Blocking-Resistant Network Services using Unblock

Will Scott, Raymond Cheng, Arvind Krishnamurthy, Thomas Anderson
University of Washington

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 analyze appropriate transport level techniques for such an overlay, and demonstrate the practicality of the system for web traffic.

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 [22, 36]. Worse, network equipment providers have shown a willingness to commoditize and profit from censorship by selling interception and filtering devices [45].

Censorship today is more than aggressive suppression of activists by oppressive governments. Many countries use a variety of automated techniques to affect communications on a wide scale. For a variety of reasons, entire domains are regularly blocked [1], sometimes due to individual pieces of objectionable content. Domain blocking is very coarse-grained and often blocks access to unrelated content for extended amounts of time. For example, in 2008 Pakistan temporarily blocked all of YouTube due to offensive "non-Islamic" videos [58]. Other content may be blocked to shift public perception (selectively cropping content from someone's news feed can significantly alter the way they perceive the world [9, 10]) or to exert economic pressure [13] by degrading service to foreign competitors, enabling new forms of protectionism in the multi-trillion dollar Internet economy.

Like many security problems, Internet censorship is not purely technical and countries all have their own approaches to law enforcement on the Internet. We define "hard" censorship as enforcement through judicial and physical means, such as the prosecution of illegal activity. In contrast, we define "soft" censorship as the removal of content that is legal to access in the associated country. Instead of tackling the full breadth of censorship problems, we focus on the more insidious problem of soft censorship. The OpenNet Initiative reports that 37 of 74 measured countries perform technical censorship [36]. Furthermore in countries with soft censorship, there is clearly a demand for robust circumvention networks. Reports in China show anywhere from "at most 3% of the population" [42] to "25% of netizens" [27] to "58% of bloggers" [43] have used tools to access blocked content at some point. This refined scope also allows us to tailor our system towards the needs of the majority of censored netizens, trading improved performance, availability, and accessibility for full anonymity.

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 censored content. We assume that users of Unblock access legal content and face little risk if they are detected beyond Internet disruption. For example in China, when a user accesses content with forbidden keywords such as "Tiananmen Square", they may experience service disruptions for unrelated content afterwards [37]. While there may be some uncertainty regarding the legality of specific content in each country, we believe Unblock is valuable as a tool to stem the ability of controlling entities to covertly censor Internet content.

Previous research on censorship-resistant networks has focused on routing-level network designs [21, 28, 56] and overlay systems [14, 53], 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. Prior censorship-resistant overlay systems use relays that are easy to identify and block. This forces users to constantly add and configure new relays, an effort that is financially and logistically exhausting [4]. As we show in Section 2, attempts to hide relay locations are largely ineffective.

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 content is not censored. Multi-hop routing, coupled with mechanisms to prevent overlay disruption, hide participants.

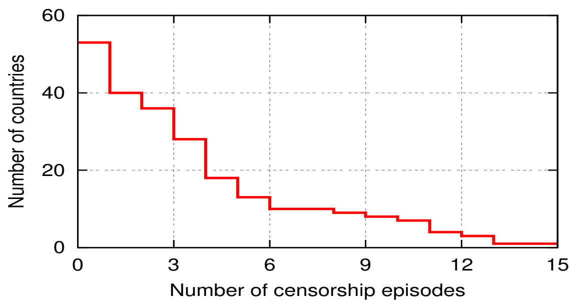

Figure 1: Number of observed censorship episodes against Tor (i.e. blocking Tor when it was previously not blocked).

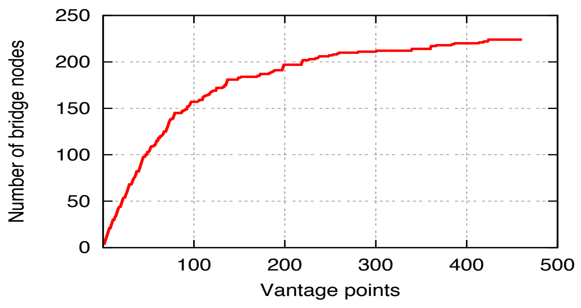

Figure 2: Number of discovered Tor bridge nodes vs number of PlanetLab vantage points.

Unfortunately, node degrees in social networks exhibit a power law distribution where many users only have a small number of friends, reducing availability. 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 increase availability. The system also employs a custom set of transport mechanisms optimized for such a multi-hop network.

In order to demonstrate the feasibility of Unblock, we implement the protocol on top of OneSwarm, an existing social overlay-based bulk file sharing system. Our implementation shows the ability of censorship resistance to piggy-back upon existing systems, and demonstrates the incentives and protocol-masking techniques such a system can employ. We evaluated the performance in controlled testbed settings and measured the ability to perform web requests under various configurations. We evaluated our mechanism for social network augmentation 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.

