FRIENDSOURCING TO DETECT NETWORK MANIPULATION

BY

RAVINDER SHANKESI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Professor Carl A. Gunter, Chair
Associate Professor Nikita Borisov
Associate Professor Matthew Caesar
Associate Professor Nick Feamster, Georgia Institute of Technology
Associate Professor Karrie G. Karahalios

Abstract

Traditionally, security on the Internet has been concerned with threats posed by edge nodes (hosts) to other hosts and to the middle nodes of the network (routers). An example of the former is the spread of viruses and an example of the latter is congestion caused by packet floods. However, in recent years, we have seen a rising number of instances in which hosts seek to defend themselves against the middle of the network. This has been particularly evident with Internet censorship, where countries seek to suppress political unrest by closing off access to social networks, particular web sites, or even the Internet itself. Another area of controversy and concern arises with violations of network neutrality, in which an Internet Service Provider (ISP) prejudices in favor of some types of packets compared to others, typically for some commercial competitive advantage or remuneration. Fresh types of abuse are emerging as well, such as ISPs who alter click traffic or advertising. All of these fall into a general category that one can describe as *network manipulation*.

Many methods have been investigated to detect and circumvent network manipulation. For instance, researchers have devoted projects to determining which words trigger various national censorship firewalls and strategies for deploying proxies to enable tunnels that prevent such firewalls from recognizing that hosts are accessing forbidden sites. Other research has deployed nodes that can be used for testing network neutrality violations. One of the key challenges that faces many of these techniques is the requirement to get "help on the inside", that is, to get cooperation from nodes that are subject to suspected network manipulation. This thesis proposes to explore this specific problem with the aim of addressing it through *friendsourcing*. Friendsourcing is a kind of crowdsourcing in which individuals use their social networks to gain help from their friends. For instance, if an individual is having trouble with a network link he may ask his friend if she is also having trouble with that link. The results could point to problems in various

places, such as a server outage, a problem with a client, an accidental network outage, or deliberate network manipulation.

Our thesis is that, *social networks are an effective and efficient way for a user to acquire a sufficiently distributed sensor network required for detecting the source of censorship on the underlying communication network*.

Proving that friendsourcing is a viable strategy to detect network manipulation required three primary features. First, we needed to show that is possible to carry out recruitment that finds a collection of hosts in the right network locations; we show this through *coverage* and *redundancy* of multiple datasets. Second, we need to prove that friendsourcing is an efficient strategy to detect network manipulation; we do this through developing and validating optimizations required for recruitment. Third, we must show that it is a practical; we show this through a real-life field study outside a laboratory setting.

Our contributions in this thesis are:

1. We were the first to show the effectiveness of friendsourcing to detect network manipulation.

2. We developed optimization algorithms that can make friendsourcing viable in practice.

3. We showed the practicality of friendsourcing, by deploying it in a real-life experiment which found a number of web-based manipulations in India.

To my parents.

Acknowledgments

I am grateful to my advisor, Carl Gunter, for his guidance, patience, and understanding. Even after all these years, I am still amazed at Carl's ability to restate a research problem precisely and succinctly. I benefited from not only his insights and knowledge of my narrow field of research, but also from his broad understanding of the field. His contributions to this thesis cannot be overstated.

I would also like to thank my thesis committee members, Nikita Borisov, Matthew Caesar, Nick Feamster, and Karrie Karahalios. Their encouragement and feedback helped make this thesis better than it would have been otherwise.

I would also like to thank all my collaborators: Omid Fatemieh, Ralf Sasse, Musab AlTurki, Rohit Chadha, Jose Meseguer, Mahesh Viswanathan, Zahid Anwar, Roy Campbell, Filippo Giaochin, and Wook Shin. I enjoyed all the discussions I had with my lab mates in the security lab, Omid Fatemieh, Fariba Khan, Sonia Jahid, Michael LeMay, Jianqing Zhang, Lars Olson, and Ahn Nguyen. Staying in 4403 also meant that I fell into lively debates and discussions with Anthony Cozzie, Matt Hicks, Hui Xue, Shuo Tang, Aparna Sasidharan, and Nathan Dautenhahn.

I also have happy memories of all the fun and exploring I did with Rohit Chadha, Dongyun Jin, and Sandeep Reddy.

I also appreciate all the time I spent with Alicia Schofield. Her understanding, patience, and her belief in me have been a great comfort.

I also appreciate my family: my brother Mahender, sister-in-law Sarita, my sister Madhavi and my brother-in-law Murali for always being there for me. My nephew Abhiram and my niece Aashritha were both born during the course my Ph.D. and are a constant source of joy.

Above all, I am grateful for my parents. My mother, for always been caring and supportive. My father, who unfortunately could not see me graduate, for setting me an example with this hard work and steadfast principles. Both of them, for their unconditional love.

Table of Contents

Chapter 1

Introduction

The Internet is a large, globally distributed, network of networks that serves billions of users world-wide. These networks can communicate with each other using the standard internet protocol suite. They individual networks themselves are managed by a variety of actors such as Internet Service Providers (ISPs), academic and commercial organizations, and even countries, each with their own distinct policy. Any network connection between two end hosts on the Internet, therefore, may involve packets traversing through multiple countries, ISPs, Organizational firewalls, as well network firewalls on end hosts. The Internet's distributed architecture and the protocols were historically built with an assumption of unreliable, but mutually cooperating entities. Core internet protocols (like TCP, UDP, and IP) as well as applications that run on top of these protocols (like email, Voice-over-IP, and the Web), therefore, are easily affected by the policies of these disparate parties.

Recent history has shown a rise in occurrences of *deliberate* network interference by many of these parties (see Chapter 2). Each of these parties has different motivations for such manipulation: organizations may prefer to block websites which are considered detrimental to productivity, ISPs may block protocols that utilize a lot of network bandwidth, and countries may block websites or keywords for religious or political considerations. This interference is referred to as *network neutrality* violation when done by ISPs or *cyber censorship* [1] when done by countries. In this thesis, we refer to such interference as *network manipulation*.

Determining the presence and source of such deliberate network manipulation is not always easy and may require performing multiple network probes from different parts of the Internet. For instance, recently, the Australian government asked ISPs within their country to block certain websites (and IP blocks) which were considered undesirable. This blacklist was not made public, and included an educational website hosted in the US. Furthermore, not all the ISPs within Australia followed the government's orders. The Electronic Frontier Foundation

(EFF) collected traceroute information from nodes from a large of various ISPs [2] to identify that blocking was occurring at certain ISPs. The EFF was able to confirm that the blocking was due to Australian government's censorship list (and not say policies of a given ISP) only by talking operators of one of the ISPs. An individual user on their own would not have the network visibility to be able to make such inferences.

In this thesis we propose, *friendsourcing* as a novel way of detecting network manipulation. Friendsourcing, is a play on 'crowdsourcing' where, instead of asking a random collection of people (i.e., a crowd), we ask a group of friends to help us in some activity. Friendsourcing, as an ad hoc strategy in solving problems, is an old idea. With the increasing use of online social networks, the effectiveness of friendsourcing is being explored for solving problems in a more systematic manner [3]. In our approach, we use a user's friend circle as a sensor network. A user who suspects that some party is deliberately interfering with their communications asks their friends to contact the same server by sending a probe from their individual machine. The *presence* of any manipulation can be detected by comparing the results of probes sent from different vantage points on this sensor network. Correlating the interference of network connections with their network attributes (i.e., organization, ISP, or country) will also give us the probable *source* of the network manipulation.

The rise in network manipulation has led to a lot of research in detection and circumvention of such manipulation (see Chapter 3 for more details). Existing research has used two broad approaches to detect network manipulation. They involve either active measurements or passive collection of data from end hosts. On the active side, a common strategy is to use a large group of users to perform a fixed suite of experiments from their individual machines. Alternately, they use existing networking research infrastructure (such as PlanetLab or Measurement-Lab) to perform the probing. On the passive side, we have techniques that collect self-reported information from a crowd, or a tool that run at a user's machine that collect information about failing network connections.

These approaches are limited either by the fact that they perform a fixed set of experiments and talk to fixed measurement servers, or by their reliance on the presence of members of the crowd who run into exactly the same problem for any given end user. These results might also be biased by the fact that they use research infrastructure that is not subject to the same restrictions as those of end users. In contrast, our approach using friendsourcing can perform on-demand,

active measurements talking to real web servers, from the vantage of real-life users. Furthermore, instead of relying on indifferent strangers in a crowd, it relies on the strength of a user's social ties to get cooperation from friends to perform the necessary measurements.

1.1 Thesis

Social networks are an effective and efficient way for a user to acquire a sufficiently distributed sensor network required for detecting the presence and probable source of network manipulation on the underlying communication network.

1.2 Challenges

Validating friendsourcing as a strategy to detect network manipulation requires us to address multiple issues. We give a list of these challenges below and show how we addressed these challenges later in Section 1.3.

1.2.1 Proof of Effectiveness

An important requirement for the detection of network manipulation is performing experiments from multiple sensor nodes on the communication network. Ideally, we would like to perform the same network probing experiment from nodes which have different network attributes, i.e., from different countries, ISPs, or organizations. For instance, if all of the user's friends belong the same ISP (or country) we will not be able to effectively determine whether the ISP (or country) was to blame for loss in connectivity. Alternately, if none of the user's friends belong to the same ISP, we will not be able to conclusively prove that it is the ISP to blame for any observed manipulation. For friendsourcing to be an effective strategy we need to prove that, on average, a user's friend circle has enough variety in such network attributes to be able to detect the presence and probable source of network manipulation.

A fundamental challenge with proving this property is the lack of publicly available data which comprises of both a social network and a corresponding communication network attributes. Another problem is the possible bias associated with

choosing any given type of social network as a basis for such an analysis. We give a brief discussion of how we address these challenges in Section 1.3.1.

1.2.2 Resiliency

A problem with relying on friends is that, not all friends of a user may be available or willing to help the user in performing the network probes. We need to show that friendsourcing as a strategy has enough resilience to handle lack of cooperation from some friends. One supporting evidence for the effectiveness of friendsourcing can be found in existing literature which shows the users are more likely to cooperate with their friends than strangers (see Chapter 3 for more details). We would also like to prove that there is enough redundancy for a given user in finding friends to perform their analysis. We given an overview of our results in Section 1.3.2.

1.2.3 Efficiency

An important challenge in using friendsourcing is to make friend recruitment and subsequent network probing efficient. Asking too few friends may result in lack of precision when it comes to identifying the possible presence or source of network manipulation. Choosing too many friends will result in an excessive number of probes being sent to the end hosts as well as between the users. This would result in our infrastructure being a unwitting source of a denial of service towards a target web server. We give an overview of the optimizations that we developed to minimize such occurrences in Section 1.3.3.

1.2.4 Practicality

We need to show that friendsourcing is practical and would be used by users in real-life outside the laboratory setting. While many design architectures can be evaluated in a laboratory setting; having a system evaluated by real, non-technical, users requires that we have an implementation that is easy to use and is platform independent. This means that we had to sacrifice some functionality for the sake of wider deployment. We give a brief overview of our implementation in Section 1.3.4.

1.3 Demonstration

1.3.1 Coverage

To prove the effectiveness of friendsourcing in detecting network manipulation, we had to come up with metrics to measure *coverage* and evaluate it on datasets. Informally, *coverage* tells us, for every user what part of their upstream entities, who can engage in network manipulation, can be verified with the help of that user's friends (see Section5.2 for a more rigorous definition). A high coverage (of 100%) indicates that a user can detect the presence and possible source of network manipulation from all of their upstream entities. The coverage of a given user may also increase as we go deeper into their social network (i.e., using friends of friends as well instead of just friends). The coverage, can also be measured restricted to some the network attribute. For instance, a user may only be interested in knowing if their ISP is source of network manipulation.

We evaluated coverage metrics using four different kinds of social network datasets: a collaboration based on authors who published papers in Arxiv in the area of high-energy theoretical physics, two location check-in based social networks Gowalla and BrightKite, and a general purpose social network GooglePlus. Not all of the network attributes for users in these datasets were directly available, so we had to infer some of the attributes (see 5.1 for a discussion). We were able to show good coverage for these different kinds of social networks suggesting that friendsourcing is good strategy for acquiring sensors for a network probe. We give more details of our datasets in Section 5.1 and coverage analysis in Section 5.2.

1.3.2 Redundancy

While a good coverage for a user shows that it is possible to acquire the necessary network probes required to perform the analysis, there is no guarantee that all of a given user's friends may be interested or available in helping the user when needed. In practice, it might only a fraction of a given user's friends who respond to a user. To demonstrate the resilience of social networks for this task, we analyzed the *redundancy* of a user's social network for any given network attribute. For instance, a user with high redundancy at the level of ISP, can find multiple friends who can perform measurements from different ISPs. Thus, it will be pos-

sible for that user to detect any network manipulation by an ISP even if only a small fraction of their friends respond to requests.

We analyzed the redundancy for ISPs as well as Countries for all the datasets. Our analysis showed that all of the datasets had high redundancy (>5) in terms of ISPs and Countries assuming a friendsourcing depth of 2 (i.e., when using friends and friends-of-friends). The one exception was that a sparse dataset (Arxiv) required a larger friendsourcing depth to achieve the same redundancy. We show the results of our analysis in Section 5.3.

1.3.3 Optimal Recruitment

Our initial analysis of a naive recruitment strategy using friendsourcing showed that, without optimizations, it will be highly inefficient and lead to large number of messages being passed between users, as well as, a large load on the end host of the measurements. We show that certain optimizations can lead to significant improvements in both messages passed and probes done, without sacrificing the quality of the measurement.

Briefly, these optimizations rely on some pre-sharing of network attributes between users. This will allow users to recruit a much smaller subset of their friends while performing the experiments. More details of the optimization algorithm and analysis of its effectiveness are given in Section 5.4.

1.3.4 Field Deployment

To show the practicality of our strategy we implemented and deployed a tool which was used by real life users. For ease-of-use, the tool was developed as a website with Tomcat back-end. The website used signed Java Applets to perform the actual network measurements from the perspective of end users. For reasons of practicality, we also used a centralized web server to serve the applet based website and to collected the results of the field study. The website maintained the friendship relationship of the users and implemented the optimal recruitment algorithms to choose the set of friends when performing network probing.

Although our architecture is flexible enough to detect various kinds of network manipulation, our implementation was restricted to detecting network manipulation of HTTP. It could detect manipulation of HTTP that was based on dropping

of packets, forging of HTTP status codes, forging of DNS replies, and modification of the web page content. It could also identify whether the probable source of network manipulation was the organization, the ISP, or the country. We give more details of our implementation and the results of our field study in Chapter 6.

1.4 Overview and Contributions

The rest of the thesis is organized into the following chapters. Chapter 2 gives an overview of network manipulation and the various actors performing network manipulation and their capabilities. It also gives an overview of the datasets we use for our analysis. Chapter 3 gives an overview of the current state of research on detection and circumvention of network manipulation. Chapter 4 gives an overview of the various requirements, assumptions and design choices we have in implementing the system. Chapter 5 gives an analysis of the coverage and redundancy metrics of our datasets. Chapter 6 describes the architecture and implementation of of field study. It also gives the results of experiments we conducted using the system. Chapter 7 has some discussion on conclusions of our study and on future extensions.

We list the contributions of this thesis below.

1.4.1 Effectiveness

We are the first study to show the effectiveness of Friendsourcing as a mechanism for detecting network manipulation. To our knowledge, this is the first study that looked at the feasibility of systematically using a social network to perform communication network diagnostics.

In Section 5.2, we analyze the effectiveness of using social networks as a mechanism for acquiring network coverage. We show that a user on a social network, by requesting only their friend neighborhood (friends and friends of friends), can get sufficiently good *coverage* (i.e. more than 90%) of the underlying communication network. We also showed the resilience of a social network, by showing that social networks we analyzed had high redundancy (>5) for detecting various network attributes.

1.4.2 Efficient Recruitment

We developed optimization algorithms that will reduce the communication overhead between friends as well as the network probe overhead in a distributed architecture for the task of detecting network manipulation. Our analysis showed that these optimizations led to a reduction in the number of messages (and probes) by more than (95%).

1.4.3 Implementation and Field Evaluation

We had a proof-of-concept implementation that used friendsourcing as viable a strategy to detect network manipulation. Our prototype was suitable for detecting blocking-based, protocol-based and DNS-based network manipulation on the web. We performed a field study with 54 real-life users in India, all of whom were acquired from an initial pair of users. Our field study was able to detect 64 distinct blocked URLs within various ISPs in India, as well as various kinds of blocking mechanisms used in 13 different ISPs in India.

Chapter 2

Background

In this chapter we give an overview of some background concepts on computer networks (and the Internet in particular) and on social networks relevant to this thesis. To avoid confusion, hereafter, we will use the term *network* to refer to communication networks (or often the Internet) and the more qualified term *social network* when referring to a network consisting of individual users. Online Social Networks (OSNs) or Social Networking Sites (SNS) are online web-based platforms that allow people to interact and share content with other people that belong to their social network. These SNS are amongst the most popular web-based platforms currently, with the most popular Facebook, the most popular SNS, reported to have 1.11 billion monthly active users in early 2013 [4]. Section 2.1 gives an overview of the relevant background on the Internet and Section 2.2 gives an overview of various social networks and relevant research on collaboration using social networks.

2.1 Communication Networks: Internet

The Internet, a globally distributed *network of networks*, relies on multiple intermediate nodes belonging to different administrative domains to locate hosts and to route packets between any two end-hosts. Communication between any two end hosts is packet oriented and uses the most commonly used protocols on the Internet, IP at the network layer (IP) and TCP or UDP at the transport layer. Routing packets between different Autonomous Systems (ASes i.e., networks administered by different autonomous entities) itself is handled by routing protocols like BGP (Border Gateway Protocol). Applications such as the World Wide Web (WWW), email, and Voice-over-IP (VoIP), are in turn built on top of the underlying Internet Protocols (TCP/IP). Many of these applications also rely on other supporting infrastructure such as the Domain Name System (DNS) for their func-

tionality. The lack of a central administrative authority in Internet's distributed architecture implies that these applications have to rely on nodes (like switches, routers) belonging to different administrative domains to behave correctly. An analysis done in 2011 [5], gives the weighted mean Autonomous System (AS) count (i.e., distinct ASes traversed per path) for network paths as 3.

2.1.1 Actors on the Internet

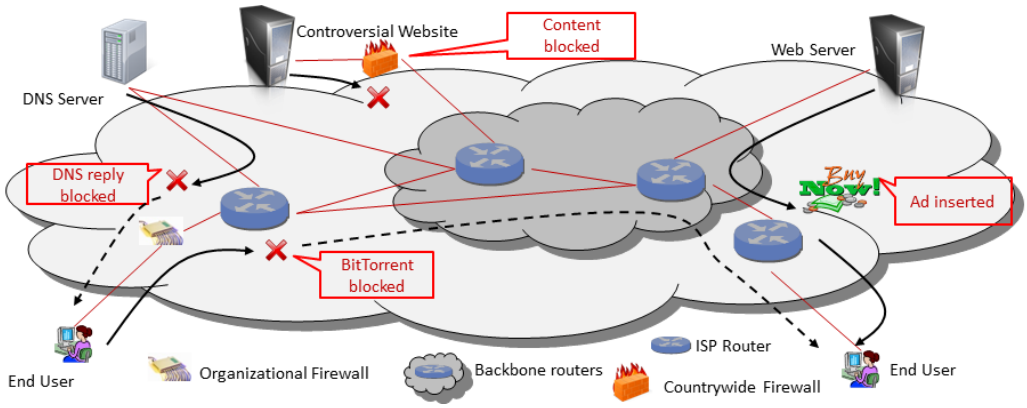


Figure 2.1: Actors and Network Manipulation on the Internet

Figure 2.1 gives on overview of the various actors on the Internet. As shown in the figure, any of the various entities along the path between a client on the edge of the network and the target end server can be the possible source of network manipulation. These entities include:

- **End Hosts:** These are the end hosts that individual users use to connect to the Internet. These computers can have software installed (either by concerned parents or sometimes by countries with restricted client software) that stops the users from connecting to certain websites.

- **Organizations:** Organizations frequently deploy a gateway that lies along the path between the users and the Internet. These organizations may block content considered illegal or detrimental to productivity of their employees. Firewalls also prevent incoming network connections to within the organization that might be considered spam or attack traffic.

- **Customer-level ISP:** End hosts and organizations connect to the Internet using an ISP. Sometimes organizations may even be multi-homed (i.e., use

multiple ISPs). ISPs have traditionally been very active in network manipulation. They have different motivations for engaging in network manipulation: throttling bandwidth for certain high bandwidth protocols or applications, blocking access to websites that might host illegal content, blocking potential spam emerging from compromised hosts within their network, and blocking or throttling access to content that might be in commercial competition with the ISP. Not all network manipulation by ISPs is considered acceptable and there is a growing demand by various advocacy groups that ISPs follow network neutrality.

- **Backbone ISPs:** Backbone ISPs (or Tier-1 ISPs) are in the middle of the Internet and have peering relationships with other backbone ISPs. Customer-level ISPs get access to the Internet through these Backbone ISPs. When packets pass from end-hosts belonging to two different customer-level ISPs, they may pass through one or more backbone ISPs. Backbone ISPs are also sometimes responsible for network manipulation [6].

- **Autonomous Systems:** Autonomous Systems are networks with a fixed set of IP routing prefixes under the control of a network operator. ASes have a single routing policy. Typically, ISPs (and large organizations) come under this definition with ISPs being assigned a Autonomous System Number (ASN). ASNs are used in BGP routing protocols to identify the AS to which a packet needs to be sent.

- **Domain Name System:** Domain Name System (DNS) is a distributed system that helps in translating a human readable URL to a specific IP address. While DNS servers do not by themselves engage in network manipulation, responses from DNS can be manipulated to send end hosts to incorrect or non-existent network locations.

- **Countries:** Country level firewalls are also a growing concern in the form of web censorship. Countries may wish to block access to content they consider illegal, or critical, or morally objectionable. Sometimes countries achieve this censorship, not by themselves, but by asking ISPs within their jurisdiction to implement it for them.

2.1.2 Traditional Threats to Network Communication

Traditionally, network communication problems on the Internet could be attributed to multiple reasons. Some of these problems were anticipated from the beginning and networking protocols had mechanisms to handle them. Over the course of the development and growth of the Internet, newer problems arose and many new mitigation techniques were developed to handle them. We given a broad categorization of these traditional threats below:

- **Capacity Issues:** Capacity issues are a frequent threat to network communication. These include congestion of an intermediate network link due to high network flows, congestion of network buffers at web servers due to high demand. Congestion problems have been anticipated from the very beginning and techniques, such as backoff mechanisms, are part protocols like TCP. These backoff mechanisms lead to well-behaving users reducing load by 'backing off' from making too many requests. ISPs have also developed different kinds of queuing mechanisms to ensure that a given flow does not end up utilizing all available bandwidth. Many new techniques such as distributed content delivery networks (such as Akamai) that allow content replication across multiple servers have also been developed to handle requests for content highly in demand.

