Trending topics in a peer-to-peer micro-blogging social network. In *2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing, P2P '2013*, pages 1–5. IEEE, 2013.

[15] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web, WWW '2010*, pages 61–70. ACM, 2010.

[16] Mirza Mansoor Baig and Hamid Gholamhosseini. Smart health monitoring systems: an overview of design and modeling. *Journal of Medical Systems*, 37(2):1–14, 2013.

[17] Peng Bao, Hua-Wei Shen, Junming Huang, and Xue-Qi Cheng. Popularity prediction in microblogging network: A case study on sina weibo. In *Proceedings of the 22Nd International Conference on World Wide Web, WWW '2013*, pages 177–178. ACM, 2013.

[18] Andrew Baum, Raymond Fleming, and Laura M Davidson. Natural disaster and technological catastrophe. *Environment and Behavior*, 15(3):333–354, 1983.

[19] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. Oversim: A flexible overlay network simulation framework. In *IEEE Global Internet Symposium, GI 2007*, pages 79–84. IEEE, 2007.

[20] Eva Bellin. Reconsidering the robustness of authoritarianism in the middle east: Lessons from the arab spring. *Comparative Politics*, 44(2):127–149, 2012.

[21] Serajul I Bhuiyan. Social media and its effectiveness in the political reform movement in egypt. *Middle East Media Educator*, 1(1):14–20, 2011.

[22] Constance Majomane Likonelo Bitso, Ina Fourie, Theodorus Jan Daniel Bothma, et al. Trends in transition from classical censorship to internet censorship: selected country overviews. 2013.

[23] Piers Blaikie, Terry Cannon, Ian Davis, and Ben Wisner. *At risk: natural hazards, people's vulnerability and disasters*. Routledge, 2014.

[24] Tamara Bonaci, Ryan Calo, and Howard Jay Chizeck. App stores for the brain: Privacy & security in brain-computer interfaces. In *2014 IEEE International Symposium on Ethics in Science, Technology and Engineering, ETHICS '2014*, pages 1–7. IEEE, 2014.

[25] Kerstin Borau, Carsten Ullrich, Jinjin Feng, and Ruimin Shen. Microblogging for language learning: Using twitter to train communicative and cultural competence. In *International Conference on Web-based Learning, ICWL '2009*, pages 78–87. Springer, 2009.

[26] Matko Bošnjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. Twitterecho: a distributed focused crawler to support open research with twitter data. In *Proceedings of the 21st International Conference on World Wide Web, WWW '2012*, pages 1233–1240. ACM, 2012.

[27] Axel Bruns, Tim Highfield, and Jean Burgess. The arab spring and social media audiences english and arabic twitter users and their networks. *American Behavioral Scientist*, 57(7):871–898, 2013.

[28] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems, SNS '2009*, pages 46–52. ACM, 2009.

[29] Sam Burnett and Nick Feamster. Making sense of internet censorship: A new frontier for internet measurement. *ACM SIGCOMM Computer Communication Review*, 43(3):84–89, 2013.

[30] Sam Burnett, Nick Feamster, and Santosh Vempala. Chipping away at censorship firewalls with user-generated content. In *USENIX Security Symposium*, pages 463–468. Washington, DC, 2010.

[31] Sam Burnett, Nick Feamster, and Santosh Vempala. Circumventing censorship with collage. *ACM SIGCOMM Computer Communication Review*, 41(4):471–472, 2011.

[32] Rajkumar Buyya, Saurabh Kumar Garg, and Rodrigo N Calheiros. Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *2011 International Conference on Cloud and Service Computing, CSC '2011*, pages 1–10. IEEE, 2011.

[33] Frank Cangialosi, Dave Levin, and Neil Spring. Ting: Measuring and exploiting latencies between all tor nodes. In *Proceedings of the*

2015 ACM Conference on Internet Measurement Conference, IMC '15, pages 289–302, 2015.

[34] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks. In *Proceedings of the 10th ACM SIGOPS European Workshop, SIGOPS '2002*, pages 140–145. ACM, 2002.

