Blocking-Resistant Network Services using *Unblock*

Will Scott, Raymond Cheng, Jinyang Li, Arvind Krishnamurthy, Thomas Anderson

Abstract

The desire for uncensored access to the Internet has motivated the development of both open proxies like Tor and social graph-based overlays like FreeNet. However, neither design is sufficient, as relays in open proxies are easily exposed and blocked, and overlays based just on social trust suffer from poor availability and performance. In this paper, we introduce the design for a new overlay service, *Unblock*, constructed from an augmented social graph. In *Unblock*, multi-hop paths through social links protect individual participants from exposure to adversaries. *Unblock* achieves good performance by introducing additional links in the network graph in a manner that minimizes vulnerability. We also develop several transport level techniques for improved latency. We demonstrate the practicality of the system for web traffic workloads.

1 Introduction

Unfettered digital communication, as provided by the Internet, has fundamentally changed the world in countless ways. Businesses, organizations, and citizens have benefited from the Internet's global reach. However, the Internet was not designed to be resilient to censorship, and governments have restricted communication to advance their social and economic agendas[38, 28]. Worse, network equipment providers have shown a willingness to commoditize and profit from censorship by selling interception and filtering devices[48].

Censorship today is more than aggressive suppression of activists by oppressive governments. "Soft" censorship is rapidly becoming a pervasive force in Internet communications. We define soft censorship as the manipulation of communications without accompanying repercussions for user actions. Selectively cropping content from a person's daily news feed and their friends' status updates can significantly alter the way they perceive the world[11, 12]. Soft censorship allows nations to exert economic pressure[14] by blocking foreign competitors from entering markets, enabling new forms of protectionism in the multi-trillion dollar[6] Internet economy.

Like many security problems, Internet censorship is not purely technical. If the adversary is a Government or ISP, they can defeat any circumvention network through extreme measures: whitelisting approved sites, outlawing encrypted traffic, forcing users to use hardware and software controlled by the government, or just blocking all international traffic. In reality, such measures carry significant political, social, and economic costs [1]. Unlike more aggressive forms of censorship, circumvention of soft censorship is not typically illegal, and such circumvention already occurs on a wide scale. The nature of the problem makes it hard to accurately report, but surveys have found anywhere from "at most 3% of the population"[45] to "25% of Chinese netizens"[32] to "58% of bloggers"[46] have used circumvention tools.

This paper presents *Unblock*, a system resistant to soft censorship. It provides access to websites in the face of current censorship techniques including IP address blacklisting, DNS poisoning, and keyword filtering. *Unblock* is designed to help the majority of users who want to access legal, but censored content. We assume users face little risk if they are detected by the censor beyond Internet disruption. It is not intended for use in countries with hard censorship.

Previous research on censorship-resistant networks has focused on routing-level network designs[57, 33, 27] and overlay systems[17, 54, 21], but neither of these have proven to be well suited for soft censorship. Routing-level designs require widespread physical deployments to be effective, which has a high startup cost. Previous attempts at censorship resistant overlay systems either don't scale (in the case of single hop proxies), or focus on anonymity at the cost of poor performance[53, 18].

Unblock is based on a third class of censorship-resistance which rests on trusted social connections. By asking users to explicitly connect with friends who they trust to conceal their identity, *Unblock* forms a global social network. Traffic is routed over these links to participants willing to relay traffic out of the overlay (which we call "exit nodes") in a region where the desired content is not censored. Multi-hop routing, coupled with security mechanisms to prevent overlay disruption, hide overlay participants from the censor.

Unfortunately node degrees in social networks exhibit a power law distribution where many users only have a small number of friends. *Unblock* improves performance and availability by introducing randomized shortcut links, untrusted connections that risk exposing a small set of users to an adversary in order to dramatically lower the median latency to an exit node. The system also employs a custom set of transport mechanisms optimized for such a multi-hop overlay.

In order to demonstrate that *Unblock* is practical for wide-spread use, we implement it as an experimental extension to OneSwarm[25], a popular social overlay-

[1]The Egyptian government's Internet shutdown in 2011 was seen to popularize rather than suppress anti-government feelings[24].

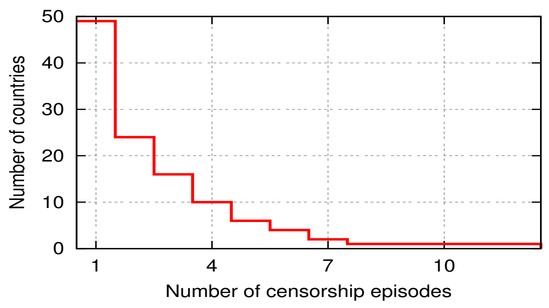

Figure 1: Number of observed censorship episodes against Tor (i.e. blocking Tor when it was previously not blocked).

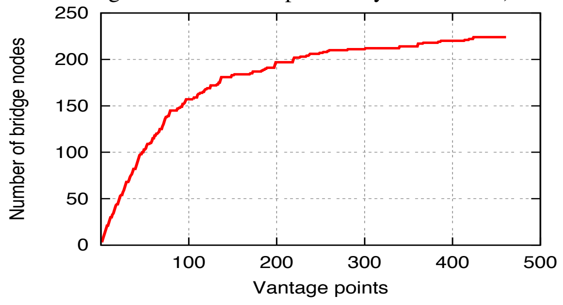

Figure 2: Number of discovered Tor bridge nodes vs number of PlanetLab vantage points used in the crawling.

based bulk file sharing system. We evaluated the performance of the prototype in controlled testbed settings and measured its ability to perform web requests under various configurations. We also evaluated the social network augmentation techniques using a simulator to measure the implications of our design decisions at scale. Our measurements show that *Unblock* provides high availability and improved performance with minimal risk of exposure of participants to an adversary.

2 Challenges in Building Censor Resistant Overlays

Existing overlays are unsuitable for providing blocking resistant services. Public open-access overlays like Tor are easily blocked by governmental censors while social network-based overlays have poor path availability and connectivity. Moreover, multi-hop traffic forwarding over overlays is slow, resulting in a low quality web browsing experience for users. In this section, we quantify these limitations to motivate the design of *Unblock*.

2.1 Open Access Overlays are Easily Blocked

Public open-access overlays are characterized by: (a) the fact that any client can use their relays to construct a circuit for routing traffic, (b) their use of a centralized management system that publishes information regarding relays. These include anonymizing overlays such as Tor[17], Ultrasurf[54], and Freegate[21], as well as open proxies that are used to evade censorship[23]. Regardless of their original intended purpose, open access overlays are a popular way for users in censored countries to reach otherwise blocked websites[23].

Fundamentally, open access overlays are vulnerable to blocking because of their centralized mechanisms for distributing relay addresses to users. For example, Tor provides a few well-known directory servers that return certified lists of relays. As the censor can look up or crawl all relays, these systems are as blockable as the very websites they want to provide access to.

To quantify the extent to which countries actively block access to Tor, we used data collected by Tor that tracked Tor access in 152 countries over the period from January 2010 to November 2011[52]. The number of clients that connected to each of the Tor directory servers were aggregated into two week periods by country. These totals were compared with the preceding period. Finally, these ratios were normalized to the total number of Tor users around the world for the two corresponding periods. The two week period acts as a low pass filter, minimizing variations in usage. By normalizing against the global user count, the analysis also accounts for overall trends in Tor usage.

We then define a censorship episode as an event where the Tor usage in a country where Tor is normally unblocked drops more than four standard deviations below expectation. Figure 1 illustrates the results from this analysis. We were surprised to see that out of 152 countries, 49 countries had a censorship episode during the two year period, and several had multiple such episodes.