- **Equipment failures:** Hardware failures such as router failures or link failures have also been anticipated from the beginning. Routing protocols can typically handle such link failures by maintaining a list of alternate paths to any given destination. ISPs also maintain multiple physical links from a given point of presence (PoP) so that there is some redundancy in case of a link failure.

- **Configuration issues:** Problems with configuration such as erroneous entries in firewalls or improperly advertised routing table updates are harder to handle automatically and may require manual intervention. For instance, in a famous incident in 1997 (AS7007 [7]), a faulty configuration of a router belonging to an Autonomous System (AS7007), led to incorrect BGP routing table updates being sent to the rest of the Internet. This caused a routing black hole, where a lot packets destined elsewhere were routed towards AS7007 and ended up being discarded there. Network operators have since come up with policies to mitigate propagation of erroneous routing updates,

and there have been many research proposals to handle BGP updates more securely [8].

However, these proposals have not been adopted in general and it is still an ongoing threat. For instance, recently in February 2008 [9], a faulty BGP update was sent by AS17557 in Pakistan to the rest of the Internet, which lead to hijacking of traffic sent to YouTube.

- **Attacks:** Deliberate attacks on the infrastructure are an ongoing threat to network connectivity. One example is a distributed denial of service (DDoS) attack where attackers use end nodes to send huge floods of packets towards a given destination overwhelming the capacity of the target server or the capacity of some network link along the way. Many techniques ranging from over-provisioning, filtering of packets, and burdening the attacker have been proposed and implemented to counter such DDoS attacks.

  Despite these measures, DDoS is an ongoing concern. It is particularly troublesome as DDoS attack directed towards one server may end up doing collateral damage to other services. For instance, in April 2013, large DDoS attack targeted towards SpamHaus [10] ended up also affecting other Internet Exchanges in London and Amsterdam.

2.1.3 Emerging Threat: Network Manipulation

While many of the threats to network connectivity are either accidental or due to deliberate actions by attackers at the edge of the network, an emerging threat is the *deliberate* interference of network connections by administrative entities in the middle of network. This increase in network manipulation has led to a push back from various stake holders to prevent such manipulation. Advocacy groups such as the EFF and regulatory authorities such as the FCC in the USA have tried to limit manipulation from ISPs [11]. However, the legal situation has not yet been clarified with the ISPs claiming that they have a right to network manipulation [12] and the advocacy groups claiming the proposed curbs are not adequate [13]. At the level of country-level censorship, Reporters Without Borders, observe March 12th as "World Day Against Cyber-Censorship" and give yearly reports on the state of web censorship across different countries. Their latest report [1] noted 12 countries (including China, Saudi Arabia, and Vietnam) as Internet Enemies and a further 14 (including Australia, India, and France) as countries under surveillance.

Unfortunately, HTTP and TCP/IP protocols do not provide enough diagnostics to end users to accurately detect the presence and source of such network manipulation. Due to the implicitly trusted nature of many of these protocols, it is often possible for the actors in the middle to deceive the end users into assuming that the server at the other end is not available, or that the content is unavailable. They have various actions at their disposal such as dropping packets as well as sending fake reset packets to close the connection. We describe the triggers that lead to network manipulation in Section 2.1.4 and the mechanisms of such attackers in more detail in Section 2.1.5.

Case-studies, Country-level blocking: Consider the following scenario which occurred recently. The Australian government asked ISPs within their country to block certain websites (and IP blocks) which were considered undesirable. The list of these blacklisted URLs were not made public and included a false-positive, an educational website hosted in the USA which fell under one of the blocked IP ranges. To further complicate matters, only some of the ISPs followed the Australian government's request and blocked the blacklisted IPs whereas other ISPs did not. An end user who is on one of the ISPs that are blocking the IP address will not be able to know if the website is down or being blocked (if so, by whom). In fact, it is possible for the ISPs to be more sophisticated with their blocking and give false HTTP status codes (page not present etc.,) instead of simply dropping the packet. Eventually, the Electronic Frontier Foundation (EFF) was able to detect the presence of blocking from certain ISPs by collecting traceroutes from network operators in Australia using different ISPs [2]. Note that technical analysis alone would not have implicated Australian government as a possible cause of network manipulation (as some ISPs within Australia were not engaging in censorship). The EFF were able to confirm that the blocking was due to Australian government's censorship list (and not say policies of a given ISP) only by talking operators of one of the ISPs. An individual user on their own would not have the network visibility to be able to make such inferences.

Case-studies, ISP level blocking: Free VoIP has made international calling cheaper to many users across the world. This also meant that VoIP is a threat to traditional public-switched telephony networks (PSTNs) which rely on revenue from high costs of international calling. In many places, PSTNs also work as customer-level ISPs. This leads to a direct conflict of interest, where free VoIP services are seen as revenue threats. In many countries, PSTNs are also managed by public-sector companies managed by their governments. This threat of rev-

enue loss lead to many countries blocking ports or network flows associated with VoIP [14]. Blocking is also used by ISPs to present their own services as cheaper or more reliable than those of their competitors. For instance, in 2006, a Canadian ISP (Shaw Communications) recommended an additional fee of 10$ from their customers who use a free VoIP service, Vonage. This lead to complaints from Vonage that it was an anti-competitive measure [15].

2.1.4 Network Manipulation: Attacker Triggers

Network manipulation may be enabled by various triggers. We list the most common triggers discussed in literature [16].

- **IP header:** A simple trigger for blocking or throttling is the source or destination IP addresses (or IP address blocks) of a packet. For instance, an organization may maintain a white-list of IPs that users from within their network can access and deny all other connections. Countries or ISPs, likewise, can block access to certain servers by maintaining a black-list of IPs that are not allowed. This is a popular mechanism due to its ease of implementation. IP addresses are at fixed locations on a packet and many routers provide efficient real-time mechanisms for blocking connections based on IP addresses.

- **Transport protocol headers:** Many applications on the Internet use a default port number of communication. ISPs can use these port numbers to block applications, such as VoIP or BitTorrent, that they consider undesirable. Port numbers are also a popular trigger to block content due the relative ease of implementation.

- **Packet Payload:** A more recent trigger for network manipulation is the packet payload itself. This was initially not trivial to implement due to the fact that examining the packet structure for content from different applications required more time and was not practical at routers which required packet flows to be at line speed. However, advances in hardware [17] have made it feasible to perform deep packet inspection (DPI) without sacrificing speed. ISPs can perform DPI to identify application flows that need to be blocked. Countries can perform DPI detect specific keywords in HTTP and block corresponding connections.

- **Time of the day:** Another trigger for traffic differentiation can be time of day. Customer-level ISPs see increased usages of their network during specific times of the day and may be relatively free at other times. ISPs can perform traffic differentiation on high bandwidth content only during those times of heavy load, so that other customers can have a fair share of the bandwidth.

- **Network Load:** ISPs may differentiate only when they are under heavy load.

- **User behavior:** ISPs may block or rate-limit a user with heavy bandwidth consumption.

2.1.5 Network Manipulation: Attacker Mechanisms

Different attackers have different capabilities for network manipulation. For instance, organizations can have higher fidelity information about network identities of individual users within their organization and therefore can have fine-grained policies for network access from within the organization. Whereas, a country level firewall may know nothing about the exact user making the request but can have broad policies blocking access to large group of users. In CORDON [18], this is referred to as the relationship between *resolution* of the censor and their scope of *influence*. We list some of the various ways in which attackers can manipulate the network connection.

- **Dropping** packets is a popular way of network manipulation, especially when the attacker is implementing a blacklist. In this, the attacker simply does not forward packets that triggered the manipulation towards to the target. The triggers could be IP headers, TCP headers, or even specific keywords in the packet. This usually lead to end hosts timing out of the connection as they do not receive any response from the target node. This connection time-out is often perceived by end-users as the result of the target web server being off line or being unresponsive due to heavy load.

- **De-prioritizing** packets is a popular way for ISPs to throttle undesirable applications that their customers are running. These might be high-bandwidth protocols such as BitTorrent or may be network connections corresponding

to applications that are commercially in conflict with some service the ISP is offering (i.e., VoIP or streaming content).

- **Protocol-level manipulation** is a popular way for attackers to influence the behavior of the end hosts. For instance, instead of simply dropping a HTTP request to a web-page that they want to block, they can instead send a fake HTTP status code back to the client that indicates that the page is unavailable on the web-server. Instead of explicitly throttling the packet transmission rate, the attacker can modify the TCP window sizes from the client so that the server will send content at a reduced rate. Frequently, ISPs may insert reset (RST) packets masquerading as a packet sent by the other end of the connection [19]. These techniques have the advantage of being stealthy and being hard to detect.

- **Content manipulation** involves attackers explicitly manipulating the content of the payload of packets. For instance, this was seen with ISPs inserting their own ads into web pages requested by their customers [20]. This is harder to detect, as often, differences in content may be due to benign reasons. For instance, web servers often serve localized content to users based on their geolocation.

2.2 Social Networks

A social network is a network made up of individuals who either know or have a working relationship with other people in the network. It can be viewed as a graph whose vertices correspond to users and vertices are connected by an edge when the two corresponding individuals have some kind of relationship, often called a *tie*. Vertices within one hop on the graph are friends and within two hops are friends of friends and so on. Social network ties may be strong or weak ties [21] depending on whether a friend is trusted or is a loose acquaintance. It was suggested that strong ties are more likely to be reciprocal than weak ties [21].

Online social networks (OSN) such as Facebook, Twitter, LinkedIn, Google-Plus are very popular platforms for users to keep in touch with their friends and to share content. These platforms have shown an increasing rise in popularity, with *network effects* causing people to migrate towards OSNs with large user bases. For instance, a recent study by Facebook [22] found that, the average distance

between any two active users on their 721 million social network is 4.74 hops. Even more impressively, this distance has *shortened* from 5.28 hops in their earlier study in 2008. The median number of friends for Facebook is more than 100.

2.2.1 Types of Social Networks

There are different kinds of social networks, depending on how the ties between different users is established. These differences mean that each social network may have their own biases. For instance, a social network like LinkedIn is mainly used for professional networking, whereas Facebook is more commonly used for personal sharing. Here we give a brief discussion of various types of social networks:

- **Sharing-based** social network is the most common form of an OSN and includes platforms such as Facebook and GooglePlus [23]. They allow users to share their statuses, locations, photographs etc., with other friends on the social network or even the general public. These are amongst the most popular social networks with the largest number of users. A survey by pew research in December 2012 [24] found that 67% of online adults in the US used OSNs, in particular Facebook. It also found that the users are biased towards younger population (83% of users with ages 18-29 used an OSN as compared to only 32% of those aged 65 or more).

- **Collaboration-based** social network consists of a network of people with ties between people that collaborated with each other in some task. For instance, these include bibliographic networks like DBLP [25] and Arxiv [26] where two people are connected in the social network graph if they have co-authored a paper together. These social networks are obviously biased in favor of users who engage in that task. They might also include other biases, for instance users on an Arxiv based social network will consist of academic users (mostly in the fields of mathematics, computer science, physics etc.,) and who frequently have access to the Internet through an academic network. The collaboration based social networks are frequently utilized in academic research due to the fact that the data is readily available for analysis. In our thesis, we analyze data using a particular bibliographic network on Arxiv [27] which consists of authors who published in the field of High-Energy Physics.

- **Location-based** social networks are another type of social network, that allow users to share their current geographic locations with their friends. Users are given a list of venues by the social networking website that are nearest to geographic location of the user. The geographic location of the user is determined by using the GPS function of a mobile phone and the check-ins are done using either a mobile website or a mobile app. Therefore, these social networks are likely to be biased towards people using mobile smart phones. In our thesis, we used datasets belonging to two different kinds of location-based OSNs, Gowalla and BrightKite [28].

2.2.2 Collaboration on Social Networks

There has been research on utilizing a user's social network to accomplish tasks. This can be contrasted with crowdsourcing in systems like Amazon's Mechanical Turk [29], which allows human solvable problems to be distributed to large number of (random) users. Friendsourcing is useful when the problem to be solved either requires data that is available only to friends (and not the public at large) or when friends are more likely to help problem compared to random strangers.

Collabio [3] utilized friendsourcing to gather personalized tags about a user. This tagging was presented to the user as a game with points being accumulated if the tags given by a user to their friend matched those given by other friends. This game mechanism allows the tags to be more accurate and increases the interest of users in participating in tagging. The authors found that while popular tags for a user can be found by strangers as well as friends, uncommon but accurate tags were much more likely to be found by friends.

A recent study [30] of 624 people found that people are likely to ask their social network, instead of a search engine, for questions that may be subjective or require expertise. It also found that up to 37% of users listed altruism as one of the motivations for answering the queries of their friends. It seems obvious to suggest that in terms of altruism, people are more likely to to help their friends as opposed to random strangers (and that closeness of a friend was motivator to answer a question). This was also measured in a study [31] that compared altruism as shown towards a friend versus a random stranger in a modified dictator game. A dictator game involves a user unilaterally deciding how to split some reward between the user itself and another participant. It is found that even amongst

random strangers there is some altruism, that is, users do not keep all the reward to themselves. The authors found that users were likely to give 52% more to friends compared to strangers. Furthermore, when the game is set up to have future interactions, the giving to friends went up by another 24%. This suggests that, at least for certain tasks, friends are more likely to help than strangers.

Tie-strength can also affect the quality of information received from friends. The original study in 1974 [32] found that weak-ties are better than strong ties in the context of users seeking jobs. This was due to the realization that users who are weak-ties are more likely to be in diverse circles and may have more interesting leads in terms of available jobs. However, a recent study in 2012 [33] looked at the effect of tie-strength in the context of answering questions posed on OSN. They found that stronger ties (as measured by the framework developed in [34]) provided a slight increase in information compared to weaker ties.

Another study [35] has found that in human social networks cooperative behavior cascades to more than one hop, to as far as three hops. Finally, an interesting study [36] found that centrally located individuals in a social network are more likely to be cooperative. These results suggest that we can get good cooperation from a social network by efficiently reaching out to centrally located users. They are a) more likely to help us, and b) because of their central location, they will reach other more cooperative nodes.

2.2.3 Trust on Social Networks

An important consideration while using social networks is the idea of trust between users and whether there is transitivity of trust on social networks. This is difficult to evaluate in general, as there are different notions of trust (trust in someone's recommendation versus trust in someone's action etc.,) and they are not always explicitly specified by the users.

Claims have been made that trust is not transitive [37], as well as that trust is somewhat transitive but distrust is not [38]. In OSNs, and in particular recommender systems, the assumption of trust transitivity helps in generating trust for users who are starting with the system. The recommendations of users trusted by a new user can be used to give recommendations to a new user. Consequently, there have been a lot of algorithms that assume trust transitivity to generate trust values between two users not directly connected [39, 40].

Transitivity of trust on a social network has also been experimentally verified in some specific domains. In particular, algorithms that assume transitivity while propagating trust have been very successful in predicting the opinions of the users. For instance, [41] studied the dataset for www.epinions.com, where users may review items, other reviews, and indicate trust and distrust of other users in the system. The paper assumed transitivity of trust between users to perform a matrix operation based propagation of trust between users who are not directly connected by any existing trust relationship. The authors were able to validate their algorithm by correctly predicting an existing hidden trust relationship around 85% of the time.

In a similar vein, [42] also tried to predict whether users trusted or distrusted each other. Instead of matrix operation based propagation of trust, the paper used a machine learning approach to infer the trust relationships between users who are not directly connected. The paper used joint relationships (whether a user in the middle trusts or distrusts two unconnected individuals) as inputs to their machine learning algorithm. They used wikipedia moderator election data, slashdot.org friend of foe data, and www.epinions.com trust data to evaluate their algorithm. They had a high prediction success rate (around 80% to 90%) when predicting an existing hidden trust relationship.

Another paper, [38] used the same datasets, but used a different mechanism for propagation of trust. This work used a random graph as a model of users where edge probabilities were inferred from existing trust relationships. By assuming trust transitivity, and using a spring embedding algorithm (where users with a shared friend are pulled together and users with a shared enemy are pushed apart), their algorithm was able to predict 80% to 89% of the hidden trust relationships.

These results indicate that, at least in certain social networks, the notion of trust transitivity works. These are encouraging for friendsourcing where we may want to go beyond immediate friends when asking friends for help.

Chapter 3

Related Work

Network manipulation, in the form of network neutrality or censorship, has engaged the interest of a number of researchers. Research in this domain has concentrated on boththeniques for detection of the manipulation as well as in developing mechanisms to overcome it. In this Chapter, we give an overview of some of the recent research in both these areas.

3.1 Detection

There have been many interesting approaches to detection of network manipulation over the years. These approaches differed by: a) the strategy for acquiring agents that do the analysis, b) the target suspect for manipulation (most commonly ISPs, more recently countries), and, c) the end user involvement. In this section, we give an overview of the various techniques and characterize them by these aspects.

3.1.1 Crowd-sourcing using measurement-servers

A common strategy involves crowd-sourcing the measurement of network manipulation, especially when the target suspects are ISPs. Often, the analysis is done by users sending specific packets to a given a set of measurement servers corresponding to the tool. BTTest [43] studied the incidence of blocking BitTorrent protocols by ISPs. In this case, users ran a Java Applet which emulated the BitTorrent protocol in talking to a measurement server. In case the connection was closed (and was not due to RST packets sent by the measurement server), it was concluded that there was explicit manipulation by ISPs along the way. BTTest found evidence of relatively high amounts of blocking (8.2% of their result sets).

Glasnost [16] is another user-driven tool that measures certain kinds of network neutrality violations by ISPs. Glasnost detects traffic differentiation based on transport protocol headers (for instance, certain Peer-to-Peer(P2P) ports) or packet payload (higher-level application protocol messages) by doing a differential analysis. The end user connects to a measurement server and runs a couple of network flows. One flow is run with specific application parameters (for example, BitTorrent) and another flow with control parameters is run with a random payload. This was also implemented as an website with a Java applet performing the set of measurements.

Netalyzer [44] is another user-driven tool (using Java Applets) that runs a set of test-suites by contacting a set of measurement servers. It can detect port filtering, DNS manipulations, path MTU and other features.

ShaperProbe [45] is another tool which detects traffic shaping by ISPs. Many ISPs offer burst speeds with high bandwidth at the beginning (measured in seconds) of a network connection, but will lower bandwidth later on. These bursty speeds will help users to download short files quickly, while not giving priority to users using long-lived flows. ShaperProbe is a platform-dependant executable, which when run by a user sends traffic to a specified measurement servers. The measurement servers look at the traffic flow rates to estimate at which point (in seconds) during a network connection the ISP will implement traffic shaping to downgrade the connection speed.ShaperProbe probe found many ISPs within the US implementing traffic shaping (not all of whom had advertised it).

In all of the above cases, the implementation involved a tool which could talk to a set of measurement servers belonging to the tool. Note that, as these were using to detect manipulation of specific protocols (such as BitTorrent) by the ISPs, the user involvement was limited to running the tool that came with a predefined set of test suites. In particular, this strategy would be less effective when the network manipulation is specific to a set of targets not involving the measurement servers, as in case of a targeted website censorship.

3.1.2 Passive Aggregation and analysis

In contrast to the techniques above which involved active measurements by end users, there have also been techniques that involve collecting data from a user and performing an off-line analysis to detect patterns of manipulation. NANO [46]

performs detection of general network neutrality violations by performing causal inference on passive data collected from a large number of agents. The measurement data collected from all the nodes is analyzed to detect if packets of a given application (or destination) face performance degradation when passing through a particular ISP compared to other ISPs.

HerdictWeb [47] is a crowdsourcing site that allows users to report problems with a specific web site and check if other users are facing similar problems with that web site. It also maintains a timeline of reports for a given web site, allowing a user to detect historical patterns in site access, say, from a given country.

3.1.3 Piggybacking on P2P networks

Another technique for acquiring users to perform the analysis involves piggybacking on P2P protocols to perform the analysis. For instance, [48] utilized clients on Gnutella to perform their analysis. In this case, they insert a rogue super-peer which redirects normal users to try connecting to their measurement server on a given port number. The probe requests from other participants using Gnutella to a measurement servers with given port numbers is used to study the incidence of blocking based on port numbers.

Dasu [49] is another approach for characterizing the performance of ISPs. Dasu runs as a plug-in in the user's BitTorrent client. It takes passive measurements of the user-generated traffic to get an accurate assessment of the ISP on real-life traffic (as opposed to measurement servers). Dasu can also do certain limited measurements to detect some ISP manipulation such as manipulation of DNS replies.

3.1.4 Automated Analysis

Another approach is to perform an automated analysis that involves clients running automated test suites on large distributed infrastructures (such as PlanetLab and MeasurementLab).

NVLens [50] and NetPolice [6] analyze traffic differentiation in backbone ISPs by sending probes with varying parameters from a distributed set of end hosts to some selected destination nodes. Aggregate loss rates are then collated by a central node to uncover suspect ISPs that might be responsible for the differentia-

tion. In contrast to HerdictWeb, which is entirely based on user reporting, CensMon [51] automates the detection of censorship to a given URL. Upon receiving a URL, a centralized server forwards it to a distributed set of agents running on PlanetLab in various countries. It then collects and reports the results of all the individual agents attempting to access the URL including DNS replies, URL filtering, content on the web page, and other information.

3.1.5 Other Approaches

Web Tripwires [20] is another approach that detects modification of in-flight http packets. The authors observed in their experiments that the web pages were modified to insert advertisements, code to block pop-ups, and even malicious code. Web Tripwires is a piece of client-based Javascript code that can execute on the client's browser when loaded with the page and perform basic integrity checks to notify the user if his page was modified in flight.

3.2 Circumvention

Due to the increasing rise of censorship, a lot of recent research has also focused on circumvention of cyber censorship. A common mechanism for circumvention of censorship uses web-proxies (such as [52]). However, this leads to *proxy discovery*, where the censor might be aware of the proxy locations and block it entirely. Key space hopping [53] is a technique developed to ensure that the client (and, therefore, a censor masquerading as a client) will only know some small fraction of the total number of proxies. Infranet [54] uses cooperating web servers, acting as proxies to clients, to deliver censored content to the end user. In it, the end user uses a covert channel in the HTTP request sequence to request a censored web site. The co-operating server replies via images which embed the censored content using steganography. Another way of hiding censored content using steganography was proposed in Collage [55]. In it, the authors utilize commonly available user-generated content sharing sites (such as photo-sharing sites) as a cover to request and deliver censored content (using steganography). Finally, there have been multiple recent research papers [56, 57, 58] that suggest getting rid of end-host proxies altogether and using instead cooperating routers on the path to act as proxies to the censored client.