2 Background and Challenges

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 frustrating 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 have two characteristic attributes: (a) any client can use any relay to construct a circuit for routing traffic, (b) they rely on a centralized directory system to publish information regarding relays. These include anonymizing overlays such as Tor, Ultrasurf, and Freegate, as well as open proxies that are used to evade censorship [17].

Open access overlays are vulnerable to blocking because of their use of a centralized directory service and because they freely distribute 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 resilience of open access overlay, we analyzed availability provided by the Tor network. Using data the Tor project has maintained from usage of its network in 243 countries from August 2007 to December 2012 [48], we aggregated the number of clients that connected to each of the Tor directory servers into two week periods by country. We compared these totals 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 analyze this data by defining 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. Out of 243 countries, Tor experienced at least one censorship episode in 53, with repeated disruptions in many of those countries.

In response to an increasingly hostile network environment, Tor has added semi-secret relays called *bridge nodes* and protocol obfuscation called obfsproxy. While obfsproxy is orthogonal to IP blocking, and employs techniques which work in tandem with Unblock, bridge nodes do not solve the problem of providing safe entry points to the overlay. The same mechanisms which help users find bridges can be abused to identify and block those same machines. 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.[1] For Figure 2, we crawled the Tor

[1]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

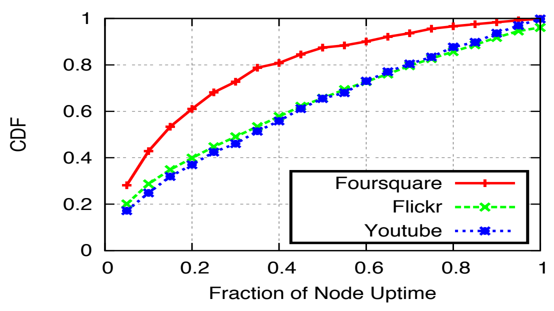

Figure 3: Fraction of nodes with paths to exit nodes on different social network datasets for varying node uptimes and with 10% of the nodes being exit nodes.

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 [3]. One could easily imagine a censor using similar crawling techniques to find and block bridge nodes. Reports confirm that China already blocks bridge nodes [50].

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 the Ostra [33] email service and the OneSwarm [20] system for anonymous P2P file-sharing. 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 graph properties of Youtube, Flickr and Foursquare using datasets collected by [32, 44]. We hypothesize these networks will be at least as dense as a network targeting censorship resistance, where users may be hesitant to advertise their participation. 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 P2P setting where users, essential for connectivity, may not be available all of the time.

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.

attacks, especially since researchers have demonstrated that one can acquire Gmail accounts for about $0.30 per account [35].

Site	Direct (ms)	Tor (ms)	Slowdown
Google	222	8,940	40.8
Facebook	1,821	16,711	9.2
Amazon	1,289	13,914	10.8
Twitter	684	9,229	13.5
Yahoo	1,810	15,850	8.8

Table 1: Slowdown introduced by Tor for loading popular domains in January 2013.

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 P2P systems [18, 26, 41, 47]. Even when more nodes served as exit nodes, we experienced similar disconnection properties. Our simulations show that the number of overlay hops is also affected by the social network structure. Many nodes have to traverse long overlay paths – much more than the three hop paths used by systems such as Tor – and the overlay path lengths are further inflated when only a fraction of the nodes are online.

2.3 Overlays Have Poor Transport Performance

By targeting soft censorship, we address a very different target audience than activists who demand strict anonymity but tolerate poor performance. It is important to offer good network performance (i.e., low latency) for convenient access to the wide-range of interactive web services. Also, by optimizing Unblock for common web browsing, we can incentivize longer uptimes of average users and improve overall capacity and connectivity of the network.

Current overlays are typically not designed to take full advantage of available bandwidth, and are faulted for offering degraded performance to users. To some extent this is unavoidable, since data is transferred multiple times across the overlay, resulting in higher latency and a greater possibility of traversing congested links. However, a well designed protocol can mitigate these factors.

To characterize the latency of overlay communications, we measured page load times using Tor compared with "normal" direct connections from a set of 112 geographically diverse PlanetLab nodes. As can be seen in Table 1, the page load time for popular sites increases by a factor of 10 when web pages are loaded over Tor. This is consistent with the issues outlined in [15, 48]. For the same experiment on the Alexa top 100 sites, the median page load time increased from 2.1s to 15.7s. Overlay transport performance is particularly important in the context of social overlays, where paths to an exit node could be longer than the three hops that is required by Tor.

The performance inefficiencies of overlay networks are generally attributed to two 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 [20]. 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 [39].