In response to these blockages, Tor has added semi-secret relays called *bridge nodes*. Unfortunately, the same mechanisms implemented to help users find bridges can be abused to identify and block bridges. Although Tor limits the number of bridges exposed to any given IP, this restriction is ineffective against a resourceful censor in possession of a diverse set of IP addresses.[2] For Figure 2, we crawled the Tor bridge discovery mechanism from multiple vantage points on PlanetLab. We found that by requesting bridge nodes from multiple locations, we were able to discover the IP addresses of nearly 240 bridge nodes in Tor. In fact, this included almost all of the Tor bridge nodes that were distributed through HTTP during the measurement period[5]. One could easily imagine a censor using similar crawling techniques to find and block bridge nodes. Reports confirm that China already blocks bridge nodes[53].

2.2 Social Network Based Overlays Have Poor Connectivity

Social network overlays have been explored in the past to improve trust and security. Examples include

[2]Tor also uses other mechanisms for distributing bridge nodes, such as automatic email responses to email queries for bridge nodes from Gmail accounts. These mechanisms are equally susceptible to crawling attacks, especially since researchers have demonstrated that one can acquire Gmail accounts for about $0.30 per account[37].

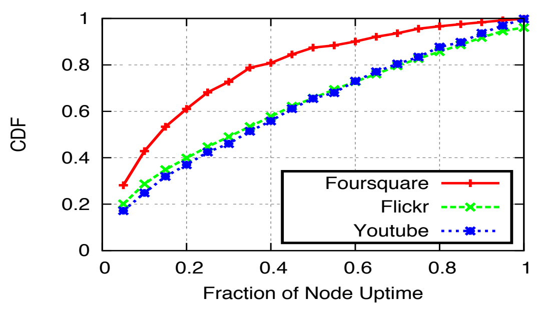

Figure 3: Fraction of nodes with paths to exit nodes on different social network datasets for varying node uptimes and with 10% of the nodes being exit nodes.

the Ostra[36] email service, the OneSwarm[25] system for anonymous P2P file-sharing, and the Drac[15] anonymizing overlay.

Social overlays route user traffic to "exit nodes", nodes located in non-censored domains willing to make connections on behalf of other users, in order to provide access to blocked websites. Social overlays are an attractive option for resilient service because the network can be formed in a completely decentralized fashion. As each user joins the overlay by connecting to his explicitly trusted peers, no single user (including the censor) can discover the identities of more than a few participants.

Unfortunately, availability in social overlays tends to suffer from sparse connectedness. We measured the network properties of a number of social networks, including Youtube, Flickr and Foursquare using datasets collected by[35, 47]. These measured networks are likely to have much denser connectivity than a social overlay formed for censorship circumvention. Nevertheless, most nodes in the measured networks have at most a handful of links to other peers and a large number of users have only one social link. This is particularly problematic in a setting where users, essential for connectivity, nevertheless come and go as typical for a P2P system.

We simulated the availability of paths through these social networks under varying churn (percent of time users spend online) when 10% of participants serve as exit nodes. Figure 3 shows the results of this experiment. We find that the availability of working paths is highly susceptible to churn. Due to the *stringy* nature of these social networks, churn disconnects some nodes entirely from any exit node, lowering the total connectivity to exit nodes from 100% to around 50% for typical node uptimes seen in peer-to-peer systems[51, 22, 44, 31]. Performance is also likely to suffer as connectivity degrades under churn. In our simulations, a small set of nodes relay hundreds or even thousands of paths making them critical to the success of the system. The performance of these networks is thus limited by the availability and bandwidth of this small set of nodes, reducing overall system robustness.

Site	Direct (ms)	Tor (ms)	Slowdown
Google	360	6,055	16.8
Facebook	649	7,982	12.3
Amazon	1758	13,982	8.0
Twitter	1588	9,306	5.9
Yahoo	1419	10,546	7.4

Figure 4: Slowdown introduced by Tor for loading popular domains in January 2012.

2.3 Overlays Have Poor Transport Performance

Both open-access and social-network based overlays suffer from transport inefficiencies that undermine their usability. In general, overlay networks have suboptimal performance as data is transferred multiple times across the overlay, resulting in higher latency and greater possibility of traversing congested links.

To characterize the latency of overlay communications, we performed measurements of page load times over Tor compared with "normal" direct connections from a set of 17 geographically diverse PlanetLab nodes. As can be seen in Figure 4, the page load time for popular sites increases by a factor of up to 20x when web pages are loaded over Tor. This is consistent with the issues outlined in[52, 18]. Figure 5 shows the same experiment for the Alexa top 100 sites [1], the median page load time is increased from 2.3s to 12.6s.[3]

The performance inefficiencies of overlay networks are generally attributed to three factors. First, forwarding traffic over multiple end-hosts limits the throughput to the slowest link. For example, in Oneswarm, multi-hop overlay paths have an average throughput of only 29 KBytes/s, leading to poor performance unless multiple paths are used[25]. Second, in Tor, all traffic between any pair of overlay nodes, even if they represent circuits for different clients, are multiplexed over a single TCP connection. This mixing results in interference across circuits during congestion control and large queueing delays[42]. Third, location is an important factor in overlay path selection. Nodes located nearby geographically often have low latency and are connected by relatively few intermediate routers. One of the benefits provided by social network based overlays is that the majority of social links connect geographically close nodes[47]. In contrast, many open access overlays do not use location as a factor in path selection, resulting in higher average per-hop and end-to-end latencies. Systems like Tor are slow precisely for these reasons, and not the availability of resources in the system as a whole.

[3]Roughly 20% of Alexa top 100 domains did not load successfully during our experiment. Causes include: no main page (e.g. googleusercontent.com, bp.blogspot.com), blocking either by institution (in the case of pornography) or a censor (e.g. taobao, qq). Many Chinese sites are not accessible either from Tor or PlanetLab due to network-level blocking.

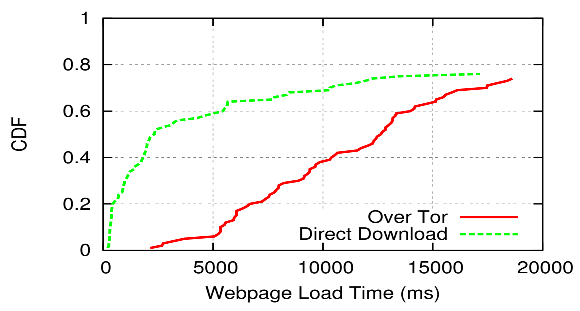

Figure 5: Load times for Alexa's top 100 domains in January 2012. Measurements are averaged from 17 geographically diverse PlanetLab nodes.

3 System Design

In this section, we describe the design of *Unblock* that aims to combine the privacy, security, and locality properties of social overlays with the flexible and robust connectivity of open access overlays. We first discuss the adversarial model that we use to inform the design of our system, outline the overall approach given the adversarial model and the challenges raised in the previous section, and then provide a detailed description of the various mechanisms.

3.1 Adversaries and Threat Model

We model our censor based on *existing* soft censorship practices seen in many parts of the world. Content censorship is performed using technical, rather than police, means – i.e., the censor silently blocks or alters access to certain sites but does not impose real-world punishments on users for using anti-blocking software. We assume that the censor has direct control over the routers inside the censored domain and is able to disrupt communications based on matching patterns in the packet header or content. Studies have confirmed that censors exploit this control to block content by polluting DNS entries, blocking IP addresses, or blacklisting keyword terms[41, 4].