Chapter 4

Design Options

Any user, who uses a system that relies on friendsourcing to detect network manipulation will go through a set of broad common steps. However, the actual realization of a system, as well as the implementation of each of these individual steps, will depend greatly on the desired features from the system and the assumptions made about the various parties involved.

In this chapter, we give an overview of these design choices and the various factors that effect these choices. Firstly, we describe the broad steps involved in the friendsourcing approach in Section 4.1. We describe a set of desirable features of any such system in Section 4.2. We also list a set of assumptions that can influence the design choices in Section 4.3. Finally, Section 4.4 describes the various choices in design of such a system, and a discussion of how each of those choices is affected by the features that need to be supported and the assumptions that are made.

Our prototype implementation corresponds to only one point on this design space. We give a description of the choices we made in our implementation in Chapter 6.1. In our analytical evaluation in Chapter 5, however, we look at multiple design choices and look at the effects of those choices on the efficiency and correctness of the system.

4.1 Conceptual Phases

Independent of the actual design choices, any user attempting to detect network manipulation using friendsourcing, will go through a set of common phases. Figure 4.1 shows the various phases involved. We give a description of these phases below.

- **Registration:** Although not shown in the figure above, an important part of the system is the registration of the user to the system. Depending on

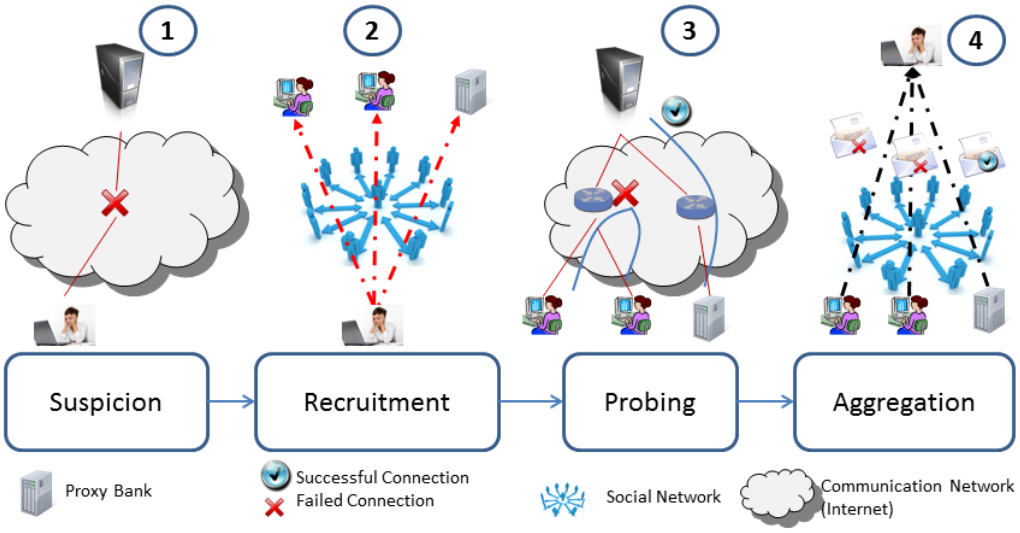


Figure 4.1: Conceptual steps in Friendsourcing

the design, this may involve the user going to a website, or running a tool. This phase may also involve collection of network attributes of a user (such as which ISP the user belongs to) either automatically, or by explicitly requesting the information from the user.

- **Suspicion:** This is the first real step in a user utilizing the detection system. In this phase, the user suspects that there are problems with their network connection. This may be a straightforward step invoked by the user based on lack of network connectivity seen by the user. Alternately, it may be trigged automatically by a tool running diagnostics on the user's behalf. In cases such as a degraded network connection, the user may not even be sure that any manipulation is taking place, but only wanted to confirm that it is not occurring.

- **Recruitment:** In this phase, the user requests help from their friends to perform the probing. In the simplest case, it can be done by the user asking all of their friends to perform the probes. Alternately, the user can decide on a specific group of friends, depending on the various criteria; availability of the friend, history of cooperation from a friend, or other network location of the friend. In some designs, this choosing of a subset of friends may be

automated by the tool itself. The recruitment may be done by the tool using the API of a social network.

- **Probing:** The recruited friends send a probe, i.e., a network request towards the target. The results of the probe could be a simple success or failure, or include more elaborate responses such as protocol response, timing failures, performance metrics (such as latency). In general, these probes could be triggered manually by the users or automatically by the tool on behalf of the user.

- **Aggregation and Analysis:** In the final phase, aggregation and analysis, the results of all the probes from all the recruited friends are gathered together. This aggregation and analysis could be done at a centralized authority or by collecting the probe results at the originating user who initiated the request. The results of the analysis may give us the identity of the manipulator or may result in the realization that more information is required to explicitly identify the suspect. In case more data is required, the user may go through the recruitment, probing, aggregation phases again with an expanded set of friends to perform the analysis.

4.2 Desirable Features

Besides the basic requirement of detecting network manipulation, we also have some features that would be desirable in any design and its implementation. Note that some of these features may be mutually exclusive when it comes to actual implementation.

- **Ease of use:** The tool should be usable by general non-technical users. Ease of use comprises of multiple factors. Firstly, using the tool should not require any particular technical know-how regarding computer networking. The user should get understandable results using the tool without having to utilize perform network diagnostics on their own. This also means that the tool should be cross platform. If possible, the tool should not require separate installation (or privileges for such an installation) on the user's machine.

- **Generality:** As far as is practical, the system should be capable of detecting a *general* class network manipulations. First, the system should be able to narrow the collection of entities likely to be responsible for violations. The user should be able to detect if it is his/her organization, ISP, country, or any other upstream administrative entity that is responsible.

- **Extensibility:** As listed in Chapter 2, traffic differentiation can occur based on various factors including transmission protocols, application protocols, packet payload or even the destination of the packet. Ideally, the system should have the capability to detect a wide range of network neutrality violations. Given that the list of potential violations is unlikely to remain static, it is desirable that the system be *extensible* with modules that can address new violations as they arise over time.

- **Effective Recruitment:** The system should be capable of performing an effective recruitment of a user's friends to perform the inference. Effective recruitment implies that the friends selected are the most useful for detecting the presence and source of manipulation. It also implies that the friends selected are available to help and they are most likely to help.

- **Efficiency:** The sensor recruitment system should be *efficient*. There are different ways of measuring the efficiency of the system. The system should minimize the number of probes being sent to the target destinations. It should minimize the number of messages that are passed for the sake of recruitment of friends and the number of messages the are passed to collate the data from those recruited friends. It should also perform this network manipulation detection efficiently.

- **Graceful Degradation:** The design and the implementation should allow for a graceful degradation in case of limited capabilities of the user. This could be in multiple aspects. Firstly, if a user cannot run the full set of tests from the tool due to limitations of their platform, the system should still allow the user to gather some meaningful results. Secondly, even if the user does not have enough friends who are current participants in the network manipulation detection system, we would like the user to have some utility to the user. This is an especially important characteristic to have for new users. If a new user has no benefit from using the system, it leads to a bad *network effect*; new users will not joining the system as they will not get

any benefit from using the system and future users will be less inclined to join the system due to the lack of other helpful users in the system.

4.3 Assumptions

Friendsourcing, as an approach to detect network manipulation, makes certain fundamental underlying assumptions independent of the design choices. A particular design can also make further simplifying assumptions, which are not strictly necessary for friendsourcing to work, that may make the resultant implementation easier. Here we list both these kinds of assumptions.

- **Safe Probing:** We assume that a user does not have to face any repercussions for making the probes. For instance, the user does not get penalized by the ISP (or government) for checking if a given site is accessible. Also, this implies that the very act of making a probe, does not push the user over some network capacity quota limit that the ISP might enforce.

  Note that in case of wireless mobile platform this assumption may not be true for all users. Many users are on a metered data plan and may not be willing to help their friends extensively due to the associated cost. Furthermore, in some countries such as Syria, users may be arrested for accessing websites that are considered critical of the government [59]. People may also be unwilling to use the system in certain countries due to existing policies preventing people from circumventing the censorship. For instance, an internet store owner was arrested in Iran [60] for selling software that allowed circumvention of blocking within the country. Finally, there have been attempts at laws making it a crime to access certain adult content [61, 62]. This will lead to users not willing to risk sending probes even for their friends.

- **Communication Path:** We assume that users on a social network have some framework by which they can communicate, both for the purpose of recruitment as well as for sharing results. We also assume that the network service providers do not interfere with the social network messaging of the users.

  This particular assumption also affects the actual implementation. For instance, if each individual user needs to run a program to listen to their

friends, it may not be practical as many ISPs do not allow their users to run services which can be reached via incoming connections. Instead, using mechanisms from an OSN to perform this communication will be more practical.

- **Consistent Policies for Manipulation:** We assume that the various manipulators consistently apply their policies for manipulation across their domain. For instance, we assume that if an ISP is blocking connections to a website, it will not apply the blocking only to some users and not the others.

- **Stable Path:** A reasonable, but not necessary, assumption that can be made is that a packet always takes the same path to its destination. More accurately, it can be assumed that a packet always passes through the same set of outgoing administrative domains of a user. We may also assume that the user has a way of recognizing these administrative domains. This assumption is generally a reasonable one. While packets on the Internet may take various paths to reach a server, the set of possible outgoing administrative domains that they traverse through are generally fixed.

  This is particularly true when we consider the possible sources of network manipulation. The organizational firewalls, network service provider and the country of a given node on the network are likely to be very stable. We will treat a user using multiple computers on different networks separately.

  In case of multi-homed organizations with multiple ISPs, however, this assumption may not be true. In such cases, we may need multiple probe requests to correctly identify the problem.

- **Trustworthy Social Network:** Another simplifying assumption is that a user's social network is trustworthy and does not consist of malicious friends who might be sending incorrect results to mislead the user. In reality, there might be many sybil users on the social network. These are users who are created by the attacker with forged identities. However, unlike in crowdsourcing, subverting the analysis results is not straight forward for sybil users of the system. The users affected are only those who are closely connected to the sybil network via their social network ties. In particular, if all the friends within a maximum hop-count radius of the originating user are honest, the system will give accurate replies. Other techniques have been

developed recently which make it easier to detect sybil networks within social networks [63].

4.4 Design Choices

As discussed earlier, there are many different design choices in implementing friendsourcing to detect network manipulation. In this section we discuss many of these choices and how they affect the various phases in friendsourcing

4.4.1 Coordinated versus Decentralized

Broadly, we can realize our friendsourcing approach with two different strategies. We could have a *decentralized* system with friends as distributed nodes and friendship ties as the connections between these nodes. In this approach, we do not have any centralized authority which collects or analyzes the presence of network manipulation. The system can use the infrastructure of an OSN to actually propagate the probe requests to friends and to analyze responses from such probes. There is no central authority to store results or to perform the aggregated analysis. We analyzed the effectiveness of a distributed friendsourcing architecture in Section 5.4.

Figure 4.2 gives an overview of such a system. As shown in the figure, a user initiates the friendsourcing when they cannot accesses a particular web server. In this example, the network connection is being blocked by the firewall of the organization to which this user belongs. The user recruits friends (who in turn ask their friends) to perform the probes. The results of these probes are passed back through the friendship ties to the originating user. This user can collect and perform a local analysis to figure out the cause of the network manipulation.

Alternately, we could use a centralized *coordinated* architecture which utilizes a trusted trusted entity in charge of collection of data, analysis of detection, and recruitment of friends. This approach has its central weakness,i.e., being a single point of failure. However, it also has many advantages including ease of deployment, ease of use, and efficiency for end users. Consequently, it is the most common approach seen in literature [16, 46, 44]. Note that although we mention a centralized architecture, it may actually be implemented as a distributed collection of servers which are conceptually performing the role of a centralized coordinator.

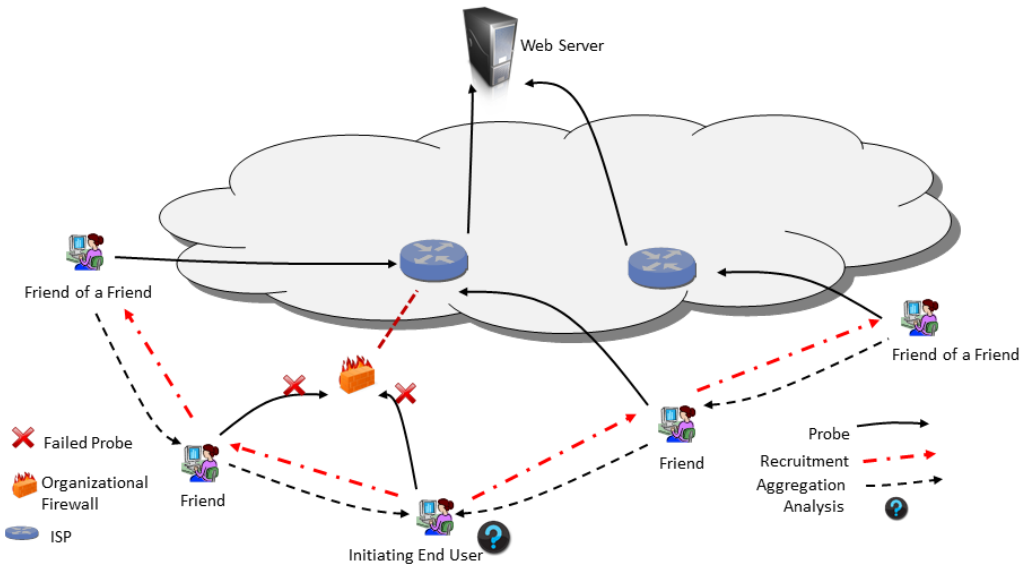


Figure 4.2: Distributed Design for Friendsourcing

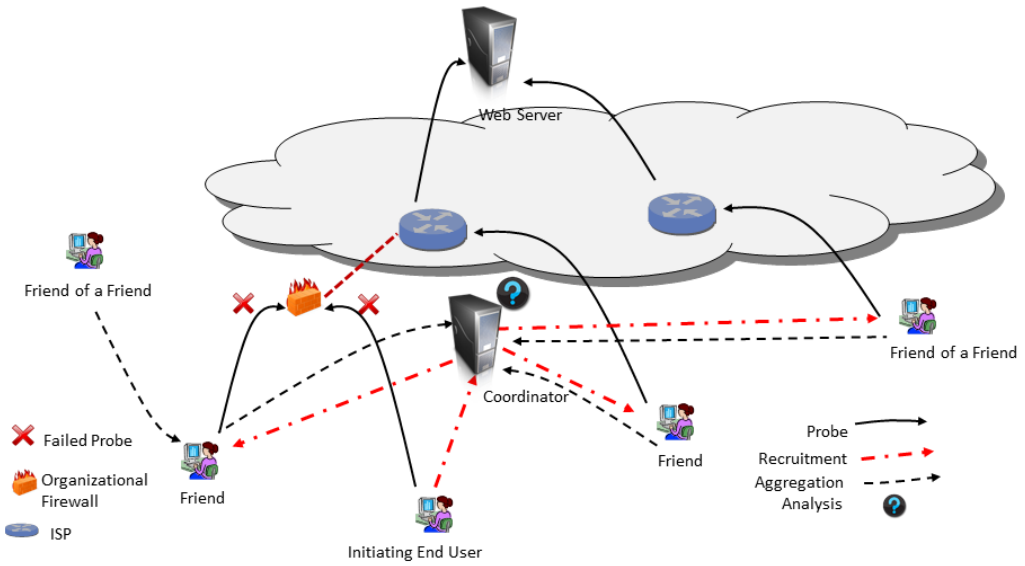


Figure 4.3: Coordinator based design for Friendsourcing

Figure 4.3 gives us an overview of the various components involved in such an approach. As shown in the figure, a user initiates the friendsourcing when they cannot accesses a particular web server. This request is sent to the (conceptually) centralized coordinator. In this example, the network connection is being blocked

by the firewall of the organization to which this user belongs. The coordinator recruits friends which it considers likely to help the user (either due to their network locations or due their previous history of interaction with the originating user). These friends perform the probes and sent the results back to the coordinator. The coordinator performs the aggregation and analysis and sends the results of the analysis back to the originating user.

We discus how this design choice affects the various phases below:

- **Recruitment** In a completely decentralized system, recruitment follows the friendship ties directly. This can be implemented in different ways. In a purely distributed environment, we can imagine every user running a daemon listening to incoming requests from their friends. This is not always practical due to policies by ISPs blocking such incoming requests to users. Alternately, we can use the services of an OSN which have APIs [64] to allow communication between friends. This strategy while being more practical and usable, will be vulnerable to an attacker blocking OSN communication. In contrast, in a coordinator based approach, the recruitment is done by the coordinator choosing the set of friends to ask for help.

  Another aspect to consider is the overhead involved in recruitment. This overhead occurs in both the number of messages being sent as well as in the time taken for these messages to be sent to all the users. Absent a global coordinator, a simple strategy for recruitment is to ask all friends who may in turn ask other friends for help. This simple strategy, however, is not very practical as the size of modern OSNs results in a huge number of such requests even when simply restricting ourselves to friends of friends (Facebook, has on average 20,000 friends of friends per user). Also, because of a lack of coordinator, mutual friends of an originating user may be receive multiple requests adding to the messaging overhead. Also, because each request may be forwarded to friends of friends and so on, we have also have a timing overhead proportional to the depth of recruitment. In a coordinated design, the coordinator can, in one step send requests to all those friends (and friends of friends) in one single step. This gets rid of the multiple stage recruitment that needs to occur in a purely distributed architecture. It also gets rid of the inefficiency of sending the same request multiple times to mutual friends of an originating user.

Finally, a design choice that needs to be made is whether the recruitment is automated by the tool or done manually by the user. A user may not always have the best knowledge of the network characteristics of their friends and therefore may not perform optimal recruitment for the task. Alternately, however, a user is likely the best judge of the strength of friendship. Therefore, they will select friends who are more likely to respond to their requests.

- **Aggregation** In a decentralized system, the probe results need to be sent back to the originating user for analysis. This will typically follow in inverse order of the messages sent for recruitment. This has all the messaging and timing overheads associated with recruitment. In a coordinated design, the results of the probe are sent by the users back to the coordinator. The coordinator can perform the aggregation and analysis and send the results back to the originating user.

  This increases the efficiency in multiple ways. Firstly, there is less overhead in the direct aggregation of the results using a centralized coordinator. More importantly, results of an analysis may be stored and, if recent, given to further requests from other users. This can have a significant improvement in efficiency if, for e.g., a popular website is suddenly inaccessible. Instead of multiple users making requests which result in a large number of probes being sent, the coordinator can share the analysis of the failure (and possible source) to subsequent users (within a short time period).

A decentralized approach has many apparent advantages. The most important one is the fact that there are centralized targets for the attackers. An attacker who wants to block the detection system, cannot do so by simply blocking access to the system. In Chapter 5 we give our analysis of evaluating a distributed architecture as well as some optimizations that can improve the efficiency of such a system.

Although, a centralized system is vulnerable to being blocked by an attacker, as discussed it has many advantages in terms of implementation, ease of user, and efficiency for end users. In our prototype evaluated with real-life users, we therefore used a centralized coordinator approach. We describe our implementation in more detail in Chapter 6.

4.4.2 On-demand versus Tool-driven analysis

Another design choice we need to made is to choose whether we have a design supporting an *on-demand* analysis where the users can dynamically choose the analysis to perform, or a tool-driven analysis where the tool keeps listening to network connections and uses friendsourcing when it sees potential problems. An on-demand analysis will allow a user to use the system whenever the user suspects there is a manipulation. This is especially useful when there is reason for a user to suspect that there is content manipulation occurring in the system. A tool-driven analysis may not have the necessary context to detect that there is any problem with the network connection as it sees no flaws in the network connection. On the other hand, a tool-driven analysis may be more accurate at detecting subtle network manipulation such as those that involve traffic-shaping which may not be readily apparent to the human user.

4.4.3 Using measurement-servers versus real-life servers

One design choice is to use measurement servers under the control of the tool to contact when performing the analysis. This has the advantage of being more accurate as we can measure the transmission characteristics at both ends of the connection and therefore attribute problems in throughput to the middle of the network. For this reason, it is a popular approach [16, 44, 43] when measuring network neutrality violation or traffic shaping. However, this may also not detect all the problems in real-life as the attacker may only be manipulating on certain real life website while not affecting connections to any other destination including those of the measurement servers.

Chapter 5

Analysis

The effectiveness of friendsourcing as an approach for detecting network manipulation can be measured both with analytical evaluation of data sets as well as field studies using real users. Analytical evaluation will help in proving whether friendsourcing as an idea is feasible. It will also help us in studying the effects of various design choices and optimizations on system performance.

In this chapter, we give coverage and redundancy analysis of multiple OSN data sets to measure the feasibility of friendsourcing to detect network manipulation. We also describe evaluation of a simulated system using this data set and describe optimizations that improve the performance numbers of the simulations. Section 5.1 describes the various data sets we use in our evaluation and their characteristics. Section 5.2 gives a definition of the various coverage metrics. It also gives the coverage metrics of the data sets we study. Section 5.3 gives a definition of the various redundancy metrics. It also gives the redundancy metrics of the data sets. Finally, Section 5.4 describes the optimizations we can use for friendsourcing and their effects on simulations of a distributed friendsourcing architecture.

5.1 Datasets

To study the effectiveness of friendsourcing in detecting network manipulation, we evaluated it using simulations on three real-life social networks. A critical aspect of our evaluations depends on the network coverage that users can expect from their social networks. We can only use datasets which consist of *both* social networks and the network location of those users. Unfortunately, there are few publicly available social network datasets that give them both. It is also difficult or impossible to infer these attributes (especially the ISP `I`) from crawling through publicly available attributes in a social network. To get an accurate eval-

uation of the coverage given by friendsourcing, we used three real-life datasets with different characteristics.