[35] Shaoyong Chen, Huanming Zhang, Min Lin, and Shuanghuan Lv. Comparision of microblogging service between sina weibo and twitter. In *International Conference on Computer Science and Network Technology, ICCSNT '2011*, volume 4, pages 2259–2263. IEEE, 2011.

[36] Yen-Yang Michael Chen, Anthony Accardi, Emre Kiciman, David A Patterson, Armando Fox, and Eric A Brewer. *Path-based failure and evolution management*. PhD thesis, University of California, Berkeley, 2004.

[37] Alok Choudhary, William Hendrix, Kathy Lee, Diana Palsetia, and Wei-Keng Liao. Social media evolution of the egyptian revolution. *Communications of the ACM*, 55(5):74–80, 2012.

[38] Shihabur Rahman Chowdhury, Arup Raton Roy, Maheen Shaikh, and Khuzaima Daudjee. A taxonomy of decentralized online social networks. *Peer-to-Peer Networking and Applications*, 8(3):367–383, 2015.

[39] Christopher Connolly, Patrick Lincoln, Ian Mason, and Vinod Yegneswaran. Trist: Circumventing censorship with transcoding-resistant image steganography. In *4th USENIX Workshop on Free and Open Communications on the Internet, FOCI '2014*, 2014.

[40] Mauro Conti, Arbnor Hasani, and Bruno Crispo. Virtual private social networks and a facebook implementation. *ACM Transactions on the Web*, 7(3):14, 2013.

[41] James Cowie, Alin Popescu, and Todd Underwood. Impact of hurricane katrina on internet infrastructure. *Renesys*, pages 1–9, 2005.

[42] Curt Cramer, Kendy Kutzner, and Thomas Fuhrmann. Bootstrapping locality-aware p2p networks. In *12th IEEE International Conference on Networks, ICON '2004*, volume 1, pages 357–361. IEEE, 2004.

[43] Leucio Antonio Cutillo, Refik Molva, and Melek Önen. Safebook: A distributed privacy preserving online social network. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM '2011*, pages 1–3. IEEE, 2011.

[44] Alberto Dainotti, Roman Amman, Emile Aben, and Kimberly C Claffy. Extracting benefit from harm: using malware pollution to analyze the impact of political and geophysical events on the internet. *ACM SIGCOMM Computer Communication Review*, 42(1):31–39, 2012.

[45] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages caused by censorship. In *The Internet Measurement Conference, IMC '2011*, pages 1–18, 2011.

[46] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. The anatomy of a scientific rumor. *Nature, Scientific reports*, 3, 2013.

[47] Ronald J Deibert. International plug'n play? citizen activism, the internet, and global public policy. *International Studies Perspectives*, 1(3):255–272, 2000.

[48] Taylor Dewey, Juliane Kaden, Miriam Marks, Shun Matsushima, and Beijing Zhu. The impact of social media on social unrest in the arab spring. *International Policy Program*, 2012.

[49] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, and John Lockwood. Deep packet inspection using parallel bloom filters. In *Symposium on High Performance Interconnects, HOTI '2003*, pages 44–51. IEEE, 2003.

[50] Ding Ding, Mauro Conti, and Renato Figueiredo. Impact of country-scale internet disconnection on structured and social p2p overlays. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM '2015*, pages 1–9. IEEE, 2015.

[51] Jochen Dinger and Oliver P Waldhorst. Decentralized bootstrapping of p2p systems: a practical view. In *8th International IFIP-TC Networking Conference, NETWORKING '2009*, pages 703–715. Springer, 2009.

[52] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[53] Yajuan Duan, Long Jiang, Tao Qin, and Ming Zhou. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING '2010*, pages 295–303, 2010.