The censor is also able to infiltrate a limited number of social links, giving it access to the overlay. It is free to generate, modify, and delete protocol messages flowing through nodes it controls, record timing and other information, and correlate traffic from multiple nodes. We call adversary controlled nodes *moles*, since they infiltrate the social network to spy on and disrupt the network.

Importantly, our adversary *does not* employ a whitelist (blocking all traffic except allowed sites), seize client machines, or otherwise coerce all users into revealing which friends are running the *Unblock* software. If the adversary were to prohibit all encrypted communications or only allow connections to adversary-controlled destinations, then our approach would not be effective.

While *Unblock* is neither sufficient for users in some countries, nor guarantee anonymity, it does provide resilient access to the vast majority of blocked material. Most blocked content is not explicitly illegal or forbidden at all, but rather the collateral damage resulting from imprecise blocking of forbidden content. For example in China, when a user accesses content with forbidden keywords, users are prevented from accessing the same IP address and port for a short period of time[41]. While censors have routinely targeted and blocked major anti-censorship systems, there has been no reported incidence of users being punished for using anti-blocking software[39].

3.2 System Overview

We proceed in three steps to build a blocking-resistant overlay that can be used for interactive web browsing.

- *Unblock* is based on a social-network based overlay wherein users who have real-world trust relationships establish a communication link between their corresponding nodes and use it to convey overlay traffic. We use a social overlay because it is easier to keep participation largely secret – individual members might be compromised by social engineering attacks, but it is harder to systematically expose and block a significant fraction of overlay communications. This fact also implies that the trust relationships used to form overlay links in this setting are more restrictive than those in a typical online social networks such as Facebook. In particular, users establish relationships only with those they trust to not collude with a censor and reveal their identities. We still need mechanisms for rendezvous and routing – that is, we need a way to discover the IP addresses of friends, to discover paths to exit nodes, to figure out which exit nodes are willing to do what services, etc. A key challenge is that these mechanisms need to be resistant to blocking. (See Sections 3.4 and 3.5.)
- To improve availability and performance of multi-hop communication, *Unblock* augments the social overlay with additional random links that provide shortcuts and a greater diversity of paths. Crucially, this mechanism reveals only a bounded amount of membership information to an attacker. (See Section 3.3.)
- The augmented overlay path is still subject to transport inefficiencies that afflict overlay mix networks. To address the performance issues, *Unblock* includes transport layer mechanisms to achieve reasonable latency and throughput. In particular, it transmits across multiple overlay paths, performs datagram based transport at each overlay hop and end-to-end congestion control across the entire overlay path, and employs a back-pressure based mechanism to reduce queueing and packet loss. (See Section 3.7.)

In the remainder of this section, we elaborate on these mechanisms and also discuss their robustness in the face of adversarial attacks.

3.3 Hybrid Overlay Network

As we discussed in Section 2, relying on social network links is insufficient because of sparse connectivity and churn. To supplement connectivity, we use a hybrid overlay: we add links to approximate a *random overlay network*. The augmented network provides users with a few additional peers that are likely located at random points in the social network. In addition to providing users with redundant connectivity to the overlay, these additional *untrusted links* provide shortcuts to remote locations in the overlay.

Before we describe the augmentation mechanism used in our system, we examine some of the desired properties of such a mechanism.

- Any augmentation mechanism that exposes locations of relays may be abused by an adversary. A key challenge then is to minimize the extent to which an infiltrating adversary can exploit this mechanism to identify and subsequently block relay nodes.
- We require the mechanism to operate in a distributed setting, without requiring a centralized coordinator (which could be blocked) and without allowing for systematic exposure of social network connections or the identities of overlay nodes.
- Additional links should be to nodes that are not in a node's immediate neighborhood; this reduces path length to far away exit nodes and also increases robustness to correlated churn.
- We require the set of *untrusted links* to be stable over time in order to enable precomputation of paths to exit nodes and also to reduce the leakage of node identities to adversaries.

Our approach is to provide each overlay node with a set of untrusted links based on its position in the social overlay. We view a node's social network connectivity as a unique *capability* and develop a distributed mechanism for providing each node with additional links based on its location. This mechanism is an extension of the random walks used for sybil detection[60]. When adversaries infiltrate the social overlay, our mechanism bounds the number of nodes exposed to moles to be proportional to the number of *attack edges* controlled by the adversary, where an attack edge is a social link between a normal user and a mole who has infiltrated the system. Importantly, our mechanism ensures that moles would not be able to accumulate an arbitrary number of untrusted links by mounting a Sybil attack wherein each mole assumes multiple identities behind a single attack edge.

Preliminaries: The protocol adds untrusted links by percolating collections of randomly sampled overlay nodes, referred to as *random node lists* (or RNLs). Each over-

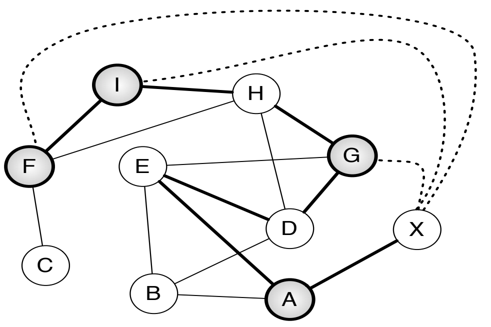

Figure 6: Example of the addition of untrusted links. In this example, an *RNL* is propagated through the path *F-I-H-G-D-E-A-X*. Nodes *F*, *I*, *G*, and *A* add themselves to the propagated *RNL*. Node *X* can then establish direct untrusted links with nodes *F*, *I*, *G*, and *A* when it receives the *RNL*. Both trusted and untrusted links can be used for data transfers.

lay node is identified by its public key and the IP address and port at which it can be contacted. The *RNL* is an ordered list of these identifiers, with the last element being the node that has been most recently added to the list. *RNLs* are propagated through the edges of the social overlay (also referred to as *trusted links*). Nodes probabilistically add themselves to *RNLs* before propagating them further. A node receiving an *RNL* can then establish *untrusted links* to nodes identified by the *RNL*. New shortcut connections to these nodes are labeled as untrusted and are not used for propagating *RNLs*; these shortcuts are used exclusively for routing overlay traffic.

RNL Propagation: *RNLs* are not flooded but rather propagated through specific paths within the trusted social network. An *epoch* is just defined to be "a long time" – a period of time where the majority of users in the system have changed their IP addresses and therefore the identity of nodes discovered in previous epochs is of little value to an adversary. This period can be on the order of a few days to weeks, depending on the underlying network [58]. At the start of the epoch, each node will take a snapshot of its current trusted links, hash the identity of each neighbor with a local secret, and use these as the set of IDs in a consistent hashing keyspace [26]. The resulting keyspace is used to determine where to forward incoming *RNLs*. The outgoing link is chosen as the link preceding the incoming link in the keyspace. The keyspace is fixed for the duration of an epoch. The use of consistent hashing implies that *RNLs* provide the same deterministic set of untrusted links during an epoch, and tries to minimize changes between epochs when new trusted links are added to the social overlay.

Upon receiving an *RNL* through a trusted link l_i, node x does the following:

1. If x has less than k friends, it will consider connecting to each item in the *RNL* with probability

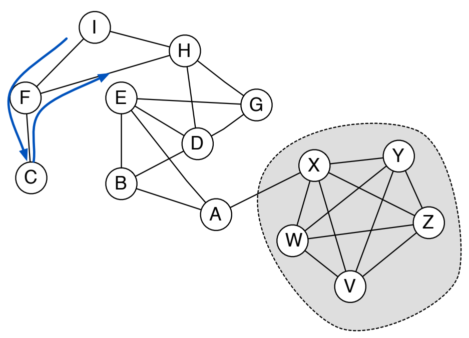

Figure 7: Example of RNL propagation. Node F has hashed its neighbors as $I \to C \to H$. Advertisements from I are passed on to C. Since C has no other friends, it will send the same list (possibly adding itself) back to F.