Arxiv: We used the dataset [27] consisting of a collaboration network of authors who have published on Arxiv [26] in the field of high-energy theoretical physics on or before 2003 consisting of 8392 users with 40774 connections (collaborating links) belonging to 2149 unique domains. This has the advantage of high fidelity, as the authors in all of the cases were associated with a particular educational institution. Each author on a paper is considered a friend of any of the other coauthors of the paper. Each author is associated with the network location of his university domain. In case of an author with multiple affiliations, we chose the affiliation associated with the latest published paper. A problem with this dataset, however, is that it is very sparse. The average user has less than 10 friends (in contrast to online social networks like Facebook where 100 is the norm) and many users have exactly one friend or are the only person from within their administrative domain. Another problem was that many of the authors (25.7%) were lone authors, i.e., they did not collaborate with any other author. For these reasons, we also analyzed the results with those users excluded.

Gowalla: Gowalla was a location-based social network where users 'checked in' at different geographic locations to be shared with their friends. We used data from the location-based social network taken from [28]. This consisted of 196,591 users with 950,327 edges (friendship links) between them. The dataset consists of 6,442,890 check-in locations given as pair of latitude and longitude coordinates, with multiple check-ins per user. We used an off line reverse geocoding lookup to infer the city and country attributes from the check-in locations. We used the most common city and country values as the home location of the user. This follows the methodology in the paper [28] where the users are assigned a home location as the average of the check-ins in the 25km by 25km grid containing most check-ins. In our case, as we are not concerned with the exact home address, we simply take the most frequently occurring city. We verified the accuracy of our off line reverse geocoding lookup by verifying a sample of the locations with an accurate online reverse geocoding web service [65].

BrightKite: We also used data from another location-based social network taken from [28]. This consisted of 58,228 users with 214,078 edges (friendship links) between them. It consisted of 4,491,143 check-in locations, with multiple check-in locations per user. Similar to the Gowalla dataset, we used reverse geocoding to get an home location for a user.

Note that, while Gowalla and BrightKite datasets give us good metrics to measure the coverage over a geographical area (including the country), we cannot directly infer the ISP (or organization attributes) of the user. To fill the ISP attribute, we randomly assigned all the users in a given city randomly into 3 or more ISPs. This assignment was based on the number of users in a given administrative location. We assigned a minimum of 3 ISPs for any given location with a user population of less than 1000. We assigned more ISPs (4,5,6) for locations with 2000, 4000, and more users. This choice of the number of ISPs per cities is an heuristic assignment that was determined based on a estimates of customer subscriber numbers for different ISPs [66].

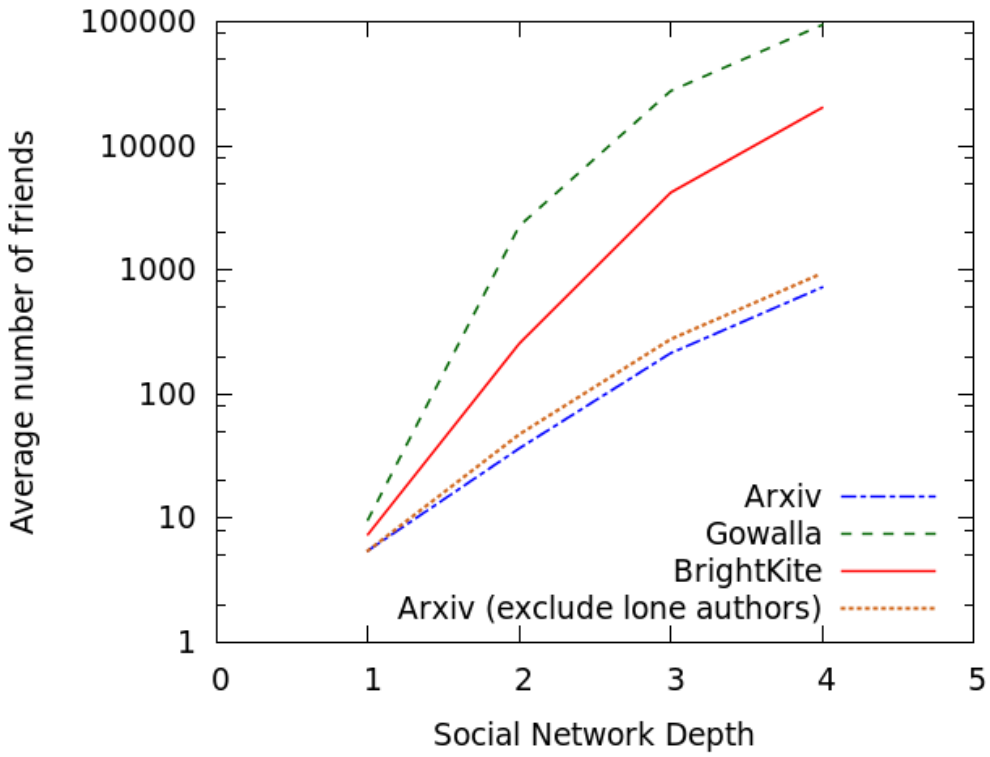


Figure 5.1: Average number of friends per user (at a social network depth) for Arxiv, Gowalla, and BrightKite

Figure 5.1 gives the average number of friends per user in both of these networks as function of social network distance. Note that the y-axis is log scale. We can clearly see that the average number of friends at depth 2 and more is much larger for Gowalla and BrightKite as compared to Arxiv. Furthermore, a good percentile (22%) of the Arxiv dataset consists of 'lone' authors who have not published with other authors. We explicitly account for this by giving the average

number of friends at a depth excluding those users as well. In terms of the average number of friends available at depth 2, both Gowalla and BrightKite networks are more representative of a general purpose OSN like Facebook than Arxiv. In fact, an analysis [22] on Facebook showed that the median number of friends for a user was 99, while a user with 100 friends had 27,500 friends-of-friends. Both of these are values for Facebook are higher than the values we see in either BrightKite or Gowalla.

5.2 Coverage Metrics

A user should be able to verify the possibility of network manipulation by any party on its network path. In this work, we concentrate on the four parties most likely to manipulate: Countries (`C`), ISPs (`I`), Organizations (`O`), and the user's own Machine `M`. While it is possible for other network entities to cause problems (such as badly configured wireless routers), the `C,I,O,M` entities would account for most of the current deliberate manipulation.

To study this, we calculate the coverage of these network attributes `C,I,O,M`, by a user's social network. A high coverage by a user's social network will result in more of a possibility of detecting the exact source of network manipulation. If network attributes were uniformly distributed amongst all users, we can see that social networks such as Gowalla and BrightKite by virtue of having a much larger number of friends at depths greater than 2 will have higher coverage than that of Arxiv. However, this may not necessarily be the case and the user's network attributes may be correlated with that of their friends resulting in a poor coverage.

We illustrate the coverage concepts with a running example of the configuration shown in Figure 5.2. It shows 6 users $U_1, .., U_6$, two organizations O_1, O_2 for users U_1, U_6 respectively, three ISPs I_1, I_2, I_3 and two countries C_1, C_2. Note that the communication links are logical. The communication links are in the direction from downstream manipulators to the upstream manipulators (i.e., the countries are upstream of the ISPs and the ISPs are upstream of the organizations). The social network links connect users with their immediate friends.

We define the terminology that we use to measure coverage here:

Definition: A *network attribute* A is a logical administrative entity that is found along a user's network path. We denote the value of a network attribute A for any given user U as *Attr(*A*,*U*)*

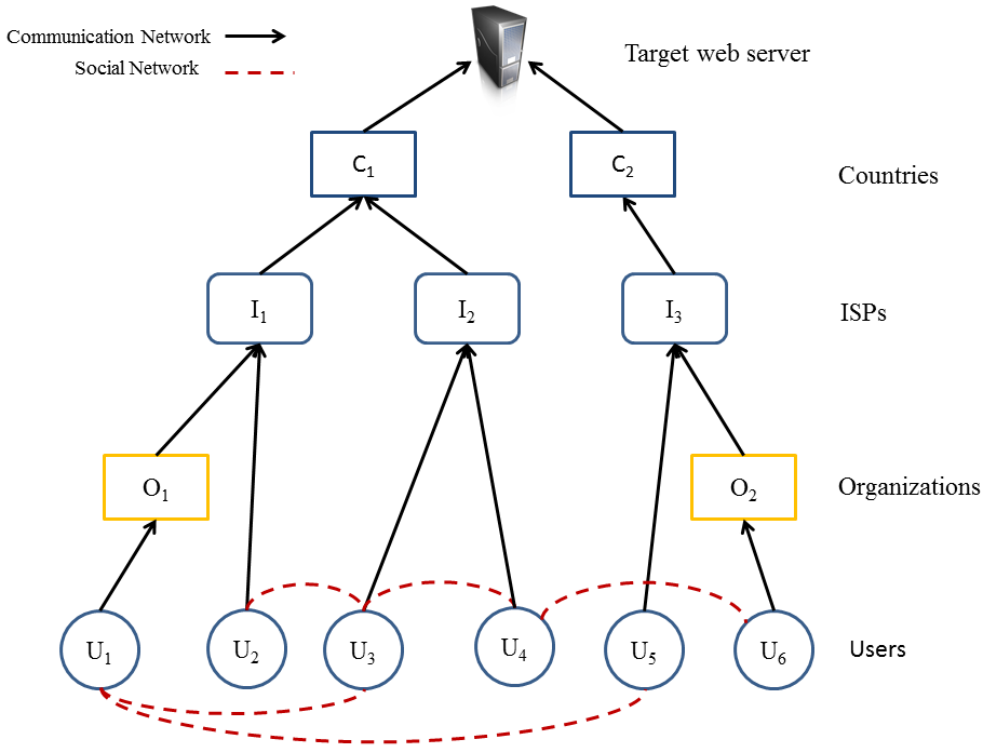


Figure 5.2: Example social and communication network

In our case, as mentioned earlier, we are only concerned with the network attributes `C,I,O,M`. Often, such as when the user is connected from their home, the organization attribute `O` is empty. In the example in Figure 5.2, we have $\text{Attr}(I, U_3) = I_2$ and $\text{Attr}(C, U_3) = C_1$.

Definition: We denote $A \rightarrow B$ to mean that the attribute B is *immediately upstream* of attribute A. The attribute hierarchy is given as $M \rightarrow O \rightarrow I \rightarrow C$, with the attribute for machine, `M`, at the bottom and the attribute for country, `C`, at the top. An attribute B is *upstream* of A, if $A \rightarrow B$ or there is an attribute C such that $C \rightarrow B$ and C is upstream of A.

In our example, by definition, ISPs are *immediately upstream* of a user's organization. ISPs are upstream of both organization and machine. Countries are upstream of all the other network attributes, `I,O,M`.

Definition: A user U *covers* a network attribute A, at depth d, if there is a friend f within d hops of u on the social network such that:
a) $\text{Attr}(A, U) \neq \text{Attr}(A, f)$, and

b) for all attributes A_{up} upstream of A, $\text{Attr}(A_{up}, U) = \text{Attr}(A_{up}, F)$
We denote the set of all attributes covered by U at depth d as *AttrCoveredAt(*U, d*)*.

In the example 5.2, user U_3 can 'cover' her machine within depth 1. This is because she has an immediate friend U_4, who shares all upstream network attributes (i.e., I_2, C_1), but can verify from an independent machine. This would enable her to attribute the blame on her machine if she cannot access the target web server from her machine but see that U_4 can. Similarly, U_3 can cover her ISP (I_2) within depth 1, because her immediate friend U_2 can send a probe through another ISP I_2 but with the same upstream country C_1. However, U_3 cannot cover her country within depth 1. She can cover the country at depth 2 through user U_5 via her immediate U_1. Therefore, $\text{AttrCoveredAt}(U_3, 1) = \{I, M\}$ and $\text{AttrCoveredAt}(U_3, 2) = \{C, I, M\}$.

Similarly, user U_3 is *country-covered* at depth 2 through the paths $U_3 \rightarrow U_4 \rightarrow U_6$ and $U_3 \rightarrow U_1 \rightarrow U_6$. U_3 is *ISP-covered* at depth 1 through user U_2.

Definition: *Attribute-coverage* of network attribute A at any given depth k, is the percentile of users in a dataset that are covered for A at depth k.

For instance, *ISP-coverage* at depth 2, gives the percentile of users in a dataset who are *ISP-covered* at depth 2. In our example, U_1, U_2, U_3 are ISP-covered at depth 1, i.e., they all have another immediate friend who can check the target web server through another ISP. Therefore, we have an ISP-coverage of $3/6$ or 50% at depth 1.

Definition: A user U is *fully covered* at depth d if:
For each network attribute A, U can cover A within depth d.

Consider the user U_3 in Figure 5.2. To cover the machine on which U_3 is working, i.e., to see if it was her machine that was causing the problem, there needs to be a probe that can be sent through all the upstream network attributes of U_3 (i.e., both I_2 and C_1) but from a different machine. U_3 can do this by contacting her immediate friend U_4 at 1-hop. To cover the ISP I_2, U_3 needs some one with a different ISP, but the same country. U_1 can do this by contacting her immediate friends U_1, U_2, both of whom use ISP I_1. However, none of her immediate friends can cover for her country C_1. To check if the target web server is indeed accessible from a different country, U_3 will have to traverse two hops on her social network.

This she can do by either contacting U_6 through U_4 or by contacting U_5 through U_1. Thus U_3 is *fully covered* at depth 2.

Note that, by this definition, some users will never be fully covered at any depth. Consider, U_1 in Figure 5.2. U_1 will never be able to cover her machine. This is because there is no other user at any depth on the social network who has all the same network attributes upstream of her machine. In particular, there is no other user who is of the same organization O_1. Therefore, U_1 will never be able to know, using the currently available friends on the social network, if it is her machine or her organization O_1 doing the manipulation.

Definition: *Complete coverage*,at a given depth d, is the percentile of users in a dataset that are *fully covered* at depth d.

In the example given in Figure 5.2, U_3 and U_4 are the only users who are *fully covered* at depth 2. Thus, the complete coverage at depth 2 for this example is 33%. U_2 will also be fully covered at depth 3, when she will be able to get a probe from another country C_2 (via the paths $U_3 \rightarrow U_4 \rightarrow U_6$ and $U_3 \rightarrow U_1 \rightarrow U_5$). Thus the complete coverage at depth 3 for this example dataset is 50%. Note that, due to both U_1 and U_6 not being fully covered at any depth, this dataset will not have a 100% coverage at any depth.

Definition: A user u has *best possible coverage depth* d if:
$\forall\, k > d$, $\text{AttrCoveredAt}(U, d) = \text{AttrCoveredAt}(U, k)$.

Coverage metrics for every user are monotonically nondecreasing over the depth parameter. The number of network attributes that can be covered for any user is finite. Thus for every user, we will have a depth d beyond which there is no further improvement in the number of network attributes that can be covered. The user may be *fully covered* at this depth d, but it is not a necessity.

Consider for instance the user U_1 in Figure 5.2. The user has a best possible coverage depth of 2 at which point she can cover her ISP and her country. However, she will never be able to cover organization O_1, as there is no other user who also has that organization in their upstream at *any* social network depth from the user U_1. Conversely, the best possible coverage depth for user U_5 is 4 as she can cover the ISP I_3 through 4 hops ($U_5 \rightarrow U_1 \rightarrow U_3 \rightarrow U_4 \rightarrow U_6$).

Definition: *Best possible coverage* at depth d, is the percentile of users in a dataset that have a best possible coverage depth of d or smaller.

For the example in Figure 5.2, U_1, U_3, U_4 have a best possible coverage depth of 2. Hence the best possible coverage at depth 2 is $3/6$, i.e., 50%. U_2 has a best possible coverage depth of 3, therefore the best possible coverage at depth 3 is $4/6$ or 66.6%.

5.2.1 Complete Coverage Results

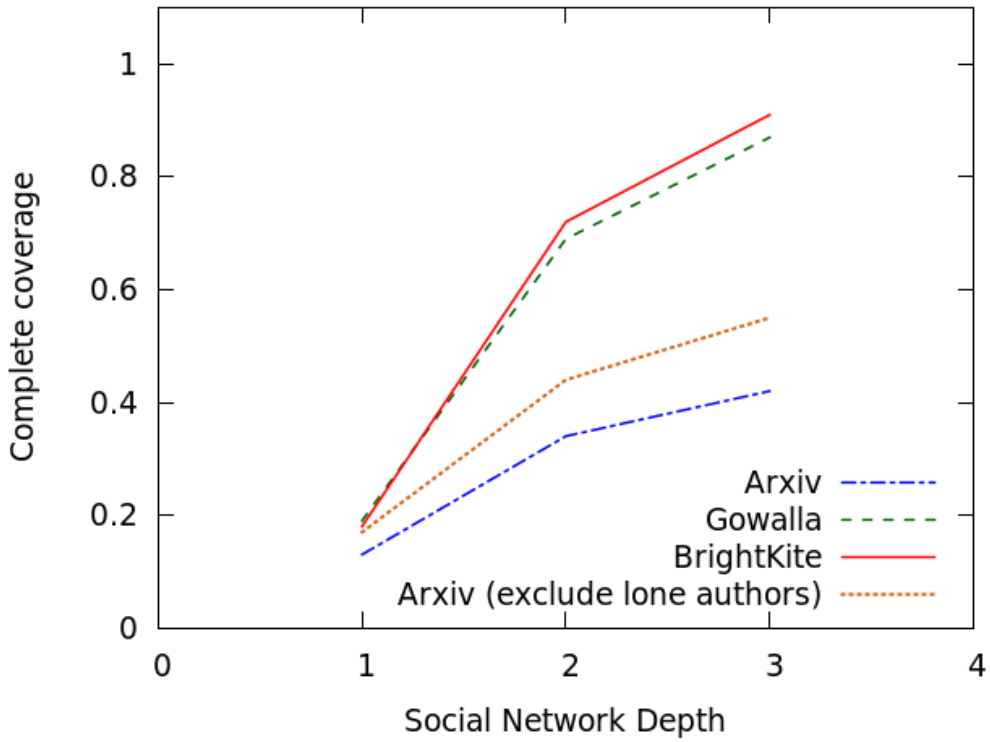


Figure 5.3: Complete coverage results

Figure 5.3 shows the complete coverage results for all three of our datasets. As we can see, the Arxiv network has relatively poor complete coverage results compared to the other two OSN. This was because a large number (25.7%) of users were lone authors who did not have any collaborators. This means that these users will not be able to acquire help from other friends to help perform their coverage analysis. We see that the complete coverage results improve slightly for Arxiv, once we remove these lone authors.

On both the Gowalla and BrightKite networks we get complete coverage for a high proportion (70%) within two hops. This is a direct consequence of the

much larger number of friends available at depth 2 on these networks compared to Arxiv as seen in Figure 5.1. A larger pool of friends in both these cases, therefore, implies a greater chance of finding someone with different network attributes. This is an encouraging result, as general purpose OSNs like Facebook are more similar to Gowalla and BrightKite than Arxiv.

5.2.2 Best-possible Coverage Results

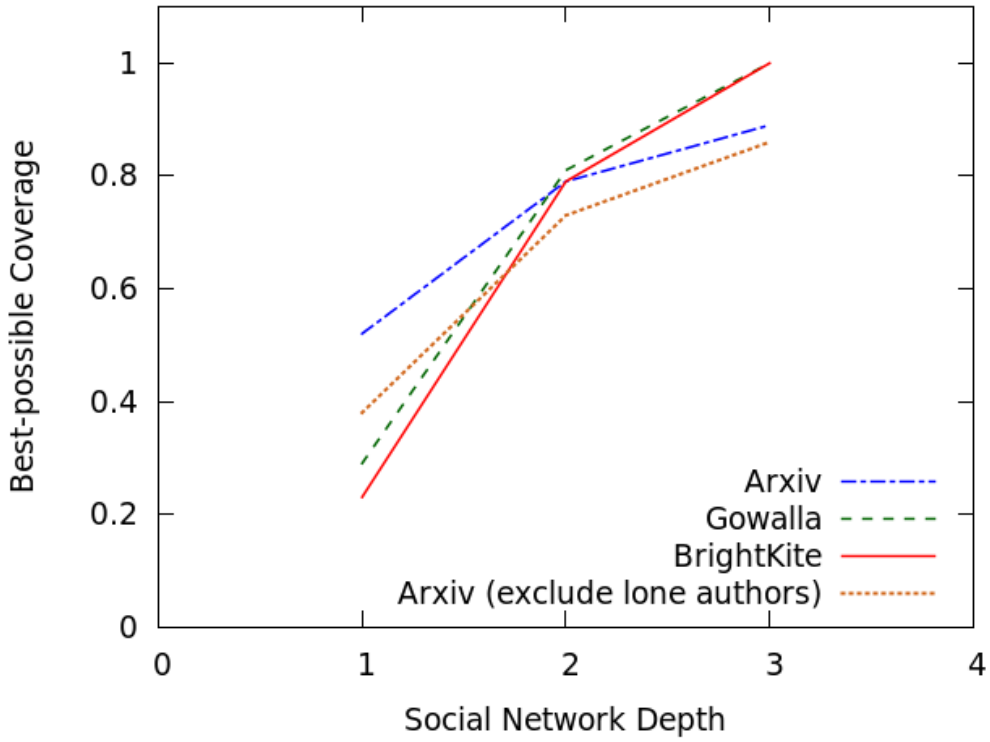


Figure 5.4: Best-possible coverage results

Figure 5.4 shows the best possible coverage results for all three of our datasets. Note that the best possible coverage results for all the datasets (including Arxiv) are high (80%) within two hops. In particular, this is a huge improvement for Arxiv dataset.

This suggests that the reason we had poor complete coverage was because of the sparsity of the Arxiv dataset, i.e., lack of users who had the relevant network attributes, and not due to lack of coverage of the social network itself. That is, if a user was unable to cover for their ISP or country, it was because there were

no other users who were within that country or ISP, and not because they were available but not within the reach of friendsourcing. Note also that almost all of the users in the Gowalla and BrightKite get their best possible coverage results within 3 hops.