[54] Enrique J Duarte-Melo and Mingyan Liu. Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. In *IEEE Global Telecommunications Conference, GLOBECOM' 2002.*, volume 1, pages 21–25. IEEE, 2002.

[55] L DAcunto, JA Pouwelse, and HJ Sips. A measurement of nat and firewall characteristics in peer-to-peer systems. In *Proceedings of Advanced School for Computing and Imaging, ASCI '2009*, volume 5031, pages 1–5, 2009.

[56] Sameh El-Ansary, Luc Onana Alima, Per Brand, and Seif Haridi. Efficient broadcast in structured p2p networks. In *International workshop on Peer-to-Peer systems, IPTPS '2003*, pages 304–314. Springer, 2003.

[57] Nicole B Ellison, Charles Steinfield, and Cliff Lampe. The benefits of facebook friends: social capital and college students use of online social network sites. *Journal of Computer-Mediated Communication*, 12(4):1143–1168, 2007.

[58] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. Examining how the great firewall discovers hidden circumvention servers. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference, IMC '2015*, pages 445–458, 2015.

[59] B. Eriksson, R. Durairajan, and P. Barford. Riskroute: a framework for mitigating network outage threats. In *The 12th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '2013*, pages 405–416, 2013.

[60] Dave Evans. The internet of things: how the next evolution of the internet is changing everything. *CISCO white paper*, 1, 2011.

[61] José Luis Fernández-Alemán, Inmaculada Carrión Señor, Pedro Ángel Oliver Lozoya, and Ambrosio Toval. Security and privacy in electronic health records: A systematic literature review. *Journal of Biomedical Informatics*, 46(3):541–562, 2013.

[62] R. J Figueiredo, S Aditya, K Jeong, and K Subratie. Seamless networking among edge devices and clouds with fog social virtual networks. In *Sensor to Cloud Architectures Workshop, SCAW '2016*, 2015.

[63] Alexandre Fonseca, A Vu, and Peter Grman. Evaluation of nosql databases for large-scale decentralized microblogging. *Universitat Politècnica de Catalunya*, 2013.

[64] Bryan Ford. Unmanaged internet protocol: taming the edge network management crisis. *ACM SIGCOMM Computer Communication Review*, 34(1):93–98, 2004.

[65] Bryan Alexander Ford. *UIA: A Global Connectivity Architecture for Mobile Personal Devices*. PhD thesis, MIT, 2008.

[66] Enrico Franchi and Michele Tomaiuolo. Distributed social platforms for confidentiality and resilience. *Social Network Engineering for Secure Web Data and Services*, page 114, 2013.

[67] Paul Francis. Network address translation (nat). *ACM SIGCOMM Computer Communication Review*, 45(2):50–50, 2015.

[68] Caroline Free, Gemma Phillips, Leandro Galli, Louise Watson, Lambert Felix, Phil Edwards, Vikram Patel, and Andy Haines. The effectiveness of mobile-health technology-based health behaviour change or disease management interventions for health care consumers: a systematic review. *PLoS med*, 10(1):351–362, 2013.

[69] Miguel Freitas. twister-a p2p microblogging platform. *arXiv preprint arXiv:1312.7152*, 2013.

[70] Chris GauthierDickey and Christian Grothoff. Bootstrapping of peer-to-peer networks. In *International Symposium on Applications and the Internet, SAINT '2008*, pages 205–208. IEEE, 2008.

[71] Geoffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick Drew McDaniel, and Aviel D Rubin. Working around bgp: An incremental approach to improving security and accuracy in interdomain routing. In *The Network and Distributed System Security Symposium, NDSS '2003*, pages 417–428, 2003.

[72] Ilya Grigorik. *High Performance Browser Networking*. O'Reilly Media, Sep 2013.

[73] Julian B Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: Overview and case study. In *First Workshop on Hot Topics in Understanding Botnets, HotBots '2003*, pages 1–10. USENIX, 2007.