$(k - l)/k * l$. In the epoch, x will see l lists of k items, and attempt to connect to $k - l$ of them in order to maintain k total links.

2. x adds itself to the incoming *RNL* with probability p. This random choice is seeded using its own node ID and l_i; thus a node might add itself to an *RNL* received through some of its trusted links, but not others.[4] This will be stable across epochs.
3. Upon adding itself to the end of the incoming *RNL*, x ensures that the size of the *RNL* is bounded by the parameter k. If it exceeds k, x removes the very first element, or the farthest node, from the *RNL*.
4. x then propagates the *RNL* through one of its trusted links, choosing the one, l_o, that precedes l_i in the consistent hashing keyspace.

Figure 6 provides an example of how *RNLs* are constructed and propagated through the trusted links.

We add a few refinements to the scheme described above. First, high degree nodes forward *RNLs* but never add themselves ($p = 0$). This avoids hotspots, prevents discovery and blocking of high-degree nodes, and allows for the propagation of nodes with lower degrees. Second, any node can choose a security policy of never adding its identity to *RNLs*. This security setting allows users in regions that practice strong censorship prioritize anonymity over availability and performance. Finally, *RNL* propagation is only possible when a node and its neighbor are both online. If the neighbor is offline, we store *RNL* updates in encrypted per-link mailboxes in a DHT. When the neighboring nodes come online, they retrieve these queued messages from the DHT, forwarding them as appropriate. (We discuss the design of the DHT in 3.5.)

The *RNL* propagation mechanism limits an adversary's ability to discover relay addresses:

- Local random decisions are seeded using only the local node ID and the incoming link ID. Thus repeated invocations of the *RNL* propagation mechanism do not reveal any additional relays than earlier *RNL* messages from the same epoch.
- A mole with a single attack edge will discover the identity of at most $2k$ participating nodes in the system. In Figure 7, the mole X would only receive the k items forwarded by A, and could receive up to k additional connections upon forwarding a malicious list of items back to A. The attacker would receive no additional information from the presence of Sybils w, v, y, z. Thus, any censor's ability to find participants will be limited by the number of moles and the edges it controls.

Parameter settings: *Unblock* is parameterized by p and k. We set these parameters based on the estimated size of the network (n) and the expected probability that a typical node is online in the system (f). If we set p to *1/log(n)*, then any two nodes propagated through an *RNL* messages are likely separated by *log(n)* hops in the social network. It has been shown that many social networks have the small-world topology property in that a sequence of *log(n)* random hops through the network often leads to a random node within the network [9, 20, 60].[5] Further, if we set k to c/f, where c is a small constant, then c of the hops provided by an *RNL* message are likely to be active at any instant in time, given that the uptimes of random nodes are likely to be uncorrelated. This allows any node in the system to have access to c additional, randomly distributed relay nodes.

It is also worth noting that the above mechanism has other desirable properties. First, high-degree nodes critical for network connectivity are protected from being revealed, as these nodes do not append themselves to *RNL* messages, thus allowing for improved path diversity to low-degree nodes. Second, *RNL* messages propagate k relays across each trusted link that separates a censored domain from other uncensored domains. Thus, the augmentation mechanism will likely increase the number of overlay connections across ISP or state boundaries by a factor of k.

In summary, the described mechanism results in a hybrid overlay that augments the underlying social network with additional links that approximate a random overlay network. Crucially, the mechanism maps the location of a node in the social network to a set of random nodes in a consistent and crawl-resistant manner, thus limiting the leakage of relay identities and safeguarding against Sybil attacks.

[4]This allows for many nodes to be propagated over some paths as opposed to propagating a few nodes over many paths, which would result in hotspots in the augmented overlay.

[5]The augmented overlay will have increased availability even if nodes in *RNLs* are not randomly sampled nodes. The benefit of approximating a random sampling is that it bounds the diameter of the augmented overlay [7].

3.4 Accessing the Internet through exit nodes

Unblock uses exit nodes to create a bridge between the overlay network and the public Internet. Exit nodes can either provide global Internet connectivity or restrict access to a small set of services. The user running the exit node is free to specify the exit policy of their node. Running an exit-node with global connectivity has been problematic for other networks[6], motivating the more expressive policy. In *Unblock*, a user can opt to only provide access to certain domains (e.g., only access to wikipedia.org, twitter.com, google.com, and nothing else), to reduce the risk of abuse. Service providers that wish to improve access to their own site can run their own nodes providing access to only their service. In this latter case, the provider may keep information about the exit node (such as its IP address) private, creating an effect similar to Tor hidden services.

In order to contact an exit node, a peer must first know of its existence. In *Unblock*, each exit node is identified by a public key, along with a recent announcement potentially signed by the *Unblock directory service*.[7] The announcement asserts which services are accessible from the node, and where the node is located (e.g., country or ISP domain).

Exit nodes announce their presence periodically through announcement messages. When nodes receive an announcement, they *immediately* forward the announcement to their neighbors. The return paths of these announcements determine a minimum latency routing tree that is used when communicating to the exit node. Announcements contain a timestamp, a nonce, the hash of the public key of the exit node, an optional set of exit node properties (such as the region where the node is located and domains reachable through the node), and an optional signed attestation from the directory service that the announcement is indeed from the exit node.

Given the augmented overlay structure of *Unblock*, there are often a large number of possible paths to choose from when routing to an exit node. The goal of the routing protocol is to provide: (a) minimized end-to-end latency, (b) multiple parallel paths when available, and (c) resilience to node failure/churn.

Our approach uses announcement paths to connect to exit nodes. Announcements create a minimum spanning tree, but the tree will often be invalid due to churn. To keep the tree current, nodes update their neighbors with their new latency when their minimum latency link disconnects or reconnects. To find multiple paths to an exit, clients route through multiple links for the first hop and then traverse the spanning tree from then on.

Signed announcements imply that the signer has verified the claims in the announcement, helping to validate client trust. Exit nodes which cannot obtain a signature from the directory server may participate in the system either by offering a local service, which clients must explicitly connect to, or by offering their own directory server, and encouraging clients to trust that authority.

[6]For example, law enforcement sometimes misattributes traffic from a Tor exit node to the owner of the node.

[7]The purpose of the directory system is described in 3.6. The directory service is replicated, e.g., on PlanetLab nodes in our current deployment, to ensure higher availability and reachability, and can be accessed directly or through the *Unblock* overlay.

3.5 Internal Overlay DHT

Unblock includes a DHT for two purposes: (a) as a mailbox for communicating *RNL* messages to offline peers and (b) as a rendezvous service for locating the current IP address of peers when a node rejoins the overlay[25]. We cannot rely on an external DHT, such as OpenDHT or Vuze's DHT, as access to these can be blocked. The system therefore has to provide a DHT-like functionality using just the nodes participating in the overlay. An additional restriction is that the DHT implementation should not expose the identities of nodes participating in the system; that is, DHT operations should be performed using just local information comprising of the trusted or untrusted links known to the participants. Another requirement is that the DHT needs to be robust to byzantine moles operating as DHT service nodes.