5.2.3 ISP Coverage Results

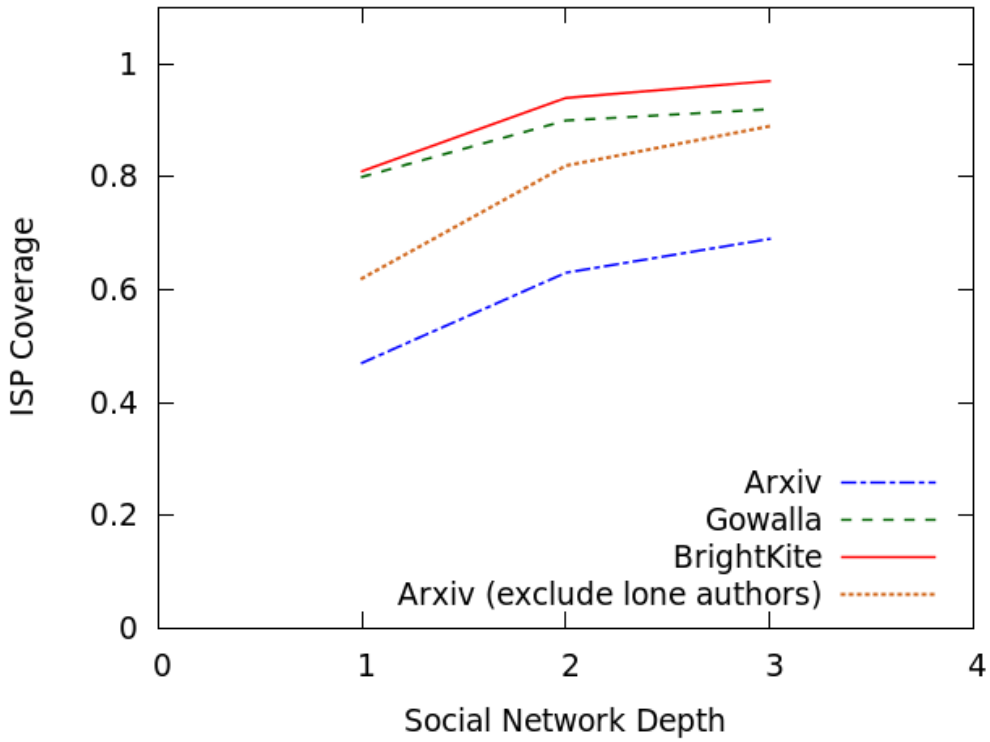


Figure 5.5: ISP coverage results

Figure 5.5 shows the ISP coverage results for all three of our datasets (as well as the Arxiv data set excluding the lone authors). Firstly, we see that there is high coverage for ISP coverage even at depth 1. This indicates that a large percentile of the population (around 80% for Gowalla and BrightKite and around 60% for Arxiv(excluding lone authors)) have an immediate friend who can verify if a target web server is accessible from another ISP. ISP-coverage which only looks at finding a friend with a different ISP is a more suitable metric to look at when we are worried exclusively about network neutrality violations by ISPs. The high ISP-coverage numbers, even at depth 1, indicate that friendsouring is a good strat-

egy for the large majority of users when they want to verify network neutrality violations by ISPs.

The smaller ISP-coverage numbers for Arxiv (at depth 1) are indicative of the fact that there are many authors who only published with other authors from the same university in the dataset (i.e., all of their friends belong to the same ISP) or, alternately, they have only published with authors in other countries. These authors will therefore not be able to get ISP coverage at depth 1. However, if one of their friends has published with other authors in a different ISP (but the same country), they will be able get ISP-coverage at depth 2, which is what we notice.

5.2.4 Country Coverage Results

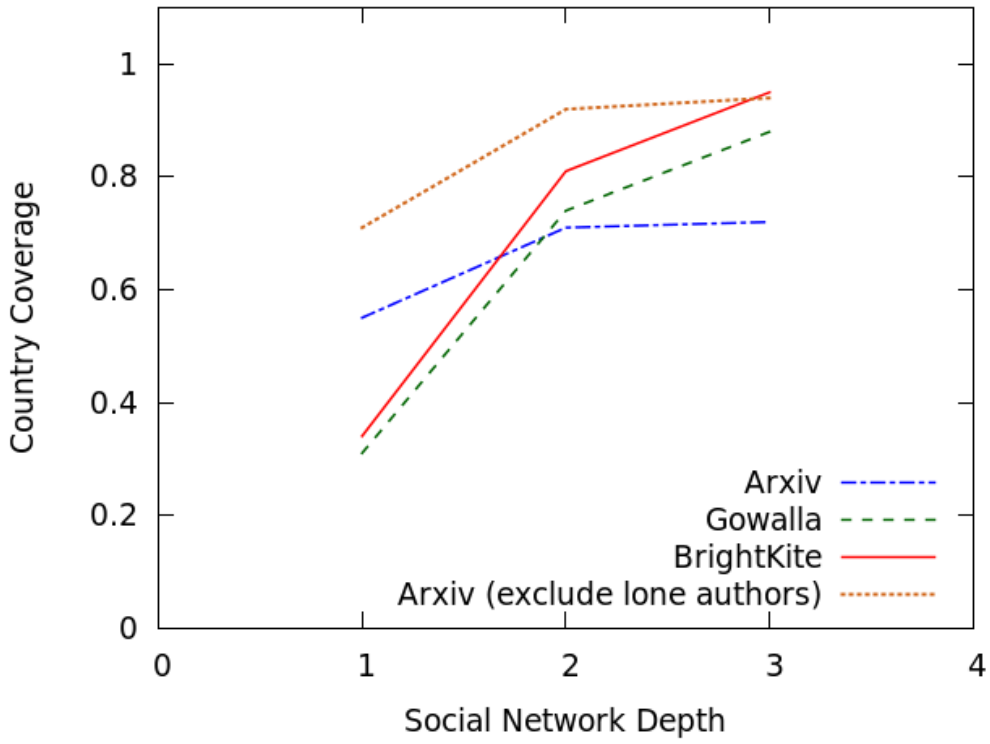


Figure 5.6: Country coverage results

Figure 5.6 shows the country coverage results for all three of our datasets. Note that unlike the ISP coverage numbers at depth 1, the country coverage numbers for Gowalla and BrightKite are smaller, at around 30%, compared to the Arxiv dataset which is very high at around 70%. For Arxiv, this is an indication that

70% of the users have co-authored with people who are from different countries. This is not surprising in case of academic users.

BrightKite and Gowalla, consisting of more general purpose users will not have same coverage. Even so, the country coverage at depth 1 shows that 30% of these users have a immediate friend in another country. The country coverage numbers at depth 2 are much higher (at around 75% and 80%) for Gowalla and BrightKite. This is a consequence of the relatively denser connectivity in both these networks. A denser connectivity in the datasets will lead to a user finding one of these 30% of users as a friend of friend. These numbers are very encouraging for friendsouring, especially because general purpose OSN like Facebook are denser than the BrightKite and Gowalla datasets we studied. It also gives us depth 2 as a good baseline for friendsourcing recruitment depth when trying to uncover country based censorship.

5.3 Redundancy Metrics

For friendsourcing to be successful in real life, it is not enough to prove that we have high coverage. For instance, it is possible that a user has high coverage of the network with the help of their friends, but may not be successful as some of their friends are not cooperative (or available). To measure the resiliency of a social network to perform network detection, we study a metric related to coverage, which we call *redundancy*.

Definition: A user U has n-*redundant cover* over a network attribute A, at depth d, if a user has n distinct friends $f_1, .., f_n$ within d hops of U such that:
a) $\forall_{1 \leq i \leq n} \text{Attr}(A, U) \neq \text{Attr}(A, f_i)$, and
b) for all attributes A_{up} upstream of A, $\forall_{1 \leq i \leq n} \text{Attr}(A_{up}, U) = \text{Attr}(A_{up}, f_i)$
We denote the set of all attributes n-redundant covered by U at depth d as *RedundantCoveredAt(*U, d, n*)*.

For example, in figure 5.2, user U_2 has a 2-redundant cover over the ISP at depth 2. This is because she has two distinct users, U_3 at depth 1 and U_4 at depth 2 (via U_3) that have a different ISP attribute value (I_2), but the same attribute values upstream of ISP (i.e., they belong to the same country). In this case, U_2 has two friends who can help her verify that it is the her ISP (and not her country) that are responsible for any network manipulation done by ISP I_1.

We can define many of the coverage metrics as *n-redundant*, analogous to the $\mathtt{n}$-redundant cover definition above. Wive some of these definitions here.

Definition: A user $\mathtt{U}$ is $\mathtt{n}$-*redundant fully covered* at depth $\mathtt{d}$ if:
For each network attribute $\mathtt{A}$, $\mathtt{U}$ can $\mathtt{n}$-redundant cover $\mathtt{A}$ within depth $\mathtt{d}$.

Definition: $\mathtt{n}$-*redundant Attribute-coverage* of network attribute $\mathtt{A}$ at any given depth $\mathtt{k}$ and redundancy $\mathtt{n}$, is the percentile of users in a dataset that are $\mathtt{n}$-redundant covered for $\mathtt{A}$ at depth $\mathtt{k}$.

Definition: $\mathtt{n}$-*redundant complete coverage*, at depth $\mathtt{d}$ and redundancy $\mathtt{n}$, is the percentile of users in a dataset that are $\mathtt{n}$-redundant fully covered at depth $\mathtt{d}$.

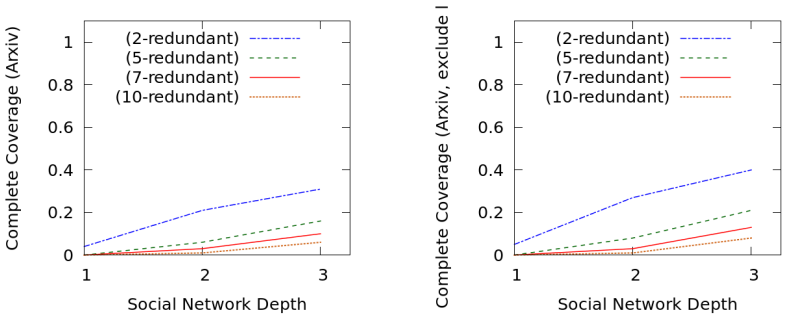
Definition: A user $\mathtt{U}$ has $\mathtt{n}$-*redundant best possible coverage depth* $\mathtt{d}$ for a given redundancy $\mathtt{n}$ if:
$\forall\ \mathtt{k} > \mathtt{d}$, RedundantCoveredAt$(\mathtt{U}, \mathtt{d}, \mathtt{n})$ = RedundantCoveredAt$(\mathtt{U}, \mathtt{k}, \mathtt{n})$.

Definition: *Best possible* $\mathtt{n}$-*redundant coverage* at depth $\mathtt{d}$ and redundancy $\mathtt{n}$, is the percentile of users in a dataset that have a $\mathtt{n}$-redundant best possible coverage depth of $\mathtt{d}$ or smaller.

For instance, a *10-redundant fully covered user at depth 2* user, has 10 distinct users for every network attribute that can disambiguate all their network attributes.That is, for each one of $\mathtt{O}, \mathtt{I}, \mathtt{C}$, the user has 10 other friends within 2 hops, who can help them disambiguate those attributes. Similarly, a 50% 5-redundant ISP coverage, at depth 1, tells us, that 50% of users have five or more friends who belong to a different ISP (but are within the same country of the user). A higher percentile of users with highly-redundant coverage, implies that users can get meaningful results even when most of their friends are not available or are uncooperative.

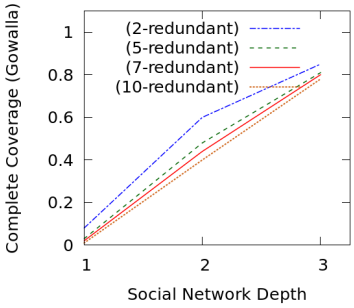
5.3.1 Redundant Complete Coverage

Figure 5.7 shows redundant complete coverage metrics for various values of redundancy. Recall that complete coverage requires us to cover for every single network attribute that the user has (including the user's own machine). As seen earlier in figure 5.3, the Arxiv dataset has smaller complete coverage when compared to both Gowalla and BrightKite datasets. We see that for high-redundancy

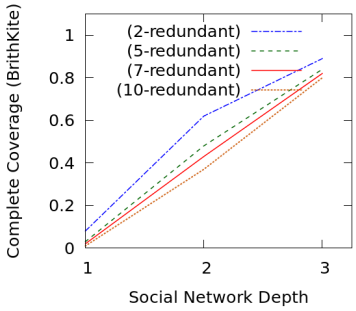


(a) Arxiv

(b) Arxiv (excluding lone authors)



(c) Gowalla



(d) BrightKite

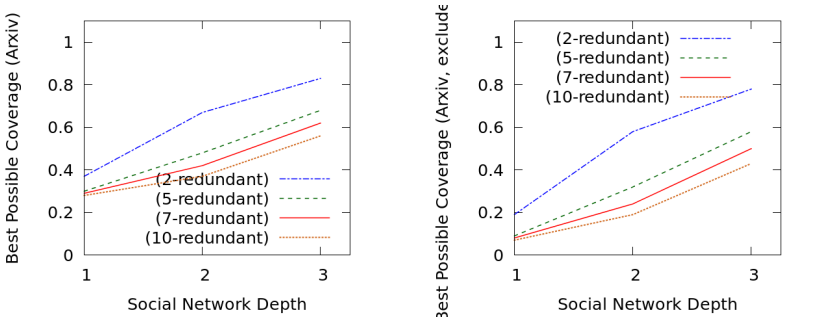
Figure 5.7: Redundant Complete Coverage Metrics

values in Arxiv 5.7(a), we get much poorer results in terms of complete coverage. In contrast, both BrightKite 5.7(d) and Gowalla 5.7(c) are relatively robust even at higher redundancy numbers.

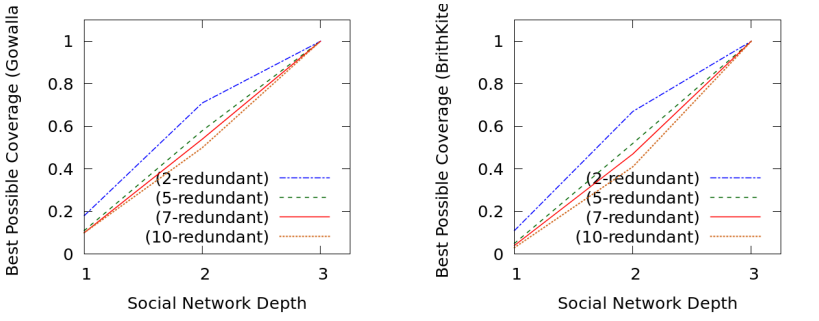
This can be explained by the sparseness of the Arxiv dataset compared to the other datasets. As we have seen in figure 5.1, there is an order of magnitude difference in the number of friends at depth 2 and higher in BrightKite and Gowalla as compared to Arxiv. This leads a much larger pool of friends for every network attribute in the BrightKite and Gowalla datasets compared to the Arxiv dataset, and consequently leads to higher redundancy numbers.

5.3.2 Redundant Best Possible Coverage

Figure 5.8 shows the best possible coverage metrics for various values of redundancy. Recall that best-possible coverage metrics are less restricted compared to complete coverage metrics. We consider a user to have a best possible coverage at depth d, if there is no other user at depth greater than d who can improve the coverage for the user. Once again we see that Arxiv(in figure 5.8(a) and figure 5.8(b))



(a) Arxiv (b) Arxiv (excluding lone authors)



(c) Gowalla (d) BrightKite

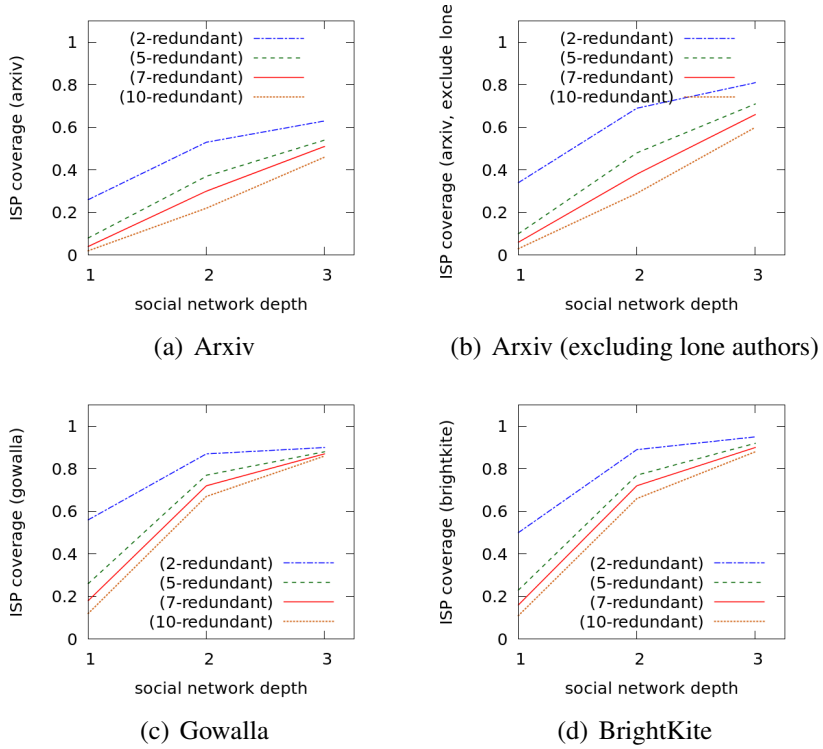
Figure 5.8: Redundant Best Possible Coverage Metrics

degrades much faster for higher values of redundancy compared to both Gowalla and BrightKite.

More interestingly, we see that at depth 3 both Gowalla and BrightKite have best-possible coverage for 100% of the users even with a high 10-redundancy. This means that in both these datasets, the user has very high resiliency in terms of coverage when we are willing to recruit up to depth 3. It also means that there is very little benefit to to beyond depth 3 for any network manipulation experiments.

5.3.3 Redundant ISP Coverage

Figure 5.9 shows the ISP coverage in the various datasets for various redundancy values. Once again, we see that for the Arxiv dataset, we get less redundancy for ISPs (or less ISP coverage numbers for high redundancy values). Both BrightKite and Gowalla have higher redundant coverage of ISPs. In particular, both of them have more than 65% of their users with ISP coverage at 10 redundancy at depth 2. This means that more than 65% of their users will have 10 users within friends



(a) Arxiv (b) Arxiv (excluding lone authors)

(c) Gowalla (d) BrightKite

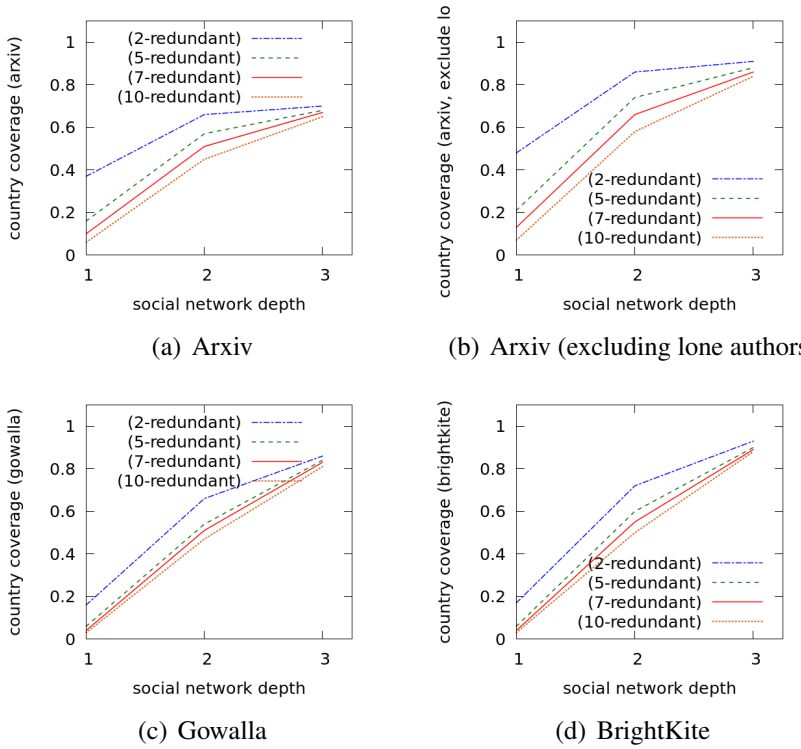
Figure 5.9: Redundant ISP Coverage Metrics

of friends who will be able to help with any network neutrality violation measurements targeting their ISP.

Furthermore, if we are willing to recruit friends at depth 3, we get a 10-redundant coverage for more than 85% of the users. In case of Arxiv, coverage at high redundancy numbers is much lower. This is easier to explain when we refer to the number of friends per user at any given depth. To get a 10 redundant ISP coverage at depth 2, a user should have at least 10 friends or friends of friends who belong to the same country but are of a different ISP from the user. Since the average number of friends in Arxiv at depth 2 is only 36.3 (see Figure 5.1), we have fewer number of users who can satisfy that criteria.

5.3.4 Redundant Country Coverage

Figure 5.10 shows the country coverage metrics for various redundancy values. This shows that the Arxiv datasets are much more suitable for getting country coverage. In particular we see in, figures 5.10(a) and 5.10(b), that at depth 1



(a) Arxiv (b) Arxiv (excluding lone authors)

(c) Gowalla (d) BrightKite

Figure 5.10: Redundant Country Coverage Metrics

Arxiv (with and without lone authors) Arxiv datasets have better country coverage when compared to Gowalla and BrightKite.

Gowalla and BrightKite do have good redundant country coverage at depth 2. In particular, around half of the users in Gowalla and BrightKite has at least 10 friends or friends of friends who belong to a different country. If we consider friends at depth 3, all the datasets have a very high ($> 80\%$) 10-redundant coverage of the country. This means that if we do recruit at depth 3, the vast majority of users will be able to have at least 10 people in their friend circle to send probes from a different country.

5.4 Optimization of Recruitment

A simple way of performing a recruitment in a distributed friendsourcing architecture is to simply ask all friends (and ask them in turn to forward it to their friends). This flooding of requests can be bounded by giving a limit on the hop count, which is reduced every time a friend forwards a request to their other friends. As discussed in 4, a simple flooding of requests to friends is not always optimal. We are

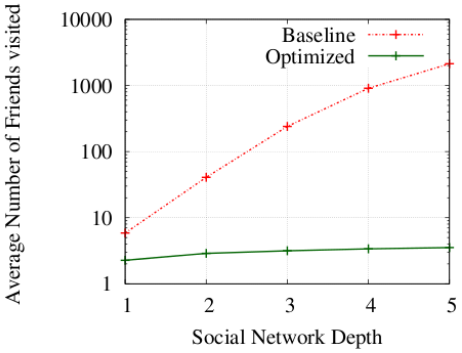
likely to get redundant results from friends who are located in the same underlying physical network. Furthermore, there is also an overhead of messages that are sent to various friends and probes being performed against the target web servers.

We analyzed simple optimization protocol which increased the efficiency of recruitment considerably. In this optimization, we pre-share network attributes amongst immediate friends to enable us to choose the friends to be contacted for any new probe request. This approach may be viewed similar to a how routing table updates are managed. Periodically, users exchange the list of their upstream network attributes that belong to them (i.e., their organizations, ISP, country). A user will keep a list of network attributes that are reachable from each of its neighbors. After the first exchange of these network attributes, each user will know, for every friend which of the network attributes can be covered using that immediate friend. For instance, a friend with the same country but different ISP can be used to cover a user's ISP. In the second round, the user share the collected information (its own attributes at depth 1 and the attributes of its neighbors at depth 2) with its neighbors.