[74] Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, and Robbert Van Renesse. Kelips: Building an efficient and stable p2p dht through increased memory and background overhead. In *International Workshop on Peer-to-Peer Systems, IPTPS '2003*, pages 160–169. Springer, 2003.

[75] Donna L Hoffman, Thomas P Novak, and Alladi Venkatesh. Has the internet become indispensable? *Communications of the ACM*, 47(7):37–42, 2004.

[76] Shuji Hokkyo, Shoji Furuchi, and Takayuki Ito. Ip address assigning method, vlan changing device, vlan changing system and quarantine process system, June 27 2005. US Patent App. 11/166,274.

[77] John Holowczak and Amir Houmansadr. Cachebrowser: Bypassing chinese censorship without proxies using cached content. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 70–83, New York, NY, USA, 2015. ACM.

[78] Liangjie Hong, Ovidiu Dan, and Brian D Davison. Predicting popular messages in twitter. In *Proceedings of the 20th International Conference Companion on World Wide Web, WWW '2011*, pages 57–58. ACM, 2011.

[79] Amir Houmansadr, Thomas J Riedl, Nikita Borisov, and Andrew C Singer. I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention. In *The Network and Distributed System Security Symposium, NDSS 2013*, 2013.

[80] Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. Exploiting social relations for sentiment analysis in microblogging. The ACM WSDM Conference Series Web Search and Data Mining, WSDM '2013, pages 537–546. ACM, 2013.

[81] Jinzhou Huang and Hai Jin. Somed: A hybrid dht framework towards scalable decentralized microblogging services. In *2013 International Conference on Parallel and Distributed Systems, ICPADS '2013*, pages 534–539. IEEE, 2013.

[82] Yalin Huang, Zongyin Zhao, Yan Liu, and Xu Yao. System level design of address allocation for a private ip network. In *Proceedings of the 26th Conference of Spacecraft TT&C Technology in China*, pages 317–326. Springer, 2013.

[83] Daniel Lazaro Iglesias, Joan-Manuel Marques, Guillem Cabrera, Helena Rifa-Pous, and Albert Montane. Hornet: microblogging for a contributory social network. *IEEE Internet Computing*, 16(3):37–45, 2012.

[84] John Jannotti, David K Gifford, Kirk L Johnson, M Frans Kaashoek, et al. Overcast: reliable multicasting with on overlay network. In *Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation, OSDI '2000*, page 14. USENIX, 2000.

[85] Bernard J Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. Micro-blogging as online word of mouth branding. In *The ACM Conference on Human Factors in Computing Systems, CHI '2009*, pages 3859–3864. ACM, 2009.

[86] 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*, pages 56–65. ACM, 2007.

[87] Shuo Jia, Pierre St Juste, and Renato Figueiredo. A multidimensional heuristic for social routing in peer-to-peer networks. In *The Annual IEEE Consumer Communications and Networking Conference, CCNC '2013*, pages 329–335, 2013.

[88] Uwe Johannen, James Gomez, and Steven Gan. *Asian cyberactivism: Freedom of expression and media censorship.* Friedrich Naumann Foundation, East and Southeast Asia Regional Office, 2004.

[89] Ben Jones, Tzu-Wen Lee, Nick Feamster, and Phillipa Gill. Automated detection and fingerprinting of censorship block pages. In *Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '2014*, pages 299–304. ACM, 2014.

[90] Pierre Juste, Kyuho Jeong, Heungsik Eom, Corey Baker, and Renato Figueiredo. Tincan: User-defined p2p virtual network overlays for ad-hoc collaboration. *ICST Transactions*, 2014.

[91] Pierre Juste, David Wolinsky, P Oscar, Michael Covington, and Renato Figueiredo. Socialvpn: Enabling wide-area collaboration with integrated social and overlay networks. *Computer Networks*, pages 1926–1938, 2010.

[92] Min Suk Kang, Soo Bum Lee, and Virgil D Gligor. The crossfire attack. In *IEEE Symposium on Security and Privacy, S&P '2013*, pages 127–141. IEEE, 2013.