The DHT is created by partitioning keyspace across the exit nodes that participate in the system. Both objects and exit node identifiers are hashed onto a circular keyspace, and objects are assigned to the exit node that is closest to it in the keyspace. To perform lookups and updates, we leverage the fact that the exit node announcements create a routing table at each node in the system. This table contains the next hop to route to each exit node. When a node wishes to query the DHT it can first look at its local routing table to find the exit node most adjacent in key-space to the desired key. The node then routes the query to the exit node through the appropriate neighbor, and nodes along the way maintain state in order to route the reply. There is no guarantee that the querying node knows about all exit nodes,[8] so each hop along the path computes the closest known exit node to the target key and routes the query towards it. DHT lookups can be routed through both trusted and untrusted links, increasing resilience.

This scheme is essentially a variant of *Virtual Ring Routing* [10], where nodes are able to provide a DHT-like abstraction by routing messages through their neighbors in a physical network. Our approach adds an additional restriction in that the DHT storage is hosted on just the signed exit nodes (as opposed to all overlay nodes) in order to improve both security and performance. First, if all nodes are allowed to serve as DHT storage nodes, then

[8]This can happen immediately after startup for example.

an adversary can mount Sybil attacks and lower DHT availability [55, 50]. Second, since most nodes would have received validated exit node announcements, they can directly route towards the appropriate storage node without requiring recursive lookups as in [10].

The DHT is used to perform rendezvous for nodes rejoining the system. In order to reconnect, nodes use the system-internal DHT to update their current address to friends. All that is required for reconnection is that at least one previous connection remains at the address it was last seen.

3.6 Overlay Security: Attacks and Defenses

There are two key considerations in the design of the overlay routing and transport protocol used in *Unblock*. The most important is defense, which prompts the use of secure DHT access, per-hop and end-to-end encryption, and of mixing messages from different connections into packets to protect users from an adversarial network. The second goal for the protocol is performance, which motivates the use of a custom application-level wire format, multi-path support, and UDP datagrams rather than TCP connections. We discuss these two separately, first laying out the security properties of *Unblock* communications, and then sketching how we gain performance within those constraints in 3.7.

The *Unblock* protocol encompasses several stages: new friend establishment, rendezvous, connection establishment, choice of exit node, and data transfer. Below, we discuss how *Unblock* prevents the exposure of node identities to an adversary and limits his ability to disrupt overlay communication for each stage.

Friend Establishment: Friend connections are established using existing mechanisms from the OneSwarm network. Willingness to relay arbitrary bulk data between friends is very similar to willingness to relay encrypted web traffic between friends. OneSwarm allows users to add untrusted links from bulletin boards to fill out the social network graph; we do not use those links, as our shortcut links provide the same goal with less risk of disclosure.

Rendezvous: *Unblock* uses the DHT to store IP address information needed for rendezvous. The stored information is encrypted to ensure that the DHT cannot be crawled to determine the IP addresses of the nodes participating in the overlay. Specifically, each node is identified by a 1024 bit RSA key pair. This key is persistent, even if the IP address of the peer changes. At startup, a node will insert a copy of its current connection information (IP address, port number) into the internal DHT for each of its direct links. These copies will be encrypted with the public key of the neighbor, and indexed into the DHT using a 20 byte, randomly generated shared secret, agreed upon during the first successful connection. This ensures the secrecy of both the key and its contents.

Connection Establishment: *Unblock* connections between neighbors use SSL based on the nodes' RSA key pairs. This prevents an adversary from knowing what service is offered, probing the connection to identify whether a given node is running *Unblock*[9], or distinguish it from other SSL connections. Control messages, such as DHT searches and exit node announcements occur directly within this connection. As part of connection establishment nodes also detect if they can transmit UDP packets to each other, and data transfer will occur through encrypted UDP packets when possible. We do not believe the presence of both TCP and UDP connections between nodes is enough to fingerprint *Unblock* because both connection types are also used by a range of remote desktop and real time chat protocols.

Exit Node: An important property of the *Unblock* protocol is resilience to adversaries claiming to offer exit capabilities. The two mechanisms a malicious exit node can leverage to directly attack the system are: (1) flooding announcements to overwhelm the system, and (2) blackholing received traffic. We mitigate these attacks through certification. Nodes in the system will only forward exit node announcements if they are signed by a trusted directory service. This property allows the trusted service to throttle the total rate of exit node announcements on the network and to verify the functionality of exit nodes before signing proposed announcements. Exit nodes will periodically request certification from the directory service through the *Unblock* overlay. In order to limit sybil attacks on the directory server, we require computational puzzles to be solved as part of the certification request.

The use of a directory server does open additional channels of attack that we must now address. First, if an adversary controls a node on an announcement path, it can selectively forward only the announcements for exit nodes it is colluding with. This attack is mitigated by the presence of shortcut links, which allow nodes exposure to additional announcement paths via random nodes in the overlay and forces an adversary to fully partition the network for an effective attack. Secondly, an adversary may attempt to directly attack the directory server, through a denial of service attack. The service can be built to withstand such attacks, since it can run across multiple machines and addresses to increase availability. If the service is successfully taken offline, the only negative effect is that advertisement trees may become stale.

Data Transfer: Once channel establishment has occurred, the end-to-end data stream is encrypted with AES based on the shared secret negotiated during connection establishment. This prevents a malicious relay from seeing the contents of the communication. The data stream

[9]This probing technique is used by China to identify secret Tor bridges.

is packetized, and packets can be sent over any available path to the destination, which means a relay cannot know if they are seeing the entire conversation.

Packets themselves are sent using per-hop DTLS, so different data will be seen entering and leaving each node. The use of DTLS also makes *Unblock* data indistinguishable from VPN and VOIP applications which use the same packaging. Nodes also pack each datagram full, by appending small packet acknowledgement messages, and then padding the message to the MTU size in order to frustrate message size correlation.

3.7 Overlay Performance

A usable system needs to provide an acceptable level of performance for typical interactive browsing. We made several protocol decisions to minimize the inefficiencies of overlay transport. These include: the use of UDP datagrams with custom flow control, the ability to take advantages of multiple paths through the overlay, and a custom application-level protocol for web requests.

Datagram Flow Control: Path conditions can change due to churn or temporary bursts of traffic. Tor multiplexes traffic between nodes, with multiple independent flows multiplexed onto a single reliable TCP connection between adjacent relays. When these flows have different characteristics, the multiplexing can result in suboptimal performance for all flows traversing the link.[10] The most immediate issue is that small, latency sensitive flows can get "stuck" behind larger bulk data transfers. To address this issue we use a datagram based transport at each overlay hop and end-to-end congestion control across the entire overlay path. This minimizes the interference between flows that share the same overlay hop.

Nodes in the system can have very different upload capabilities, which will result in queuing. Flows originating at a high bandwidth node will quickly fill the buffers of subsequent low bandwidth relays. Aggravating this issue, overlay paths span multiple hops, often spanning several continents. End-to-end congestion control responds to congestion over timescales of RTT, leading to slow ramp up and slow recovery from loss. We address these issues by adding explicit per-hop flow control, where nodes communicate how much they are willing to buffer for each active connection.

This mechanism minimizes queueing and eliminates packet loss on overlay nodes by regulating the flow of data from upstream nodes using credits. Credit to send data to a downstream node is replenished through control messages. When a node detects that a queue is building up, it stops issuing credits to upstream nodes, thus temporarily slowing or stopping the flow. This design is similar to mechanisms used in ATM networks [29], which suggest that some queue must be allowed to form in order to fully utilize the bottleneck node [49].

[10]Prior studies have diagnosed these issues in the context of Tor and proposed backwards-compatible fixes to Tor, while retaining the basic per-hop TCP transport and single path transfers [18, 2, 42].