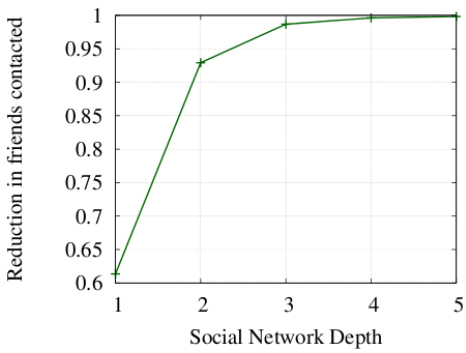
After a few iterations, this information will give for every friend, the list of network attributes it can cover for a given depth. So we might have a friend who can only clarify whether a server is reachable or not at depth 1, but at depth 2 (that is using her friends), she might be able to distinguish all the possible upstream network attributes along our path. This serves two purposes. In case the list of attributes with a given friend is mutually exclusive, the friend cannot be a good sensor for that particular request (other than to verify if the server is accessible at all). This will reduce the number probe requests to a subset of the friend list. If the list of upstream domains disambiguated is identical for one or more of the friends, we only need to send the probe request to one of them. This will reduce the total number of messages in the system.

We evaluated the result of these optimizations on a simulated distributed system with users from the Arxiv dataset. We show the results of our optimizations on both the load on the target web server as well as the overhead on the system itself (in terms of users visited etc.,) below:

Load on the Target Web server: Figure 5.11(a) shows the average number of friends visited per single inference request. Since each friend visited also sends a single probe, it is also a measure of the load on the target server due to a single manipulation detection request by a user. Note that the y-axis is in log scale. As we can see, the unoptimized version is very inefficient in terms of the number

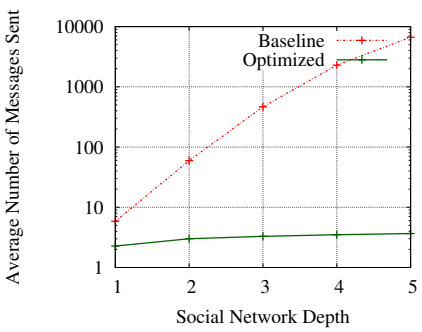


(a) Average number of friends visited (and probes sent) per single inference request as measured against depth of recruitment

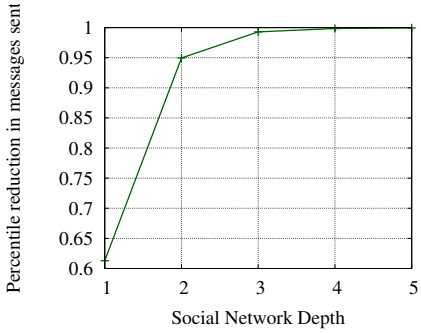


(b) Percentile reduction in friends visited per user request

Figure 5.11: Friends visited per original user request on Arxiv dataset



(a) Total messages sent on the network on average per single probe request as measured against depth of recruitment



(b) Percentile reduction in total messages sent on the network per single inference request between baseline and optimized versions

Figure 5.12: Messages sent during friendsourcing on Arxiv dataset

of messages taken. This also explains the large number of messages exchanged between users for every single inference request that we noted in Figure 5.12(a). We note that, on average, each user on our social network has 5 friends. However, within just 3 hops, the average user's friend circle grows up to 240 and reaches 2,130 for depth 5. Figure 5.11(b) shows the percentile reduction in the number of friends visited and, consequently, probes sent to the target by those friends as a result of our optimizations. As shown, we see a huge drop (more than 90%) in the number of probes sent starting at two hops. Note that this effect would be even more pronounced in densely connected networks (such as Gowalla).

Number of Messages: Figure 5.12(a) shows the average number of messages sent per one original network neutrality request in the system. In this we count

the number of messages sent between friends on the social network that were originated due to a single original probe request sent by a randomly chosen user. Note that the y-axis is on a log-scale. As we can see, the baseline version is very inefficient when compared against the optimized version. On average, a single inference request with a hop-count of 3 resulted in close to 470 messages exchanged between the users. This was because we simply sent the request to all possible friends on the social network within a hop-count of 3. Furthermore, the results of the probes need to be collated along the social network path, merged and returned back until we get the final result at the originating user.

Figure 5.12(b) shows the percentile reduction in the total number of messages per request of the optimized design when comparing it to the baseline design. We see tremendous reduction in the total number of messages sent per single verification request especially after depth 2. This is due to the reduction in number of friends being contacted at larger depths. The optimized version, on the other hand, only sends a request to a handful of users on the social network graph by looking at the shared upstream information corresponding to friends. A users is chosen only if it can 'cover' a new node along the originating user's upstream path.

Chapter 6

Field Study

While simulations and analysis can give us an estimate of the feasibility of friendsourcing as a strategy to detect network manipulation, a realistic evaluation of our strategy requires testing it in the presence of real-life users, testing real-life network manipulations. To this end, we implemented SiteViews, a web-browser based implementation to detect manipulation of web-based traffic. We performed a study with a total of 54 real unpaid users over a period of two weeks in India. All of these users were acquired through us contacting two individuals at the beginning of the study. Our study found a number of instances of manipulation in India that were both ISP-specific as well as country-wide. We describe the results of this evaluation in this chapter.

Section 6.1 gives an overview of the assumptions and goals that affected our design choices. It also gives an overview of the limitations of our implementation. Section 6.2 describes the architecture and implementation details of SiteViews. Section 6.3 shows the user interface of SiteViews and describes some of its functionality. Section 6.4 gives details of the study group that was acquired during the course of the study. Our study found evidence of a large amount of network manipulation of websites in India. Section 6.5 gives the analysis of the network manipulations: including the types of manipulation, the sources of manipulation, and the possible motivations.

6.1 Features, Assumptions, and Limitations

As described in Section 4.2, there are many desirable features from any design that uses friendsourcing. We give a discussion those features that we could incorporate and the effects of implementing those features on our design choices.

6.1.1 Ease of use

This was our biggest consideration when designing SiteViews. We wanted the tool to be usable without requiring installation or detailed knowledge of networking from the users. This also meant that our user interface was designed to be automated with minimal interactions from the user. To achieve this goal, we implemented SiteViews as a publicly accessible website that can be accessed on any platform (without having to be developed and tailored for each individual desktop or mobile OS).

The actual task of sending the probes and aggregation of the results of the probes can be done manually or in an automated fashion. To reduce the burden of overhead on users (and to decrease the possibility of misdiagnosis by users), we chose to do automated probing and aggregation. Because of our choice to go with an automated architecture, we had to use a mechanism for active probing at the client end on web-based platforms. In our implementation, this was performed automatically with the help of a signed Java Applet. In this aspect, our design choice is very similar to many other popular tools such as Glasnost [16], Netalyzer [44]. The use of a web-based app meant that users were able to access the tool without having to install it on their own machines.

Limitations: This centralized web-site based design with Java Applets, does have its drawbacks. The chief drawback of which is the fact that, the website itself may be blocked by any attacker to prevent such an analysis. We do note however, that this architecture can be implemented as an app on a SNS. In such cases, it might be difficult for the attacker to block access to just the app without blocking access to other features on a SNS.

6.1.2 Generality and Extensibility

Our design is able to detect web-based manipulation by countries, ISPs, and optionally organizations. This is done by correlating detected network failures from multiple locations with their network attributes. Our tool infers the country and ISP of a user during their registration using publicly available geolocation and whois data [67, 68]. The user is asked to enter an organization (if any) and can also correct the inferred country and ISP attributes. These attributes are also verified every time the user starts a new session from a different location. This archi-

tecture allows us to extend the functionality by implementing additional features in our applet.

Limitations: Due to the limitations of the applet based platform as described earlier in Section 6.1.1, our tool will not be able to detect all the kinds of network manipulation that a native application may be able to measure. Applets do not have access to precise network level information from the underlying system, and can only get access to the status of the network communication at the level of abstraction provided by the Java platform. This is not a big problem in the domain of web-based manipulation as we were able to access the HTTP status information. However it will not be feasible, for instance, to implement something like [43] which detects insertion of RST packets by ISPs. That approach relies on detecting the fact that a packet train arrives from the destination later than the fake RST packet inserted by the ISP. In our case, we will simply report it as a loss of connection and expect to find that the ISP is a possible cause, through inference from a group of measurements.

6.1.3 Effective Recruitment and Efficiency

We use a centralized coordinator based approach to improve efficiency and effectiveness of recruitment. The network attributes of each user are collected at the time of registration of the user (and verified at the beginning of a new session of a user). The coordinator will look at the network attributes of all of the available friends of a given user to choose the best possible subset of users to recruit. This also has the advantage of being more efficient than a purely distributed design. Firstly, there will be a reduction in the number of messages being sent to the users. Also, the system can cache probing results for use by other users of the system. This has the advantage of reducing the load on a target servers as well.

Limitations: A centralized coordinator approach has the limitation of being easily blocked, as described in Section 6.1.1. Another limitation of our implementation, but not the architecture, is that we currently do not use the history of a user when choosing users to recruit. This is because of the fact that in our architecture the actual probing is done automatically by the applet.

6.1.4 Graceful Degradation

Our system supports graceful degradation in case of limited capabilities of a user. Firstly, we use web-proxies as friends for users who have limited number of friends. This allows them to perform some analysis, even if it is limited by the number of web-proxies. Finally, Java Applets are not supported on most mobile platforms, which meant that we would not able perform automated probing or aggregate data from those platforms. In such cases, we chose to present the results from analysis of other friends without actually getting the data from the user itself. This meant that the user would, for instance, be able to detect if their country was the cause of network manipulation, but not if their mobile service provider was causing the network manipulation.

6.2 Architecture and Implementation

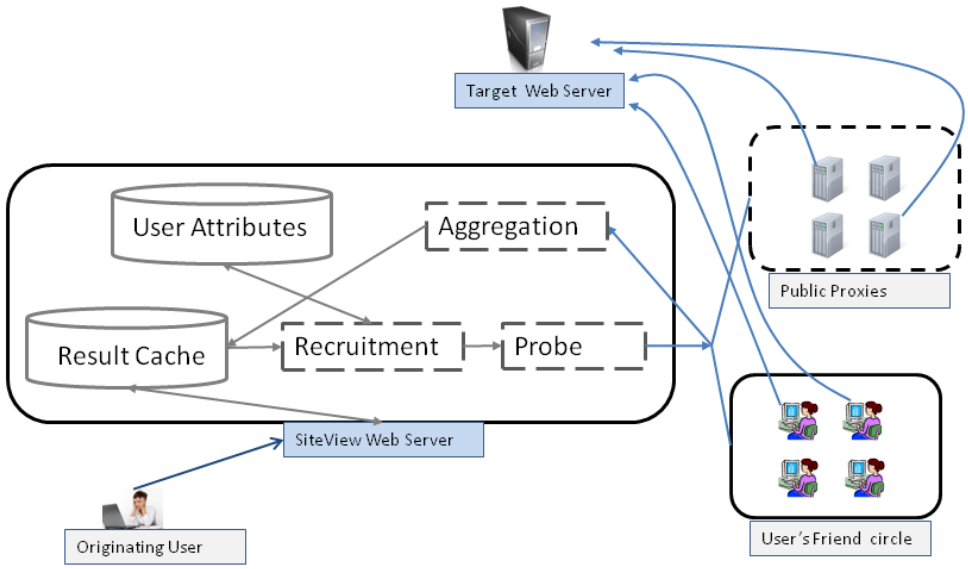


Figure 6.1: Architecture overview

SiteViews is accessible to users as a website with signed Java applets taking place of probing modules. In terms of the design choices described in Section 4.4, we use a *coordinated, on-demand* tool which measures network manipulation of *real-life* web servers. We use nginx 1.2.5 [69] to host the server. Figure 6.1 gives

an overview of the major components in our prototype implementation. While a general purpose application is more capable of taking measurements, we believe that a website based tool is more suitable for our design choices (i.e., *generality*, *ease of use*). A general, non-technical user is likely to be more used to visiting a web page than downloading a tool and running it off line. This also helps *generality*, as we do not need to develop executables to specific OS and mobile platforms. Furthermore, in case the user cannot run the Java Applets (due to permission settings or lack of availability on mobile platforms) we have a *graceful degradation*, in the sense that a user can get information about the accessibility of the website from other vantage points without being able to contribute any information from its own machine.

We give a short overview of the major components in the implementation of SiteViews.

Result Cache: The result cache consists of analysis done for previous requests from users. It also contains the contents of the web page along with the time, ISP, and Country from which the web page was accessed. In case a user requests a page available in the cache (and it is also not stale), we provide the results from the cache. Note that caching of results for failures is only done after a minimum number users have made identical observations of failure. For ease of implementation, this threshold was set in our case to 3. This criteria can be certainly be improved upon to include, for instance, the past history of user in reporting accurately.

User Attributes: The user attributes table stores the following details about the user: list of friends, home ISP, home country. In case the user is seen to be checking in from a different location, we treat them as a new user with the same list of friends. These attributes are populated when the user is first registered to SiteViews. In particular, users can invite other friends, who are added to the user's friend circle when they register. The attributes, including the list of friends, can be modified by the user in subsequent visits. Users have an incentive to increase their friend circle, as it helps them perform detection sooner.

Public Proxies: We maintained a small list of public http proxies, to perform requests to websites on behalf of the user. This was done so that we include results from a country outside India (in our case the US), as our study was limited to India. We also had 3 proxies in India so that a new user can get some meaningful results at the beginning. This choice of public web servers was made to ensure *graceful degradation* in case of new users who do not yet have any cooperating

friends. We also realized that, at least for the initial field testing of the prototype implementation, there will not be enough participants in the system. Once there are larger number of participants, we can reduce the reliance on such third party resources.

Recruitment: The recruitment module is used when the result of a user request is not available from within our Result Cache. This is used to select a subset of friends, public proxies which would be the most optimal to request probing. This was done as a result of our analysis in Section 5.4. A straightforward, ask all friends and friends of friends, approach will result in too many requests being sent to the target web server. These selections are based on their network locations as well as availability. The intuition behind the algorithm is to find for every network element (country, ISP, and optionally organization) of a given user, another active user within the friend circle who does not share one of those attributes. The recruitment algorithm is listed in Section 6.2.1.

Probe: The probe module is used to forward the requests to the targets selected by the Recruitment module. It performs two different operations in SiteViews. In case the target is a public web proxy, it fetches the target website using the proxy. In case the target is friend (or friend of friend), the request is put in a queue maintained by the SiteViews server. This queue is polled by applets running on the users of SiteViews at frequent intervals. The signed applets will fetch the website and the results are sent back to the aggregation module.

Aggregation: The aggregation module performs an analysis of the results from all the different sources for a given request. In case of failure to fetch a web page, we perform a simple correlation to determine the probable cause of the failure. These results (along with the successful, if any, web page content) are stored in the Result Cache and sent to the originating user. In case of website content, the aggregation module stores a single copy of identical web pages in the cache.

6.2.1 Optimizations

The most natural friendsourcing approach would be to request all the friends to fetch a given target, collate the results, and send it back to the originating user. While this would be the easier to implement, our analysis showed that this lead to many inefficiencies (see Section 5.4). A big problem with the flooding based approach, had to do with over-recruitment of users. In general, many of the probes

done by friends were redundant as they gathered information about the same network attributes multiple times. For instance, an analysis by Facebook [70], showed that the median number of friends for a user was 99, while a user with 100 friends had 27,500 friends-of-friends. To improve the efficiency of our architecture, we had to improve the propagation of the probe requests through the social network. We describe the list of optimizations that we perform below.

Shared Upstream Information: Upon registering to SiteViews we collect the upstream information for every user (i.e., organization `O`, ISP `I`, and country `C`). The ISP and country information can be inferred from the user's IP address. The user is prompted to also give, optionally, the name of the organization. Periodically, we update for each user the list of network attributes (i.e., `O,I,C`) reachable from itself and its neighbors. We also update this information to include the list of network attributes available at depth 2, i.e., friends-of-friends. Note that, in our implementation, the publicly available proxies are added as friends of all users.

This information gives, for every friend, the list of upstream domains it can disambiguate for a given depth. So, we might have a friend who can only clarify whether a server is reachable or not at depth 1, but at depth 2 (that is using her friends), she might be able to distinguish all the possible upstream domains along our path. We give the pseudo code for generating this information here.

```
struct upstream {
    set<user> country;  //friends outside country
    set<user> ISP;      //friends outside ISP
    set<user> Org;      //friends outside org
    set<user> colleague;
};
struct user {
    str country;// Country of the user
    str ISP;    // ISP of the user
    str Org;    // Organization (possibly empty)
    set<user> friends;// set of immediate friends
    upstream  up;
};
populate_upstream(user a)
{
    // populate friends,friends of friends
    f_of_f = set();
    foreach x in a.friends {
        f_of_f.add(x);
        f_of_f.add(x.friends);
```

```
    }
    f_of_f.erase(a);

    foreach x in f_of_f {
        if a.country != x.country
            a.up.country.add(x);
        else if a.ISP != x.ISP
            a.up.ISP.add(x);
        else if a.Org != "" && a.Org != x.Org
            a.up.Org.add(x);
        else
            a.up.colleague.add(x)
    }
}
```

Optimal Selection of Friends: Once we have the shared upstream information of users, we can utilize it to optimally select a subsection of users to recruit. In case the list of upstream domains with a given friend is mutually exclusive, the friend cannot be a good sensor for that particular request (other than to verify if the server is accessible at all). This will reduce the number probe requests to a subset of the friend list. If the list of upstream domains disambiguated is identical for one or more of the friends, we only need to send the probe request to one of them. This also reduces the total number of messages in the system. We give the algorithm for optimal selection of friends in Directed Probing.

Directed Probing: Frequently, users have a suspicion of the possible suspects and need only to clarify their suspicion. In our implementation, we allow users to give parameters specifying which of the administrative domains they want to verify. The recruitment module selects the necessary friends based on the user arguments. For instance, the user may only wish verify if the website is accessible from outside the country based on the assumption that a country level firewall is blocking access to content. In such cases, the algorithm will be called with only the `C` attribute set. By default all the `C,I,O` attributes for a given user are set while recruiting friends for that user.

```
recruitment(user a, str C, str I, str O)
{
    set<user> rec,x;
    if country != "" {
        //and --> set intersect
        x = a.up.country and live_users;
        rec.add( pick_one(x) );
```

```
    }
    if ISP != "" {
        x = a.up.ISP and live_users;
        rec.add( pick_one(x) );
    }
    if Org != "" {
        x = a.up.Org and live_users;
        rec.add( pick_one(x) );
    }
    x = a.up.colleague and live_users;
    rec.add( pick_one(x));
    return rec;
}
```

Caching Probe Results: We perform limited caching at every node of the possible suspect results. In our implementation we implement caching of requests in two separate lists. We keep a long-lived list of entities, when the results of earlier probes have identified some form of network manipulation by those entities. We also keep a short-lived list of a successful result (along with the web page content).

The intuition behind keeping a long-lived list of violating entities is simple. If a given entity (an ISP or organization or country for example) is performing some kind of network manipulation, it is likely that a larger number of users from within that administrative entity will request checks. Hence, caching any possible failures will speed up any further requests coming from those users.

6.3 SiteViews User Interface

In this section, we describe the user interface of SiteViews and some of its functionality.

Settings: Figure 6.2 shows the initial settings screen that a user sees after they create a user name and password to log into SiteViews. As we can see, the country attribute was set to India for the study. We infer the ISP of the user using publicly availably IP registry information. However, as these publicly available information can be sometimes incorrect, this inferred ISP can be edited by the user. There is also an optional 'Organization' attribute that the user can set to indicate their company or organization.

We also allow users to edit two important settings. By default, we use a friend-sourcing depth of 2, i.e., we ask friends as well as friends of friends. This was

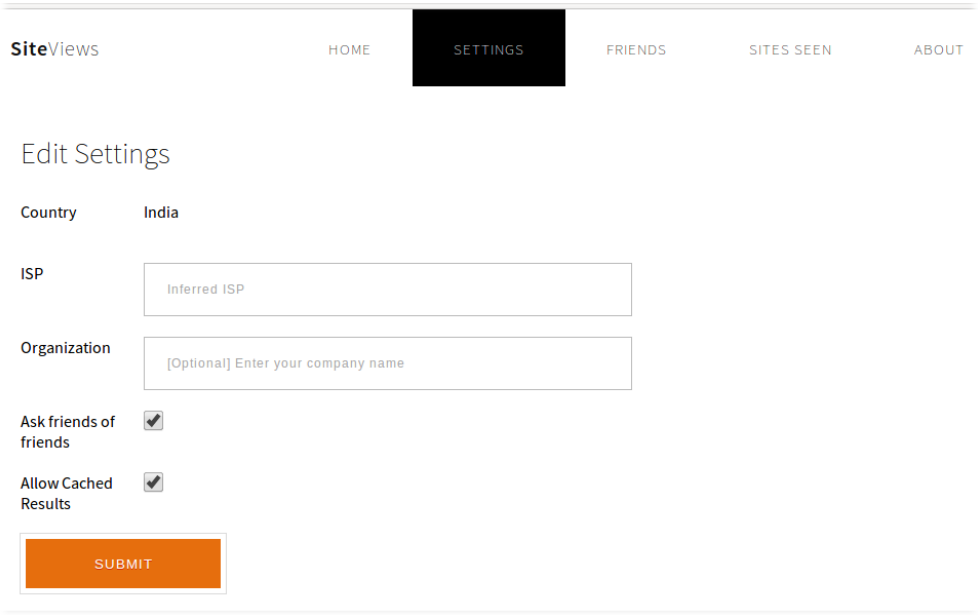


Figure 6.2: Network Settings for SiteViews

due to the coverage results in section 5.2 and redundancy results in section 5.3. Both the analytical results suggested that we get a large improvement in both coverage and redundancy when using depth 2 as opposed to depth 1. However, the user can disable this to limit themselves to just friends. Another attribute, that improves the efficiency is to cache results from previous users. This allows us to quickly respond to the user without putting undue load on the target web servers. However, the user can explicitly disable this as well. Note that this setting page is show again to the user, if we see them login from a different ISP. In case the user keeps using the same machine however, we redirect the user to the welcome screen bypassing the settings page.

Welcome Screen: Figure 6.3 shows the initial screen the user 'Alice' sees after logging in to SiteViews. The users have a simple interface to enter any URL they want to check. When the user submits a URL, we try connecting to the URL from within the Alice's browser using a signed applet. Results of this probe from the user's point of view are sent back to SiteViews for storing. SiteViews also connects the Alice's friends and friends of friends who are current online to get complete coverage. As we did not use any users outside of India, we used a proxy from within US to get coverage of the country.