[93] Alexander Y Kaplan, Sergei L Shishkin, Ilya P Ganin, Ivan A Basyul, and Alexander Y Zhigalov. Adapting the p300-based brain–computer interface for gaming: a review. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(2):141–149, 2013.

[94] Liran Katzir and Stephen J. Hardiman. Estimating clustering coefficients and size of social networks via random walk. *ACM Transactions on Web*, pages 19:1–19:20, 2015.

[95] Won Kim. Cloud computing: Today and tomorrow. *Journal of Object Technology*, 8(1):65–72, 2009.

[96] Gary King, Jennifer Pan, and Margaret E Roberts. How censorship in china allows government criticism but silences collective expression. *American Political Science Review*, 107(02):326–343, 2013.

[97] Hans K Klein. Tocqueville in cyberspace: Using the internet for citizen associations. *The Information Society*, 15(4):213–220, 1999.

[98] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web, WWW '2010*, pages 591–600. ACM, 2010.

[99] Anders Olof Larsson and Hallvard Moe. Studying political microblogging: Twitter users in the 2010 swedish election campaign. *New Media & Society*, 14(5):729–747, 2012.

[100] Christopher S Leberknight, Mung Chiang, and Felix Ming Fai Wong. A taxonomy of censors and anti-censors: Part i-impacts of internet censorship. *International Journal of E-Politics*, 3(2):52–64, 2012.

[101] Soo Bum Lee, Min Suk Kang, and Virgil D Gligor. Codef: collaborative defense against large-scale link-flooding attacks. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, CoNEXT '2013*, pages 417–428. ACM, 2013.

[102] Sergey Legtchenko, Sébastien Monnet, Pierre Sens, and Gilles Muller. Relaxdht: A churn-resilient replication strategy for peer-to-peer distributed hash-tables. *ACM Transactions on Autonomous and Adaptive Systems*, 7(2):28, 2012.

[103] Derek Leonard, Zhongmei Yao, Vivek Rai, and Dmitri Loguinov. On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. *IEEE/ACM Transactions on Networking*, 15(3):644–656, 2007.

[104] Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F Mokbel. Lars: A location-aware recommender system. In *IEEE International Conference on Data Engineering, ICDE '2012*, pages 450–461, 2012.

[105] Kevin Lewis, Jason Kaufman, Marco Gonzalez, Andreas Wimmer, and Nicholas Christakis. Tastes, ties, and time: A new social network dataset using facebook. com. *Social Networks*, 30(4):330–342, 2008.

[106] David Liben, Jasmine Novak, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences of the United States of America*, pages 11623–11628, 2005.

[107] Patrick Lincoln, Ian Mason, Phillip A Porras, Vinod Yegneswaran, Zachary Weinberg, Jeroen Massar, William Allen Simpson, Paul Vixie, and Dan Boneh. Bootstrapping communications into an anti-censorship system. In *The 2nd USENIX Workshop on Free and Open Communications on the Internet, FOCI*, 2012.

[108] Yunhao Liu, Zhenyun Zhuang, Li Xiao, and Lionel M Ni. Aoto: Adaptive overlay topology optimization in unstructured p2p systems. In *IEEE Global Telecommunications Conference. GLOBECOM '2003.*, volume 7, pages 4186–4190. IEEE, 2003.

[109] Gilad Lotan, Erhardt Graeff, Mike Ananny, Devin Gaffney, Ian Pearce, et al. The arab spring— the revolutions were tweeted: Information flows during the 2011 tunisian and egyptian revolutions. *International Journal of Communication*, 5:31, 2011.

[110] Kuo-Fong Ma, Hidemi Tanaka, Sheng-Rong Song, Chien-Ying Wang, Jih-Hao Hung, Yi-Ben Tsai, Jim Mori, Yen-Fang Song, Eh-Chao Yeh, Wonn Soh, et al. Slip zone and energetics of a large earthquake from the taiwan chelungpu-fault drilling project. *Nature*, 444(7118):473–476, 2006.