Nodes in *Unblock* therefore detect if they are a bottleneck, and manage their credits accordingly. Nodes can detect that they are non-bottleneck nodes when they are limited by credits rather than their own bandwidth. This allows us to fully use available throughput while minimizing latency at intermediate hops.

End-to-end Congestion Control over Multiple Paths: The routing algorithm ideally yields multiple paths to a specific exit node. Data from the incoming stream is split into chunks, which are then transmitted across all available paths using UDP datagrams. The receiving endpoint assembles the packets and delivers it to the application in the correct order. *Unblock* handles congestion over end-to-end paths using a TCP style transfer window for each overlay path that is updated using the traditional additive increase multiplicative decrease mechanism upon packet losses over that path (as in MPTCP [56]).

We also use a redundancy mechanism to balance the goals of latency and throughput. Based on how much data has been transmitted, the sender will determine if the stream is a data-intensive, throughput-bound stream, or a bursty, latency bound stream. Initially, all transfers are assumed to be latency sensitive and messages will be duplicated and sent along multiple overlay paths. The amount of duplication is steadily reduced as more bytes are transferred over the end-to-end path. This balances the goal of minimizing latency when transmitting small pieces of content with the goal of using all of the available throughput for larger transfers.

Application Level Protocol: While our transport supports tunneling arbitrary TCP connections, we also provide an optimized protocol for web requests, using a local SOCKS proxy on the client and a custom server on the exit node. This addition is preferable to running a SOCKS proxy on the exit node directly, because the SOCKS protocol is chatty, and several round trips are spent between the client and the proxy at the beginning of every connection. By resolving that locally, we can send web requests along with the destination information in the first data packet sent to the exit node. Unlike a normal SOCKS proxy, the previous announcement from the exit node has informed the client which domains are supported by it, nor do we need authentication at the SOCKS layer.

3.8 Deployment

Unblock leverages resources provided by the participants in the system in order to provide a self-scaling network, in contrast to other systems which pay to operate a number of proxies[54, 21] or those systems where there is a distinction between users of the system and relays that constitute the transport infrastructure (e.g., Tor).

Unblock is currently built as an extension to OneSwarm, which is also a social-network based overlay that enables privacy preserving peer-to-peer file sharing [25]. Bundling *Unblock* with another application addresses some of the challenges associated with deployment and incentives. First, it allows us to the develop and test our system under real-world conditions by enabling experimental extensions on nodes already conveying OneSwarm traffic. Second, since OneSwarm is actively used by thousands of users daily, initial users of *Unblock* can benefit from a resource-rich network comprising of users in countries with little or no censorship.

We do however have to ensure that the incentives for the two types of users are reasonably aligned. For instance, there should be benefits for OneSwarm users to serve as relays for *Unblock* traffic, considering that *Unblock* users would not necessarily participate in file sharing. Interestingly, by establishing trusted or untrusted links with *Unblock* users, OneSwarm users can have larger degrees of connectivity, which in turn translates to more diversity of overlay paths between OneSwarm users. In other words, it suffices that *Unblock* users are passive transport relays as opposed to active file sharers.

4 Evaluation

In this section, we present experiments that evaluate *Unblock*. Currently, the *Unblock* extension has been enabled by a small fraction of OneSwarm users who have opted in experimental updates, and we do not have access to the topology of the user base, so it is difficult to quantify the performance and robustness of *Unblock* using the deployment. We therefore evaluate *Unblock* using simulations and controlled wide-area testbed experiments.

We use a simulator to evaluate the security and performance properties of *Unblock* at scale. We measure the impact of using an augmented overlay. Shortcut links are shown to maintain connectivity for more nodes for typical uptimes seen in peer-to-peer systems. We then explore the trade-off between better availability and risk of disruption of service by a censor adversary.

Next we evaluate the performance of our transport layer implementation using a multi-hop test framework in PlanetLab. We examine the individual mechanisms that comprise the transport layer used in *Unblock* and also compare its performance against standard transport mechanisms used in systems such as Tor.

4.1 Simulation Results

Using the simulator, we find that the shortcut discovery protocol effectively improves the connectivity to any particular exit node in the face of churn, while restricting the number of honest users that are exposed to a censor's *moles* in an attack. Even with a strong model of an adversary that can block all edges of the exposed nodes in the network, shortcuts effectively improve connectivity.

We perform these measurements using simulated networks based on the datasets collected by [35, 61]. For some of these datasets, as in the Youtube social network, we were able to obtain the geographical location of the user. In such cases, we attribute a latency between users using predictions from *iPlane* [34]. Exit nodes and moles are chosen at random from available nodes in the graph. We performed our evaluation for varying churn, wherein the node uptimes and downtimes are modeled using Poisson distributions. Lastly, shortcuts are only created between nodes that have degree less than 50. This restriction protects high-degree nodes from being overloaded and restricts disclosure of high-value nodes to censors.

Figure 8(a) shows the improvement in the availability of paths to exit nodes as we augment the underlying social network for the Youtube dataset with additional untrusted links. In this experiment, we set 10% of the nodes to be exit nodes. We perform our experiment for a range of node uptime values. For each value of expected node uptime fraction f, we set the number of untrusted links discovered by the *RNL* mechanism to be $2/f$. This parameter setting implies that each active node, in expectation, will have two untrusted links to other active nodes in the system. The results show that the augmented social overlay provides dramatically higher availability of paths to exit nodes, especially when the node uptime fraction is low (as is the case with most peer-to-peer systems [51, 22, 44, 31]).

The Youtube social network comprises about a million users. We performed the analysis described above on both smaller and larger social networks (e.g., the Foursquare network with about hundred thousand users and the Flickr network with about two million users) and obtained results that were qualitatively similar. For example, addition of untrusted links improved availability from 39% to 97% for the Flickr social network and from 59% to 99% for the Foursquare social network under the assumption that the fraction of node uptime is 0.2.

We also examined the improvement in latency of the path to an exit node using the Youtube dataset. Figure 8(b) shows the CDF of latencies to any available exit node when nodes are online for 50% of the time. We examine this with and without untrusted links, and observe that the use of untrusted links also significantly lowers latency. We also model a strong adversary that monitors exposed shortcut nodes from 10000 moles in the network. The censor also has the power to block all of a node's links if exposed to its moles.[11] As expected, we found a linear relationship between the number of attack edges and the number of honest nodes exposed to moles.

[11] In practice, a censor would be able to block communications to the relay only from those nodes within the censored domain.

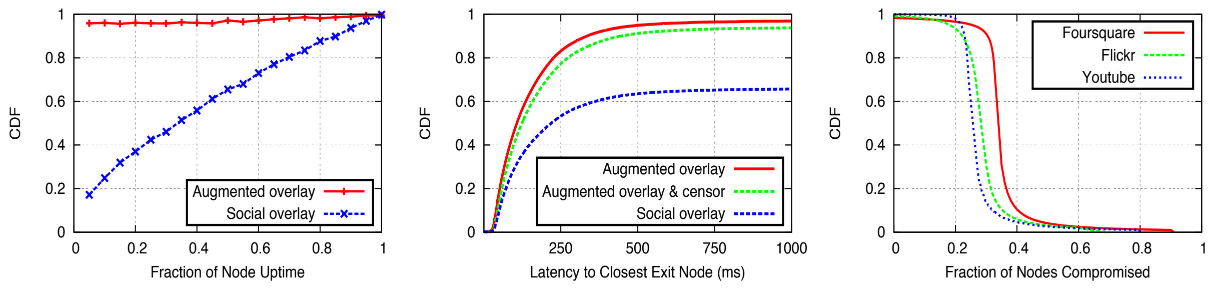

Figure 8: (a) Fraction of nodes with paths to exit nodes on the Youtube social network dataset for varying node uptimes and with 10% of the nodes being exit nodes. (b) Impact of untrusted links on latency to exit nodes when 50% of users are online. (c) Fraction of nodes with paths to exit nodes under adversarial attacks on availability.