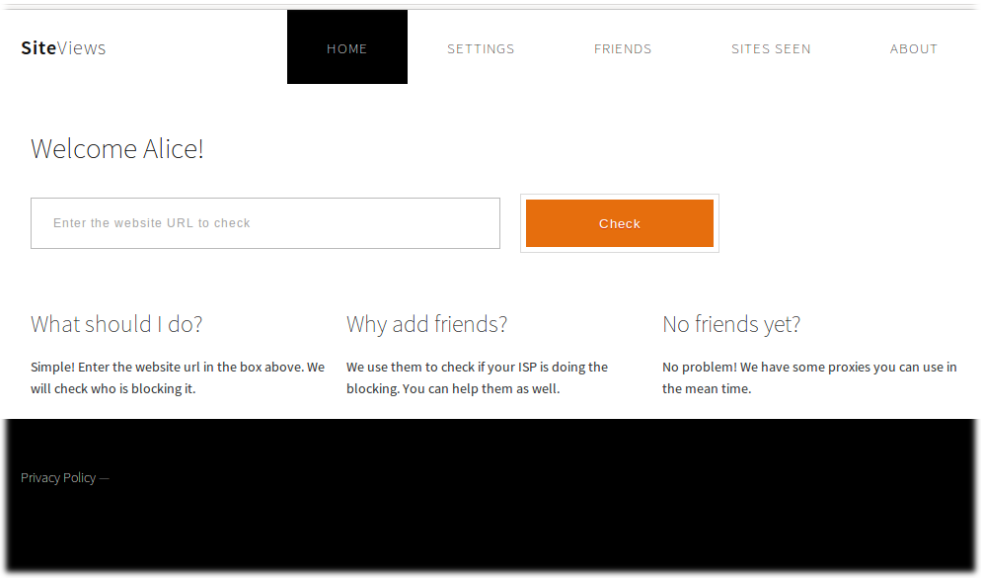


Figure 6.3: Initial Screen for SiteViews

After analysis of the status and data from all the probes from the recruited users, we analyze the results to determine the source of the network manipulation, if any. We redirect the user an analyzed page showing the results of the probe.

Analysis: Figure 6.4 shows the results of a user checking the website `www.bbc.co.uk` using SiteViews. The important information to the user is given at the bottom. It correctly identifies that there was no blocking from within the user's machine or the user's ISP. It also recognizes that there was a difference in content see from within the country and from the proxy in US. This is indicated by a differently colored status message ('Country: content changed') from the other two cases. In case the user had indicated an optional organization, we will have another status for the organization coverage result as well.

The search box and checked options at the top allow a user to 'Re-Check' the probes using different users, if available. This allows a user, who may not trust the results from the automatically recruited friends chosen by SiteViews to get another data point from other friends. Note also that we allow a user to gather information on a per attribute basis. For instance, the user can select the option 'Check ISP' while ignoring the option 'Check Country', if all they care about is ISP-coverage.

The user can click on any of the status messages at the bottom to get the data the led to that status from SiteViews. For instance, the user can click on the 'ISP-OK' message shown in figure 6.4 to see the resultant web page from the

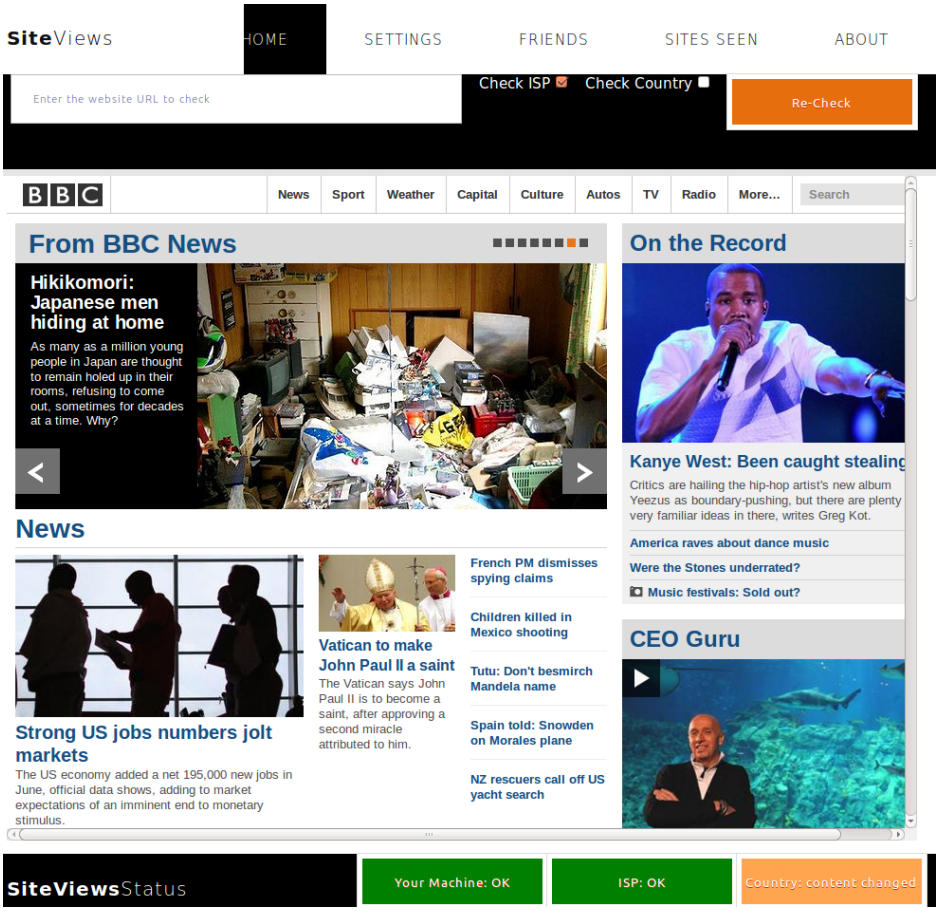


Figure 6.4: www.bbc.co.uk as seen by a user in India using SiteViews

user's friend that led to this analysis. In this particular instance, the user will not see any difference between the two different screen shots because there was no manipulation done by the ISP. However, the user is likely to be curious about how the web page is different as seen from outside the country. This she can do by clicking on the status 'Country: content changed'.

Blocked Websites: Figure 6.5 shows the results of a user trying to access a blocked website www.wapindia.net from within her ISP. This is a mobile based website that gives free content, such as songs and ring tones, for mobile phones. This may consist of illegal content. Furthermore, many ISPs in India are also wireless mobile service providers and may have an incentive to block free available content that competes with their own stores.

In this case, the Indian Music Industry (IMI) filed a suit against a large number of these web servers and had a court order (from a regional court) to block these sites [71], which included www.wapindia.net. However, this blocking

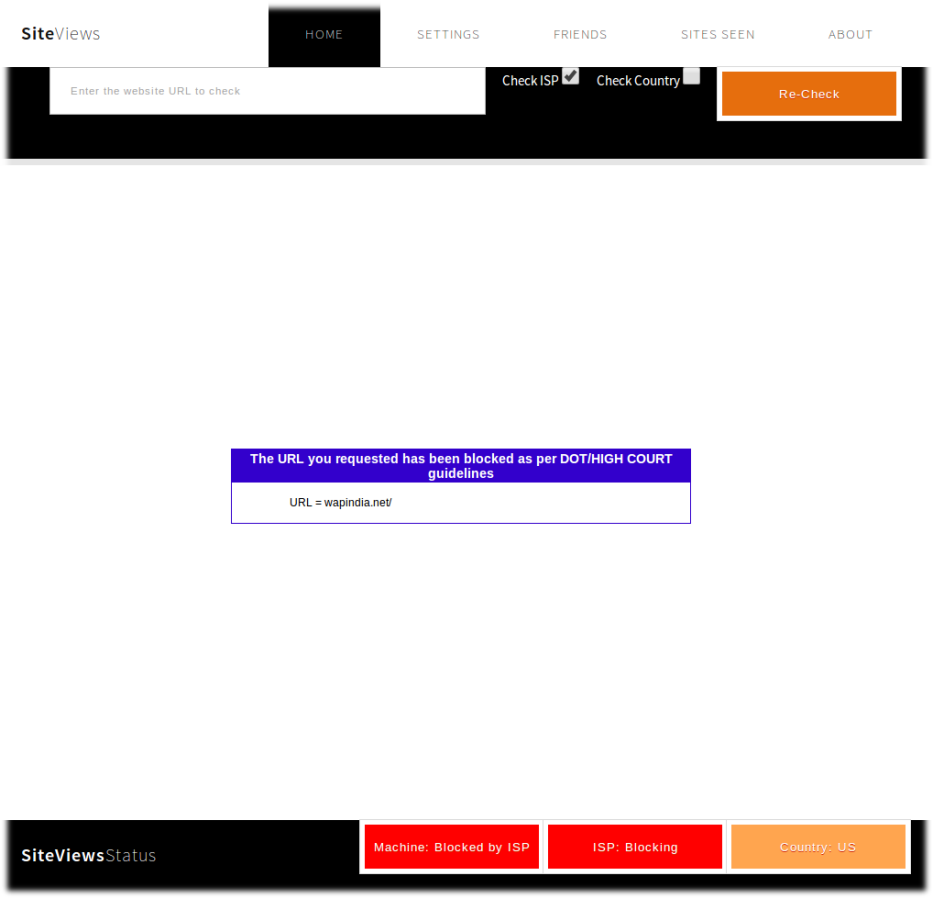


Figure 6.5: `www.wapindia.net` as seen by a user in India using SiteViews

was not uniformly applied by all the ISPs in India. Therefore we see in the figure 6.5that the site is accessible from outside the India as well as from other ISPs. The user can click on either of the statuses 'ISP: Blocking' or 'Country: US' to check how the website looks from another country or from a non-blocking ISP within India.

Figure 6.6 shows the user's interface after they click on the 'Country: US' status in figure 6.5. This allows the user to verify that the website is actually working as intended. In particular, this method of allowing a user to see the websites from their friend's, or in this particular case from the proxy's, perspective has many benefits. Firstly, it engages the user to explore which of her network providers are actually engaged in network manipulation. As a side effect, this also helps us gather information from multiple users to be stored in our cache. Secondly, it is often difficult to detect if there is any subtle content-level manipulation done by web servers or countries. Automated mechanisms will generate a lot of false

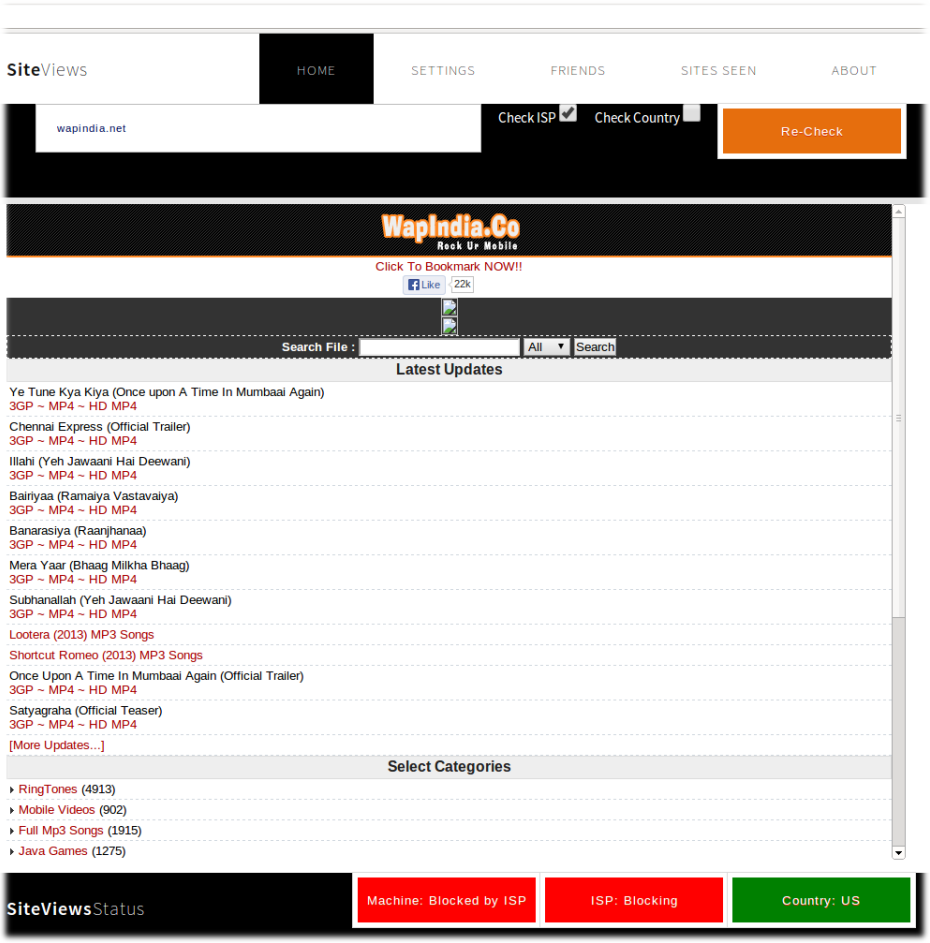


Figure 6.6: `www.wapindia.net` SiteViews

positives, such as different ads or benign localization done by web servers. By allowing the users to explore manipulation, we can let them judge the purpose of the manipulation on their own. Finally, by engaging users to stay on the website, we can ensure that they are available to help other users who may be trying to detect manipulation at the same time.

6.4 Study Setup

We give the results of field evaluation using real-life users on our prototype implementation of SiteViews. Our evaluation consisted of two small studies involving a group of 54 users, over a two week period in India. The users were recruited through word-of-mouth (with us starting by contacting two people). The choice

of India as a case-study was for two reasons: a) India had seen a spike in Internet censorship in the last year both imposed by the government and by ISPs, b) We would be reasonably sure of the *safe probing* assumption in Chapter 4.

Figure 6.7: User Locations for the study

The users came from 11 major metropolitan cities across India (Figure 6.7). A possible reason for the bias towards the major cities is that the initial set of users using the system are people working in the IT industry. The IT industry in India is concentrated among a few big cities (and their satellite cities), which is where the users live.

Figure 6.8 gives us the distribution of users over the ISPs we have seen in the study. As we can see 4 ISPs had the vast majority of the users in our study setup (55 out of the 64 users). This split can be explained by the relative size of customer bases for ISPs in India. A recent report [66], in July 2012, by the Telecom Regulatory Authority of India, gives the number of distinct broadband ISPs in India as 156. However, around 87% of the customer base belonged to just 4 ISPs.

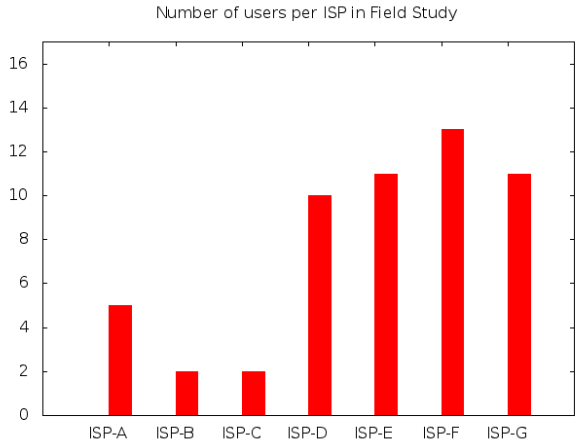


Figure 6.8: User distribution over ISPs

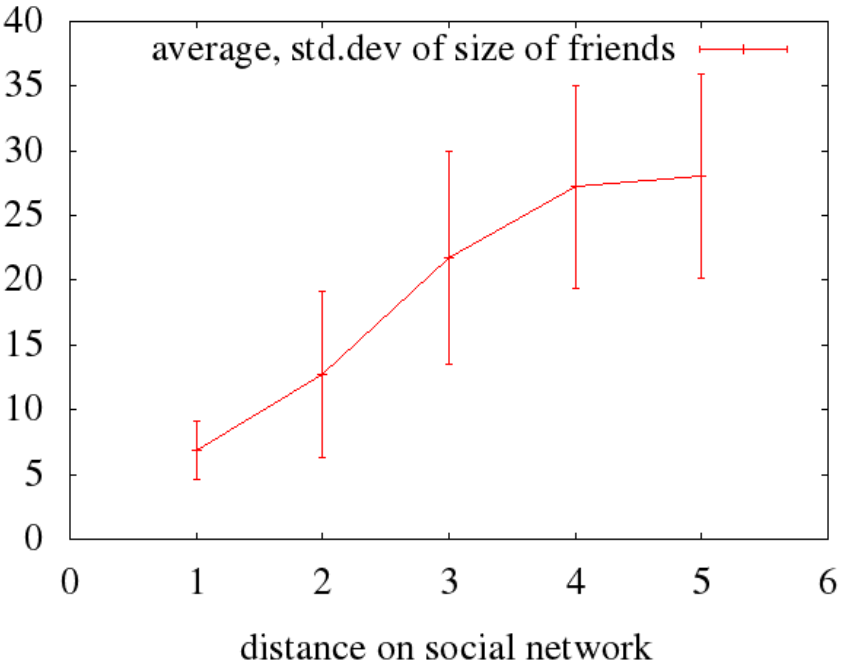


Figure 6.9: Average size of a user's friend-circle within given number of hops from the user

In our field-study, the maximum number of friends a user had was 12 (with 16 users having the minimum of just 1). As seen in Figure 6.9, however most users quickly get to the maximum clique sizes within 4 hops. This is a well-known characteristic of social networks, where people are within a few hops of the hubs in the social network. We added 4 proxies (1 outside India, 3 within India) as friends to every user to support users during the boot-strapping phase.

6.4.1 Coverage Metrics

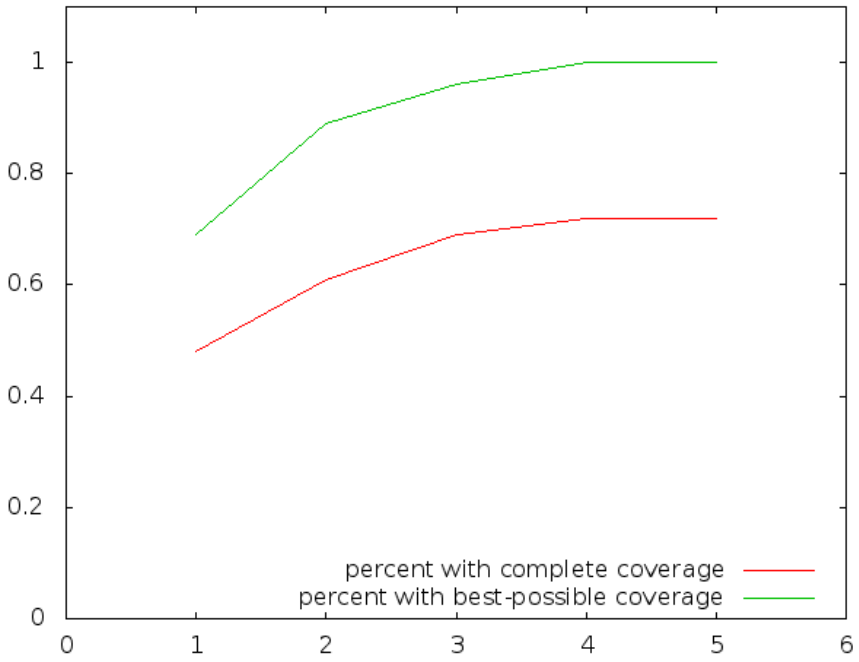


Figure 6.10: Percentage of users with complete and best-possible coverage at a given social network depth

As we discussed earlier in Section 5.2, complete coverage tells us, for any given social network depth, the percentile of users who can exactly detect the 'source' of network manipulation at that depth. Some users will never achieve complete coverage because there is no other user in the system (even at the maximum possible depth) that can disambiguate some network attribute. Figure 6.10 shows the coverage metrics of the users with regards to the social network depth. As shown, many users (89%) will to get their best possible coverage within just 2 hops of their social network. The lower percentile for the complete coverage is because

of users who have manually specified an organization (and do not have any other user from within that organization within the social network depth).

6.4.2 Redundancy Metrics

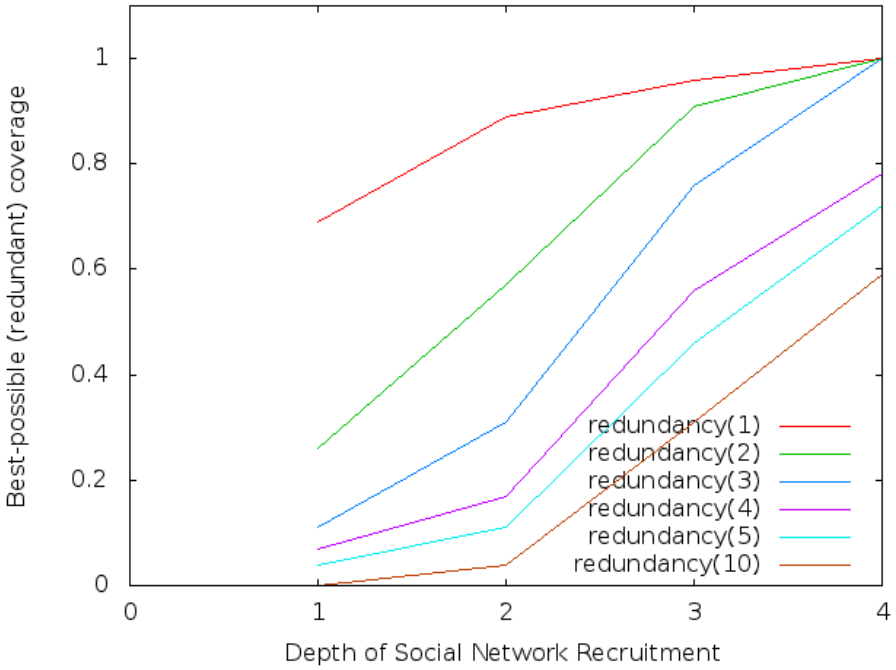


Figure 6.11: Best-possible (redundant) coverage at a given social network depth

As discussed earlier in Section 5.3, redundancy n tells us, for any given social network depth, the percentile of users who can have coverage with at least n number of users within the friend circle available for disambiguating the result. This is a useful metric when measuring the resiliency of a social network in detecting some network manipulation. For instance, a 90% of users with 5 redundant best-possible coverage at depth 2 tells that 90% of users have at least 5 users within two hops in the social network for each network attribute required to determine best-possible coverage.