[111] Ratul Mahajan, Miguel Castro, and Antony Rowstron. Controlling the cost of reliability in peer-to-peer overlays. In *The International Workshop on Peer-to-Peer Systems, IPTPS '2003*, pages 21–32, 2003.

[112] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding bgp misconfiguration. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 3–16. ACM, 2002.

[113] S. Marti, P. Ganesan, and H. Garcia-Molina. Sprout: P2p routing with social networks. In *EDBT Workshop*, pages 425–435, 2004.

[114] Ivan Martinovic, Doug Davies, Mario Frank, Daniele Perito, Tomas Ros, and Dawn Song. On the feasibility of side-channel attacks with brain-computer interfaces. In *21st USENIX Security Symposium, USENIX '2012)*, pages 143–158, 2012.

[115] Afzal Mawji and Hossam S Hassanein. Bootstrapping p2p overlays in manets. In *IEEE Global Communications Conference, GLOBECOM '2008*, pages 1–5. IEEE, 2008.

[116] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *The International Workshop on Peer-to-Peer Systems, IPTPS '2002*, pages 53–65. 2002.

[117] Sally J McMillan and Margaret Morrison. Coming of age with the internet a qualitative exploration of how the internet has become an integral part of young peoples lives. *New Media & Society*, 8(1):73–95, 2006.

[118] Raymond E Miles and Charles C Snow. Causes of failure in network organizations. *California Management Review*, 34(4):53–72, 1992.

[119] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[120] Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet measurement, IMC '2007*, pages 29–42. ACM, 2007.

[121] Prateek Mittal, Matthew K. Wright, and Nikita Borisov. Pisces: Anonymous communication using social networks. In *The Network and Distributed System Security Symposium, NDSS 2013*, 2013.

[122] Zubair Nabi. The anatomy of web censorship in pakistan. *arXiv preprint arXiv:1307.1144*, 2013.

[123] Arvind Narayanan, Vincent Toubiana, Solon Barocas, Helen Nissenbaum, and Dan Boneh. A critical look at decentralized personal data architectures. *arXiv preprint arXiv:1202.4503*, 2012.

[124] Sharon Erickson Nepstad. Mutiny and nonviolence in the arab spring exploring military defections and loyalty in egypt, bahrain, and syria. *Journal of Peace Research*, 50(3):337–349, 2013.

[125] Daiyuu Nobori and Yasushi Shinjo. Vpn gate: A volunteer-organized public vpn relay system with blocking resistance for bypassing government censorship firewalls. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI '2014*, pages 229–241. USENIX, 2014.

[126] James S O'Rourke IV, Brynn Harris, and Allison Ogilvy. Google in china: government censorship and corporate reputation. *Journal of Business Strategy*, 28(3):12–22, 2007.

[127] Zizi Papacharissi. The virtual sphere the internet as a public sphere. *New Media & Society*, 4(1):9–27, 2002.

[128] Harris Papadakis, Paraskevi Fragopoulou, Evangelos P Markatos, and Mema Roussopoulos. Ita: innocuous topology awareness for unstructured p2p networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1589–1601, 2013.

[129] Fawaz Paraiso, Philippe Merle, and Lionel Seinturier. socloud: A service-oriented component-based paas for managing portability, provisioning, elasticity, and high availability across multiple clouds. *Computing*, 98(5):539–565, 2016.

[130] Alexandre Passant, Tuukka Hastrup, Uldis Bojars, and John Breslin. Microblogging: A semantic and distributed approach. *Proceedings of 4th Workshop on Scripting for the Semantic Web, SFSW '2008*, 2008.