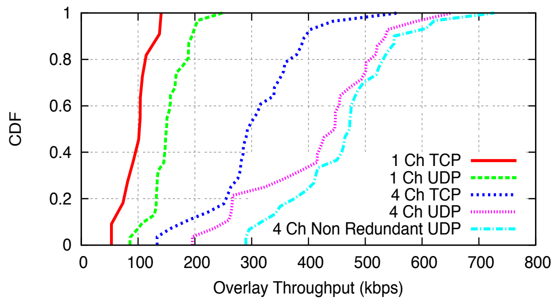

Figure 9: Throughput performance of *Unblock*. UDP performance improves with more paths until the endpoints are bandwidth limited. Non Redundant represents throughput when packets are only sent once, at the cost of latency.

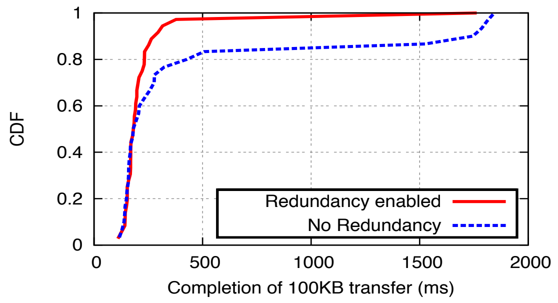

Figure 10: Latency performance within the overlay. With redundancy, latency suffers less from slow/flaky paths.

More importantly, Figure 8(b) shows that there is minimal degradation in both connectivity and performance as a consequence of having the strong adversary.

Finally, we examined the impact of various types of disruption attacks by adversarial nodes. We modeled an adversary who had compromised a fraction of the nodes in the social overlay and has the ability to drop protocol messages and disrupt transport channels by dropping packets. In particular, we considered an adversary who dropped RNL messages, forwarded exit node announcements, and then dropped the data packets of an overlay flow. Note that it is more effective for the adversary to forward exit node announcements so as to position itself on more overlay transport paths. Figure 8(c) shows the fraction of nodes with working paths to exit nodes as we vary the fraction of live nodes that are compromised. We find that connectivity in the augmented overlay is adversely impacted only in the case of a determined adversary who has compromised more than 20% of the nodes.

4.2 Transport Performance

We next consider microbenchmarks that allow us to examine the performance and latency enhancements made possible by different versions of the transport layer. Performance was evaluated using PlanetLab nodes located across the US. In all trials, the topology consisted of four disjoint paths from client to server, each with three hops. All nodes were selected randomly from the available pool, with nodes reselected between each trial. We also imposed bandwidth rate limits of 1Mbps at each node. In Figure 9, we present observed throughput achieved with the various transport improvements: Transferring data using an encrypted UDP transport, transferring data concurrently over multiple paths, and dynamic use of redundant packet transmissions. Throughput is measured as the time required to transmit one megabyte of data. Using multiple paths with UDP improves throughput linearly until three paths, where bandwidth of either the source or destination limits its ability to transmit or receive more. We also examine the throughput of multipath flows that do not perform any redundant transmissions in order to characterize the capacity lost due to redundancy; this scheme provides only a marginal increase in throughput indicating that the cost of redundant transmissions is low.

Figure 10 provides microbenchmark results that evaluate the use of redundant transmissions. We measure the transmission time for a 100 kilobyte flow across the same topology as the other experiments, with and without the adaptive use of redundant transmissions. While most links in our testbed had robust performance characteristics, when slow or flakey links were encountered, redundant transmissions were able to maintain a low latency connection by mitigating retransmissions and in-order delivery delays.

We conclude with an evaluation of web page load per-

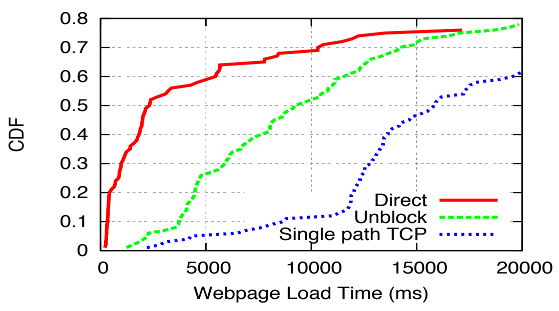

Figure 11: Web page load time across the *Unblock* overlay. Unblock represents load times across a three hop overlay using the optimized *Unblock* transport protocol. Single path TCP shows baseline load times for the same topology using per-hop TCP over a single overlay path.

formance through the overlay. We evaluated the performance of our transport by inserting the *Unblock* overlay as a relay to a SOCKS proxy. The PhantomJS headless webkit browser was used to measure page load times of popular websites. Much of the time spent rendering a page comes from dependent resources, making network latency more important than many systems admit. Pages were loaded from the same set of domains as Figure 5.

This set of experiments demonstrates the huge importance of lowering latency in order to efficiently handle the small, bursty traffic associated with web requests. Figure 11 shows that *Unblock* has a fairly constant 2-5 second page load penalty compared with loading pages directly. The use of UDP, the ability to take advantage of multiple channels, and the credit-based flow control already provides a significantly less variable and lower latency service than the baseline transport that uses per-hop TCP connections over a single overlay path. We are continuing to improve performance as we complete a public release. The wider deployment will also let us see reactions to censorship, and potentially gain greater understanding of the mechanism used by censors world wide. We are already recording both live performance of *Unblock* and censorship measurements from around the world at `unblock.cs.washington.edu`.

5 Related Work

Providing privacy and anonymity for Internet data transfers is a longstanding goal of the research community, and we draw on many existing ideas in our design.

Anonymous Communications: A basic approach to achieving anonymous data transfers is to interpose a third party (or proxy) between the source and destination to hide the source's identity from the destination [3]. A proxy, however, allows a single entity to learn the identities of both communicating parties. This lead to the development of schemes that convey traffic through multiple intermediaries. Crowds [43] provides anonymity by having an intermediary either choose a random successor or simply transmit to the destination. Tor [17] leaves the choice of relays to the source.

Censorship resistance: Naturally, anonymizing solutions have been adapted to achieve censorship resistance [40, 13]. A key stumbling block is that most proxy-based anonymizing solutions aren't membership concealing. The Tor developers recognized this challenge [53] and use semi-secret bridges, but they can be exposed with some effort (as we show in Section 2). Recently, there has been a proposal for rearchitecting the Internet to provide Tor like functionality at the network level [33]. Also related to such an approach is Telex [57], which utilizes steganography over random bits within an HTTPS header to hide a tag that a middlebox can later detect and use to reroute the packet. Both approaches require pervasive changes at the network level and face significant deployment challenges.

Sybil defenses: Many defenses have been proposed to combat Sybil attacks. These include strong identities minted by a logically centralized authority [19], computational puzzles and bandwidth contributions to make peers prove that they are not Sybils [8], and leveraging social networks [60, 30]. Defenses based on social networks, such as SybilGuard [60, 59], might seem appropriate for our setting as they limit the creation of trust relations to unknown identities based on the social network properties of the requesting nodes. They are however insufficient for the threat model we consider partly because they do not provide any mechanisms for concealing the membership of the social network and partly because they provide weak bounds on the number of trust links created to Sybils by the network as a whole.