Figure 6.11 gives us the best-possible coverage redundancy numbers for our study group. This shows that around half of the users had at least 2 different people for all network attributes, and around 25% of users had at least 3 within two hops. Note that this is a hard criteria to satisfy, because we need users for each network attribute. For a 3-redundancy coverage, we need to have 3 people with

the same ISP (and, optionally, same organization) as well as 3 people outside the ISP. This would be essential to clarify, for example, that the problem is not with the ISP but with the user's computer. Frequently, however, the users may only be

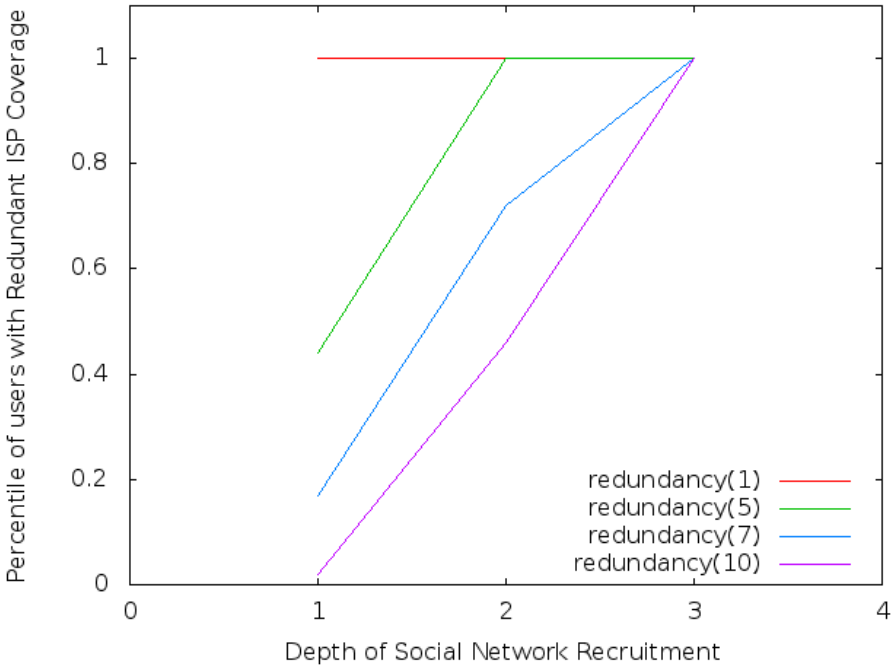


Figure 6.12: ISP (redundant) coverage at a given social network depth

concerned about a particular network attribute, say the ISP. Figure 6.12 gives us various ISP redundancy values. We see a high redundancy number for ISPs in our study group. This indicates that friends are not clustered together within a single ISP. The results for ISP coverages were 100% for 5-redundancy(at depth 2). Even at 10 redundancy, 46% of users satisfy ISP coverage at depth 2.

6.5 Web Manipulation Results

Our field evaluation was able to find different kinds of manipulation by ISPs of web content. In total we found 64 distinct blocked URLs within the country. On average users checked 15 distinct URLs per person. However, we found that both in terms of blocked URLs and in terms of total URLs being checked, there was a highly skewed distribution with top 6 users accounting for finding 51 of the blocked URLs (not uniquely, however). Many blocked URLs were also checked

for by a large group of users. We give more details of these URLs in Section 6.6. Most of the popularly checked for content pertained to media and torrent sites.

We were able to correlate some of the detected blocked URLs to court orders [71]. However, even in these mandated cases, not all ISPs blocked the websites uniformly. These manipulations differed by ISP, by target type, and motivation. We give an overview of some of our findings here.

6.5.1 Targets for censorship

We found that in India, the targets of censorship were varied. We categorized these blocked URLs in to broad categories, which were manually assigned.

- **Media Content and mobile:** Many sites hosting Indian media content (videos, songs) were blocked by ISPs. However, we found a good deal of variation amongst ISPs with regards to their interference. One particular ISP (ISP-G), which happens to own a significant media-empire (and an extensive mobile network), blocked more media content websites as well as websites advertising free mobile tunes compared to other ISPs.

- **Political or religious content:** Other common targets were websites that were seen as libelous or politically charged or liable to inflame religious conflict. Note that, around the time period of our study, there were instances of communal violence [72] in northeastern parts of India. The Indian government, purportedly, to prevent inflammatory religious content ordered censorship of a lot of content on the Internet.

  However, there was a lot of discussion in India complaining about the fact that other content from journalists and news articles were also being blocked. This meant that many of the URLs were being discussed in social networks and news articles at the time of the study, which may have led to us seeing some instances of these URLs in our results. We were able to correlate many blocked websites with court orders directing all ISPs in India to block those websites[71]. We also found that most of ISPs blocked these websites, but the some of the blocked websites were accessible from a few ISPs within India.

- **Torrent sites:** Websites which provide links to torrents were frequently (and universally) blocked by all the major ISPs in our study. In many cases,

the users are also given an explicit web page which warns them that these pages are blocked.

6.5.2 Types of network manipulation

We found multiple ways in which ISPs chose to implement network manipulation. We give a list of these below:

- **Timeout:** This was amongst the most popular mechanism that we observed. Many ISPs resorted to dropping the packets altogether, resulting in a time-out at the client. This was seen being applied to whole domains, as well as, only to specific web pages within otherwise allowed domains.
- **HTTP Protocol based manipulation:** HTTP protocol based errors were another popular way of modifying the content. In many cases, we saw HTTP 403 (Forbidden), or, HTTP 503 (Service Unavailable) for specific web pages. This would normally be interpreted as the browser (and the user) as a legitimate error from the web server at the other end. Note that these web pages were verified to be working properly from other ISPs (or outside India). Some other errors were seen as HTTP protocol errors by the Java networking stack and API.
- **Content modification:** Some ISPs resorted to showing a different content from that of the actual web page. We saw on instance of a web page that simply served ads in place of a blocked web page. Two ISPs, in particular, performed their censorship by simply returning a empty web page consisting of just the following content to the user `<BODY></BODY>`. There were also many instances of web pages that explicitly indicated that they were blocked by the ISP. Figure 6.13 shows a media content site `www.desisong.com` being blocked by one of the ISPs.

6.6 Metrics of Network Manipulation

In this section, we look at analysis of the various blocked URLs, ISP mechanisms for blocking, categorize the blocked URLs by their type.

(a) Website as seen originally

(b) Website blocked

Figure 6.13: Coverage and False-Positive Rates

isp	OK	timeout	Error	403	401	504	502	503
ISP-A	35	19	0	0	0	0	0	0
ISP-B	33	19	0	0	2	0	0	0
ISP-C	51	1	0	2	0	0	0	0
ISP-D	25	1	0	20	0	0	0	8
ISP-E	26	24	0	0	0	4	0	0
ISP-F	51	2	1	0	0	0	0	0
ISP-G	25	5	1	0	0	0	23	0

Table 6.1: Table showing incidences of blocking mechanism used per ISP

6.6.1 Blocking mechanisms by ISP

Table 6.1 shows how the various blocked URLs are treated by different ISPs. As shown in the figure, there was no uniformity in blocking the URLs by all the ISPs. The OK, column corresponds to URLs that we accessible (with or without modification) by the user. The timeout is the most popular mechanism for blocking. This was an indication that the packets were dropped by the ISP. HTTP status codes 403 (forbidden web-page) and 502 (bad gate-way) were the other popular blocking messages sent by the ISPs.

Note that although some of these ISPs (ISP-F and ISP-C) appear to be more permissive, in reality these web pages delivered to the users were not always the ones sent by the web servers. In many of these cases, the web pages were either of empty content or with the modified content `<BODY> </BODY>`. Some ISPs also gave a warning web page explicitly indicating that the website was blocked(see Figure 6.13).

ISP	file-hosting	mobile	political	blog	media	adult	religious	torrent
ISP-A	1	1	2	3	5	1	2	5
ISP-B	2	0	1	3	10	2	2	3
ISP-C	0	0	1	0	1	2	1	0
ISP-D	2	2	2	2	12	1	2	6
ISP-E	1	2	2	2	14	1	3	3
ISP-F	0	0	0	1	0	1	0	2
ISP-G	1	5	1	2	17	0	2	1

Table 6.2: Table showing frequency of types of content blocked by URLs per ISP

6.6.2 Blocked URLs categories by ISP

Table 6.1 shows the various how the frequencies various blocked categories of URLs by different ISPs. These categories were assigned to the urls, by us, manually. Most of the category labels were self-evident; for instance, a free file-hosting site was something that allow uploading and storing of content, a religious content website was something that contained content specifically praising (or criticizing) some religion etc.,.

As is evident, quite a bit of the blocking that we detected was to media content which consisted of sites which contained music or video files of popular (mostly Indian) content. Torrent sharing sites were also a popular target. While the number of instance of adult websites that were seen to be blocked were minimal, this may have been due the fact that users did not feel comfortable using our tool to check for those website.

Chapter 7

Conclusion and Future Work

Network manipulation is a growing concern for a variety of reasons. Many approaches have been presented in recent past to detect and circumvent such interference. A fundamental challenge in many such approaches is the procurement of large collection of end nodes to perform the measurements. End users who want to perform customized checks for detecting manipulation on specific end targets do not have the capabilities to develop such a collection easily and on demand. Automated analysis using publicly available infrastructure on the web, while useful, provides only relatively limited coverage.

We have demonstrated the feasibility of using friendsourcing, a process in which a user can use the capabilities of their social network to perform an on-demand investigation of network manipulation. We developed, a prototype tool SiteViews, that was able to detect various kinds of network manipulation in India and was able to attribute them to their likely actors. We also showed the scalability of our approach using evaluations done on data taken from three different kinds of real-life social networks.

There are multiple possible directions for future work. Firstly, we can make our measurement Java Applet more generic to measure new kinds of manipulation. Secondly, we can measure the effectiveness of the approach compared to crowdsourcing. Our current implementation of the tool, gave the users a *quid pro quo*, i.e., they will receive help from their friend circle in exchange for providing the same. However, it is also interesting to measure whether, users will be willing to help random strangers with their queries. Existing research on cooperation on social networks suggests that users will likely help their friends more than random strangers, but even limited cooperation might turn out to be valuable. Finally, we can also apply existing research on handling malicious or unreliable users on the social network to our implementation. While it was not a large concern in our prototype implementation, with users being limited to friends-of-friends, it will be bigger concern once we start accepting help from less trusted users.

References

[1] Reporters Without Borders, "Beset by online surveillance and content filtering, netizens fight on," http://en.rsf.org/beset-by-online-surveillance-and-12-03-2012,42061.html, March 2012, [Accessed June 9, 2013].

[2] Electronic Frontier Foundation, "Australian Networks Censor Community Education Website," https://www.eff.org/deeplinks/2013/04/australian-networks-censor-community-education-site, April 2013, [Accessed June 9, 2013].

[3] M. S. Bernstein, D. Tan, G. Smith, M. Czerwinski, and E. Horvitz, "Personalization via friendsourcing," *ACM Trans. Comput.-Hum. Interact.*, vol. 17, pp. 6:1–6:28, May 2008. [Online]. Available: http://doi.acm.org/10.1145/1746259.1746260

[4] TechCrunch, "Facebooks Growth Since IPO In 12 Big Numbers," http://techcrunch.com/2013/05/17/facebook-growth/.

[5] S. P. Kasiviswanathan, S. Eidenbenz, and G. Yan, "Geography-based analysis of the internet infrastructure," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 131–135.

[6] Y. Zhang, Z. M. Mao, and M. Zhang, "Detecting traffic differentiation in backbone ISPs with NetPolice," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC'09. ACM, 2009, pp. 103–115.

[7] CNET, "Router glitch cuts Net access," http://news.cnet.com/2100-1033-279235.html, [Accessed May 30, 2013].

[8] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of BGP security issues and solutions," in *Proceedings of the IEEE*, vol. 98, no. 1. IEEE, 2010, pp. 100–122.

[9] RIPE, "YouTube Hijacking: A RIPE NCC RIS case study," http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study, March 2008, [Accessed 1 June 2013].

[10] ArsTechnica, "Police arrest suspect accused of unprecedented DDoS attack on Spamhaus," http://arstechnica.com/security/2013/04/police-arrest-spamhaus-attacker-accused-of-unprecedented-ddos-attack/, April 2013, [Accessed 1 June 2013].

[11] ArsTechnica, "US net neutrality rules finalized, in effect November 20," http://arstechnica.com/tech-policy/news/2011/09/us-net-neutrality-rules-finalized-in-effect-november-20.ars, September 2011, [Accessed June 8, 2013].

[12] ArsTechnica, "Verizon sues to halt FCC's net neutrality rules," http://arstechnica.com/tech-policy/news/2011/10/verizon-sues-to-halt-fccs-net-neutrality-rules.ars, October 2011, [Accessed June 8, 2013].

[13] ArsTechnica, "Net neutrality supporters file lawsuit against net neutrality rules," http://arstechnica.com/tech-policy/news/2011/09/net-neutrality-backers-file-lawsuit-against-net-neutrality-rules.ars, September 2011, [Accessed June 8, 2013].

[14] VoIP Solutions and conferencing reviews, "10 ISPs and countries known to have blocked VoIP," http://www.voip-sol.com/10-isps-and-countries-known-to-have-blocked-voip/, January 2007, [Accessed June 9, 2013].

[15] ArsTechnica, "Vonage claims unfair "tax" by Canadian ISP," http://arstechnica.com/old/content/2006/03/6339.ars, March 2006, [Accessed May 31, 2013].

[16] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost: enabling end users to detect traffic differentiation," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, ser. NSDI'10. USENIX, 2010, pp. 405–418.

[17] Cisco, "Cisco SCE 2000 Series Service Control Engine," http://www.cisco.com/en/US/products/ps6151/index.html, [Accessed June 9, 2013].

[18] T. Elahi and I. Goldberg, "CORDON–A Taxonomy of Internet Censorship Resistance Strategies," University of Waterloo, Tech. Rep., 2012.

[19] Electronic Frontier Foundation, "Packet Forgery By ISPs: A Report on the Comcast Affair," https://www.eff.org/wp/packet-forgery-isps-report-comcast-affair, November 2007, [Accessed on June 7, 2013].

[20] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver, "Detecting In-Flight Page Changes with Web Tripwires," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. USENIX, 2008, pp. 31–44.

[21] M. S. Granovetter, "The Strength of Weak Ties," *American journal of sociology*, pp. 1360–1380, 1973.

[22] Facebook, "Anatomy of Facebook," https://www.facebook.com/notes/facebook-data-team/anatomy-of-facebook/10150388519243859, November 2011, [Accessed June 9, 2013].

[23] Google, "Google Plus," https://plus.google.com.

[24] Pew Internet, "Pew Internet: Social Networking," http://pewinternet.org/Commentary/2012/March/Pew-Internet-Social-Networking-full-detail.aspx, December 2012, [Accessed June 13, 2013].

[25] "DBLP," http://www.informatik.uni-trier.de/~ley/db/.

[26] "Arxiv," "http://www.arxiv.org".

[27] "KDL Dataset," "http://kdl.cs.umass.edu/data/hepth/hepth-info.html".

[28] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'11. ACM, 2011, pp. 1082–1090.

[29] Amazon, "Amazon Mechanical Turk," https://www.mturk.com/mturk/welcome.

[30] M. R. Morris, J. Teevan, and K. Panovich, "What Do People Ask Their Social Networks, and Why?: A Survey Study of Status Message Q&A Behavior," in *CHI'10: Proceedings of the 28th international conference on Human factors in computing systems*, ser. CHI'10. ACM, 2010, pp. 1739–1748.

[31] S. Leider, M. M. Mbius, T. Rosenblat, and Q.-A. Do, "Directed Altruism and Enforced Reciprocity in Social Networks," vol. 124, no. 4, pp. 1815–1851, 2009.

[32] M. S. Granovetter, *Getting a job: A study of contacts and careers*. Harvard University Press (Cambridge, Mass), 1974.

[33] K. Panovich, R. Miller, and D. Karger, "Tie strength in question & answer on social network sites," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ser. CSCW'12. ACM, 2012, pp. 1057–1066.

[34] E. Gilbert and K. Karahalios, "Predicting Tie Strength with Social Media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI'09. ACM, 2009, pp. 211–220.

[35] J. H. Fowler and N. A. Christakis, "Cooperative behavior cascades in human social networks," in *Proceedings of the National Academy of Sciences*, vol. 107, no. 12. National Academy of Sciences, 2010, pp. 5334–5338.

[36] P. Brañas-Garza, R. Cobo-Reyes, M. P. Espinosa, N. Jiménez, J. Kovářík, and G. Ponti, "Altruism and social integration," vol. 69, no. 2. Elsevier, 2010, pp. 249–257.

[37] B. Christianson and W. Harbison, "Why Isn't Trust Transitive?" in *Security Protocols*, ser. LNCS'97. Springer-Verlag, 1997, pp. 171–176.

[38] T. DuBois, J. Golbeck, and A. Srinivasan, "Predicting Trust and Distrust in Social Networks," in *SocialCom/PASSAT'11*. IEEE, 2011, pp. 418–424.

[39] U. Kuter and J. Golbeck, "Using Probabilistic Confidence Models for Trust Inference in Web-based Social Networks," in *ACM Transactions on Internet Technology*, ser. TOIT'10. ACM, 2010, pp. 8:1–8:23.

[40] C.-N. Ziegler and G. Lausen, "Spreading Activation Models for Trust Propagation," in *e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004 IEEE International Conference on*. IEEE, 2004, pp. 83–97.

[41] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of Trust and Distrust," in *Proceedings of the 13th international conference on World Wide Web*, ser. WWW'04. ACM, 2004, pp. 403–412.

[42] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting Positive and Negative Links in Online Social Networks," in *Proceedings of the 19th international conference on World Wide Web*, ser. WWW'10. ACM, 2010, pp. 641–650.

[43] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi, "Detecting BitTorrent Blocking," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, ser. IMC'08. ACM, 2008, pp. 3–8.

[44] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the Edge Network," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. ACM, 2010, pp. 246–259.

[45] P. Kanuparthy and C. Dovrolis, "ShaperProbe: End-to-End Detection of ISP Traffic Shaping using Active Methods," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC'11. ACM, 2011, pp. 473–482.

[46] M. B. Tariq, M. Motiwala, N. Feamster, and M. Ammar, "Detecting Network Neutrality Violations with Causal Inference," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, ser. CoNEXT'09. ACM, 2009, pp. 289–300.

[47] Herdict Web, "OpenNet Initiative's Herdict Web," http://www.herdict.org/, [Accessed 31 May 2013].

[48] R. Beverly, S. Bauer, and A. Berger, "The Internet Is Not a Big Truck: Toward Quantifying Network Neutrality," in *Passive and Active Network Measurement*, ser. PAM'07. Springer-Verlag, 2007, pp. 135–144.

[49] M. A. Sánchez, J. S. Otto, Z. S. Bischof, and F. E. Bustamante, "Dasu - ISP characterization from the edge: a BitTorrent implementation," in *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*, ser. SIGCOMM'11. ACM, 2011, pp. 454–455.

[50] Y. Zhang, Z. Morley, and M. M. Zhang, "Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs," in *In Proc. 7th ACM Workshop on Hot Topics in Networks*, ser. HotNets'08, 2008.

[51] A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis, "CensMon: A Web Censorship Monitor," in *FOCI11, USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2011.

[52] Anonymizer, "Anonymizer," http://www.anonymizer.com.

[53] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger, "Thwarting Web Censorship with Untrusted Messenger Discovery," in *Privacy Enhancing Technologies*, ser. PET'03, March 2003, pp. 125–140.

[54] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, "Infranet: Circumventing web censorship and surveillance," in *Proceedings of the 11th USENIX Security Symposium*. USENIX, 2002, pp. 247–262.

[55] S. Burnett, N. Feamster, and S. Vempala, "Chipping Away at Censorship Firewalls with User-Generated Content," in *Proceedings of the 19th USENIX conference on Security*, ser. USENIX Security'10. USENIX, 2010, pp. 463–468.

[56] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman, "Telex: Anticensorship in the Network Infrastructure," in *Proceedings of the 20th USENIX Security Symposium*, August 2011.

[57] A. Houmansadr, G. T. Nguyen, M. Caesar, and N. Borisov, "Cirripede: Circumvention Infrastructure using Router Redirection with Plausible Deniability," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS'11. ACM, 2011, pp. 187–200.

[58] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer, "Decoy Routing: Toward Unblockable Internet Communication," in *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet*, ser. FOCI'11. USENIX, August 2011.

[59] atthegrapevine, "North Korea, Syria and Iran - just three of the places where the Internet goes to die," http://www.atthegrapevine.com/politics/north-korea-syria-and-iran-just-three-of-the-places-where-the-internet-goes-to-die, 2013, [Accessed on June 29,2013].

[60] ncr-iran.org, "Iranian regime arrests man for selling Internet filtering software," http://www.ncr-iran.org/en/news/human-rights/13354-iranian-regime-arrests-man-for-selling-internet-filtering-software.html, [Accessed on June 29,2013].

[61] Guardian, "New laws on porn 'will criminalise thousands'," http://www.guardian.co.uk/artanddesign/2008/oct/26/photography-pornography-law, [Accessed on June 29,2013].

[62] KSL, "Bill Would Make Viewing Porn on School Computers a Crime," http://www.ksl.com/?nid=148&sid=804683, [Accessed on June 29,2013].

[63] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil Nodes using Social Networks," in *Proceedings of the Network and Distributed System Security Symposium*, ser. NDSS'09, 2009.

[64] Facebook, "Graph API," https://developers.facebook.com/docs/reference/api/.

[65] Google, "Google Maps API," https://developers.google.com/maps/, [Accessed on June 20, 2013].

[66] Telecom Regulatory Authority of India, "The Indian Telecom Services Performance Indicators," http://www.trai.gov.in/WriteReadData/PIRReport/Documents/Indicator%20Reports%20-%20Jun-12.pdf, October 2012, [Accessed June 1,2013].

[67] "Whois," http://en.wikipedia.org/wiki/Whois.

[68] "ip2location," http://www.ip2location.com/.

[69] Nginx, "Nginx," http://nginx.org.

[70] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The Anatomy of the Facebook Social Graph," in *arxiv.org/CoRR*, 2011.

[71] ArsTechnica, "SOPA masala: 387 Indian ISPs must block 104 piratical websites," http://arstechnica.com/tech-policy/2012/03/sopa-masala-indian-isps-must-block-104-piratical-websites/, [Accessed May 30,2013].

[72] wikipedia, "2012 Assam Violence," http://en.wikipedia.org/wiki/2012_Assam_violence, [Accessed on June 15, 2013].