[131] Timothy Perfitt and Burkhard Englert. Megaphone: Fault tolerant, scalable, and trustworthy p2p microblogging. In *2010 Fifth International Conference on Internet and Web Applications and Services, ICIW '2010*, pages 469–477. IEEE, 2010.

[132] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering spam with behavioral blacklisting. In *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '2007*, pages 342–351. ACM, 2007.

[133] Venugopalan Ramasubramanian and Emin Gün Sirer. Perils of transitive trust in the domain name system. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, IMC '2005*, pages 35–35. USENIX Association, 2005.

[134] Alessandra Sala, Lili Cao, Christo Wilson, Robert Zablit, Haitao Zheng, and Ben Y Zhao. Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th International Conference on World Wide Web, WWW '2010*, pages 861–870. ACM, 2010.

[135] Daniel Sandler and Dan S Wallach. Birds of a fethr: open, decentralized micropublishing. In *The 8th International Workshop on Peer-to-Peer Systems, IPTPS '2009*, page 1, 2009.

[136] Mohamed Sarwat, Justin Levandoski, Ahmed Eldawy, and F Mokbel. Lars*: A scalable and efficient location-aware recommender system. IEEE Transactions on Knowledge and Data Engineering, 2013.

[137] Marius Senftleben, Mihai Bucicoiu, Erik Tews, Frederik Armknecht, Stefan Katzenbeisser, and Ahmad-Reza Sadeghi. Mop-2-mop–mobile private microblogging. In *Financial Cryptography and Data Security, FC '2014*, pages 384–396. Springer, 2014.

[138] Agusti Solanas, Constantinos Patsakis, Mauro Conti, Ioannis S Vlachos, Victoria Ramos, Francisco Falcone, Octavian Postolache, Pablo A Pérez-Martínez, Roberto Di Pietro, Despina N Perrea, et al. Smart health: a context-aware health paradigm within smart cities. *IEEE Communications Magazine*, 52(8):74–81, 2014.

[139] Yair Sovran, Alana Libonati, and Jinyang Li. Pass it on: Social networks stymie censors. In *Proceedings of the 7th International Conference on Peer-to-peer Systems*, IPTPS'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[140] Kate Starbird and Leysia Palen. (how) will the revolution be retweeted?: information diffusion and the 2011 egyptian uprising. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW 2012*, pages 7–16. ACM, 2012.

[141] P. St.Juste, Heungsik Eom, Kyungyong Lee, and R.J. Figueiredo. Enabling decentralised microblogging through p2pvpns. In *The Annual IEEE Consumer Communications and Networking Conference, CCNC '2013*, pages 323–328. IEEE, 2013.

[142] Ion Stoica, Robert Morris, David Liben, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, pages 17–32, 2003.

[143] Daniel Stutzbach, Reza Rejaie, and Subhabrata Sen. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *IEEE/ACM Transactions on Networking*, 16(2):267–280, 2008.

[144] Yanjun Sun, Omer Gurewitz, and David B Johnson. Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys '2008*, pages 1–14. ACM, 2008.

[145] Kurt Thomas, Chris Grier, and Vern Paxson. Adapting social spam infrastructure for political censorship. In *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats, LEET '2012*, pages 13–13. USENIX Association, 2012.

[146] Kevin Thompson, Gregory J Miller, and Rick Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, 1997.

[147] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[148] Bram van de Laar, Hayrettin Gürkök, Danny Plass-Oude Bos, Mannes Poel, and Anton Nijholt. Experiencing bci control in a popular computer game. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(2):176–184, 2013.

[149] Brain Vetter, Feiyi Wang, and Shyhtsun Felix Wu. An experimental study of insider attacks for ospf routing protocol. In *Proceedings of International Conference on Network Protocols, ICNP '1997*, pages 293–300. IEEE, 1997.

[150] Arno Wacker, Gregor Schiele, Sebastian Holzapfel, and Torben Weis. A nat traversal mechanism for peer-to-peer networks. *Peer-to-Peer Computing*, 2008:81–83, 2008.