6 Conclusion

The desire for uncensored Internet access has motivated the development of systems for censorship circumvention. However, most popular systems are easily blocked and often offer poor performance. In this paper, we presented the design and implementation of *Unblock*, a blocking-resistant overlay network that can reroute Internet traffic to avoid censorship. *Unblock* is designed to combine the security, privacy, and locality properties of routing over social networks with the more robust connectivity properties of open access overlays. It is designed to be resilient to various blocking and infiltration attacks and provides high performance transport over multi-hop overlays. Through large scale simulations of the system and measurements of a prototype implementation deployed on PlanetLab, we show that *Unblock* can provide high availability and improved performance. We believe the ideas behind *Unblock* will allow it to improve upon both the privacy and performance of existing proxy-based censorship circumvention systems.

References

[1] Alexa top 500 global sites, 2012. http://www.alexa.com/topsites.

[2] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker. Defenestrator: Throwing out windows in Tor. In *PETS*, 2011.

[3] Anonymizer. http://anonymizer.com.

[4] Anonymous. The collateral damage of internet censorship by dns injection. In *SIGCOMM CCR*, 2012.

[5] J. Appelbaum. Tor bridge nodes, 2011. Private communication.

[6] R. Atkinson, S. Ezell, S. Andes, D. Castro, and R. Bennett. The Internet Economy 25 Years After .com. *The Information Technology and Innovation Foundation*, 2010.

[7] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.

[8] N. Borisov. Computational puzzles as Sybil defenses. In *P2P*, 2006.

[9] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: design, analysis and applications. In *INFOCOM*, 2005.

[10] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: network routing inspired by dhts. In *SIGCOMM*, 2006.

[11] S. Chen. China tightens Internet censorship controls. http://www.bbc.co.uk/news/world-asia-pacific-13281200, May 2011.

[12] Beijing orders new controls on 'weibo' microblogs. http://www.bbc.co.uk/news/world-asia-china-16212578, Dec. 2011.

[13] Citizens Lab. Everyones guide to bypassing Internet censorship. http://deibert.citizenlab.org/Circ_guide.pdf.

[14] Congressional executive commission on China, annual report. http://www.cecc.gov/pages/annualRpt/annualRpt11/AR2011final.pdf, 2011.

[15] G. Danezis, C. Diaz, C. Troncosco, and B. Laurie. Drac: An architecture for anonymous low-volume communications. In *PETS*, 2010.

[16] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *USENIX Sec.*, 2004.

[17] R. Dingledine and S. Murdoch. Why Tor is slow and what we're going to do about it. http://blog.torproject.org, 2009.

[18] J. R. Douceur. The Sybil Attack. In *P2P*, 2002.

[19] A. D. Flaxman. Algorithms and models for the webgraph. In W. Aiello, A. Broder, J. Janssen, and E. Milios, editors, *Expansion and Lack Thereof in Randomly Perturbed Graphs*. Springer-Verlag, 2008.

[20] Freegate. https://simurghesabz.net/.

[21] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP*, 2003.

[22] How to bypass internet censorship. https://www.howtobypassinternetcensorship.org/.

[23] P. Howard, A. Duffy, D. Freelon, M. Hussain, W. Mari, and M. Mazaid. Opening closed regimes: What was the role of social media during the arab spring? http://pitpi.org/?p=1051, 2011.

[24] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson. Privacy-Preserving P2P Data Sharing with OneSwarm. In *SIGCOMM*, 2010.

[25] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *STOC*, 1997.

[26] J. Karlin, D. Ellard, A. Jackson, C. Jones, G. Lauer, D. Mankins, and W. Strayer. Decoy routing: Toward unblockable internet communication. In *USENIX FOCI*, 2011.

[27] S. Kelly and S. Cook. Freedom on the net 2011: A global assessment of Internet and digital media. *Freedom House*, 2011.

[28] H. T. Kung, T. Blackwell, and A. Chapman. Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation and statistical multiplexing. In *SIGCOMM*, 1994.

[29] C. Lesniewski-Lass and M. F. Kaashoek. Whanaungatanga: Sybil-proof distributed hash table. In *NSDI*, 2010.

[30] J. Li, J. Stribling, R. Morris, F. Kaashoek, and T. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. *IEEE Infocom*, 2005.

[31] G. Liang. Surveying internet usage and its impact in seven chinese cities. Technical report, Chinese Academy of Social Sciences, 2007.

[32] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson. Tor Instead of IP. In *HotNets*, Nov. 2011.

[33] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *OSDI*, 2006.

[34] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.

[35] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi. Ostra: leveraging trust to thwart unwanted communication. In *NSDI*, 2008.

[36] M. Motoyama, D. McCoy, K. Levchenko, G. M. Voelker, and S. Savage. Dirty Jobs: The Role of Freelance Labor in Web Service Abuse. In *USENIX Sec.*, 2011.

[37] Opennet initiative global Internet filtering map. http://map.opennet.net/.

[38] OpenNet-Inititative. Internet filtering country profiles: China. http://opennet.net/research/profiles/china-including-hong-kong.

[39] Psiphon. http://psiphone.civisec.org.

[40] R. Clayton and S. Murdoch and R. N. M. Watson. Ignoring the great firewall of China. In *Privacy Enhancing Technologies*, 2006.

[41] J. Reardon and I. Goldberg. Improving Tor using a TCP-over-DTLS tunnel. In *USENIX sec.*, 2009.

[42] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. Inf. Syst. Secur.*, 1998.

[43] RFC 4981. http://www.faqs.org/rfcs/rfc4981.html.

[44] H. Roberts, E. Zuckerman, J. York, R. Faris, and J. Pal-

frey. 2010 circumvention tool usage report. The Berkman Center for Internet & Society, 2010.

[45] H. Roberts, E. Zuckerman, J. York, R. Faris, and J. Palfrey. International bloggers and Internet control: Full survey results. The Berkman Center for Internet & Society, 2011.

[46] S. Scellato, C. Mascolo, M. Musolesi, and V. Latora. Distance matters: Geo-spacial metrics for online social networks. In *WOSN*, 2010.

[47] S. Sengupta. Group says it has new evidence of Cisco's misdeeds in China. *The New York Times*, Sept. 2011.

[48] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. Low extra delay background transport (LEDBAT). In *IETF Internet Draft*, 2006.

[49] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *P2P*, IPTPS '01, pages 261–269, London, UK, UK, 2002. Springer-Verlag.

[50] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *SIGCOMM*, 2006.

[51] Tor metrics portal: Performance. https://metrics.torproject.org/ performance.html.

[52] China blocking Tor. https://blog.torproject.org/blog/china-blocking-tor-round-two.

[53] Ultrasurf. http://www.ultrareach.com/.

[54] G. Urdaneta, G. Pierre, and M. van Steen. A survey of DHT security techniques. *ACM Computing Surveys*, 43(2), 2011. `http://www.globule.org/publi/SDST_acmcs2009.html`.

[55] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *NSDI*, 2011.

[56] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the Network Infrastructure. In *USENIX sec.*, 2011.

[57] Y. Xie, F. Yu, K. Archan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are IP addresses. In *SIGCOMM*, 2007.

[58] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against Sybil attacks. In *IEEE Symposium on Security and Privacy*, 2008.

[59] H. Yu, M. Kaminsky, P. B. Gibbons, and A. D. Flaxman. SybilGuard: defending against Sybil attacks via social networks. *SIGCOMM*, 2006.

[60] R. Zafarani and H. Liu. Social computing data repository at ASU. http://socialcomputing.asu.edu, 2009.