[151] Hao Wang, Yang Richard Yang, Paul H Liu, Jia Wang, Alexandre Gerber, and Albert Greenberg. Reliability as an interdomain service. *Computer Communication Review*, pages 229–240, 2007.

[152] Qiang Wang, Olga Sourina, and Minh Khoa Nguyen. Eeg-based" serious" games design for medical applications. In *2010 International Conference on Cyberworlds, CW '2010*, pages 270–276. IEEE, 2010.

[153] Qiyan Wang, Xun Gong, Giang TK Nguyen, Amir Houmansadr, and Nikita Borisov. Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing. In *Proceedings of the 2012 ACM conference on Computer and Sommunications Security, CCS '2012*, pages 121–132. ACM, 2012.

[154] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas Hopper. rbridge: User reputation based tor bridge distribution with privacy preservation. In *The Network and Distributed System Security Symposium, NDSS 2013*, 2013.

[155] Yuzhi Wang, Ping Ji, Borui Ye, Pengjun Wang, Rong Luo, and Huazhong Yang. Gohop: Personal vpn to defend from censorship. In *16th International Conference on Advanced Communication Technology, ICACT '2014*, pages 27–33. IEEE, 2014.

[156] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. Stegotorus: a camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '2012*, pages 109–120. ACM, 2012.

[157] Kathryn R Wentzel and Cynthia A Erdley. Strategies for making friends: Relations to social behavior and peer acceptance in early adolescence. *Developmental Psychology*, 29(5):819, 1993.

[158] Philipp Winter, Tobias Pulls, and Juergen Fuss. Scramblesuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society, WPES '2013*, pages 213–224. ACM, 2013.

[159] Gadi Wolfsfeld, Elad Segev, and Tamir Sheafer. Social media and the arab spring politics comes first. *The International Journal of Press/Politics*, 18(2):115–137, 2013.

[160] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI '12*, pages 179–182, 2012.

[161] David Isaac Wolinsky, Pierre St Juste, P Oscar Boykin, and Renato Figueiredo. Addressing the p2p bootstrap problem for small overlay networks. In *IEEE Tenth International Conference on Peer-to-Peer Computing, P2P '2010*, pages 1–10. IEEE, 2010.

[162] Eric Wustrow, Colleen M Swanson, and J Alex Halderman. Tapdance: End-to-middle anticensorship without flow blocking. In *USENIX Security Symposium*, pages 159–174, 2014.

[163] Tianyin Xu, Yang Chen, Xiaoming Fu, and Pan Hui. Twittering by cuckoo: decentralized and socio-aware online microblogging services. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 473–474. ACM, 2010.

[164] Tianyin Xu, Yang Chen, Jin Zhao, and Xiaoming Fu. Cuckoo: towards decentralized, socio-aware online microblogging services and data measurements. In *Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement, HotPlanet '2010*, page 4, 2010.

[165] Kenji Yamanishi and Yuko Maruyama. Dynamic syslog mining for network failure monitoring. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '2005*, pages 499–508. ACM, 2005.

[166] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering, ICDE '2003*, pages 49–60. IEEE, 2003.

[167] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2004.

[168] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys & Tutorials*, 8(1):24–37, 2006.

[169] Dejin Zhao and Mary Beth Rosson. How and why people twitter: the role that micro-blogging plays in informal communication at work. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work, GROUP '2009*, pages 243–252. ACM, 2009.

[170] Lidong Zhou, Lintao Zhang, Frank McSherry, Nicole Immorlica, Manuel Costa, and Steve Chien. A first look at peer-to-peer worms: Threats and defenses. In *International Workshop on Peer-to-Peer Systems, IPTPS '2005*, pages 24–35. Springer, 2005.

[171] Peng Zhou, Xiapu Luo, Ang Chen, and Rocky KC Chang. Stor: Social network based anonymous communication in tor. *arXiv preprint arXiv:1110.5794*, 2011.