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.

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 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 routers in their domain and is able to disrupt communications through 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 [37].

We expect a censor will also be able to infiltrate a limited number of social links, giving it access to the overlay. It is able to generate, modify, and delete protocol messages flowing through nodes it controls, create sybils, 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 users into revealing which friends are running Unblock software. While censors have routinely targeted and blocked popular anti-censorship systems, there have been no reports in China (or many other countries) of users being punished for using anti-blocking software [36]. Such measures carry significant political, social, and economic costs [2]. If the adversary were to prohibit all encrypted communications, our approach would not be effective.

Unblock is resilient against an adversary who disrupts flows based upon host, protocol, or keyword analysis. Under active surveillance within the overlay, an adversary will be unable to determine the traffic of other users. We are as robust to protocol analysis as other overlays. The Unblock overlay can be used in parallel with protocol obfuscation like Obfsproxy to deter protocol fingerprinting.

[2]The Egyptian government's Internet shutdown in 2011 was seen to popularize rather than suppress anti-government sentiments [19].

3.2 System Overview

We proceed in three steps to build a blocking-resistant overlay targeting interactive web browsing. Our Unblock implementation is built on top of an existing social-network based overlay aimed at peer-to-peer file sharing, which allows us to test our system on an existing deployment, and hide our traffic within an existing protocol.

Social-network based overlay: Users in Unblock have real-world trust relationships. They 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. Technical mechanisms are needed for rendezvous and routing – that is, how to discover the IP addresses of friends, and the paths to exit nodes. A key challenge is that these mechanisms need to be resistant to blocking. (See Sections 3.4 and 3.5.)

Overlay augmentation: 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.)

Optimized transport: The augmented overlay path is subject to transport inefficiencies that afflict overlay mix networks. To mitigate the performance impact, Unblock specifies transport layer mechanisms for achieving reasonable latency and throughput. We understand that the protocol will need to evolve as censors develop more sophisticated protocol fingerprinting techniques, but anticipate mitigating such attacks through the same protocol obfuscation techniques employed in other anti-censorship systems. (See Section 3.7.)

3.3 Hybrid Overlay Network

As discussed earlier, relying on social network links is insufficient. To supplement connectivity, we use a hybrid overlay: we add links to approximate a *random overlay network*. The augmented network provides users with additional peers that are likely located at random points in the social network. In addition to providing users with redundant connectivity, these additional *untrusted links* counteract the stringiness of the social network, greatly reducing the graph diameter.

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

- Any augmentation mechanism that exposes relay identities can be abused by an adversary. We need to

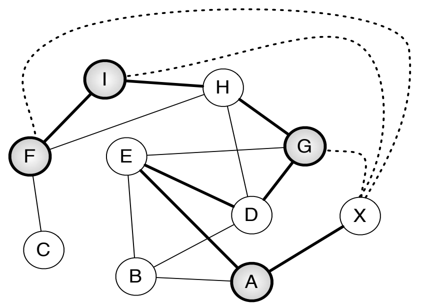

Figure 4: Example of the addition of untrusted links. In this example, an *RNL* is propagated through the path *F-I-H-G-D-E-A-X*. Nodes *F*, *I*, *G*, and *A* add themselves to the propagated *RNL*. Node *X* can then establish direct untrusted links with nodes *F*, *I*, *G*, and *A* when it receives the *RNL*. Both trusted and untrusted links can be used for data transfers.

minimize the extent to which an infiltrating adversary can exploit this mechanism.

- At least some of the additional links should be to relays that are not in a node's immediate neighborhood; this reduces path length to far away exit nodes.
- We require the set of untrusted links to be stable over time in order to reduce the leakage of node identities.

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. We consider the Sybil attack model introduced by systems such as SybilGuard [60]. When adversaries infiltrate the social overlay, we bound the number of nodes exposed to the adversary to be proportional to the number of *attack edges* - defined as a social link between an adversary controlled machine and a legitimate user. Importantly, our mechanism ensures that adversaries are not be able to uncover an arbitrary number of node identities by mounting a Sybil attack wherein they assume multiple identities behind a single attack edge.

3.3.1 Mechanisms for Disseminating Relays and Computing Network Parameters

Preliminaries: We form untrusted links by circulating collections of randomly sampled overlay nodes, referred to as *random node lists* (or RNLs). Each overlay 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

```
1. When node $x$ establishes a trusted link $l_i$, it computes
   $D(x,l_i)$, which is a boolean value that determines whether
   or not $x$ will add itself to RNLs received through $l_i$. $D(x,l_i)$
   is probabilistic with the random choice seeded using node
   ID $x$ and link ID $l_i$, and $x$ retains this computed value for
   subsequent operations.
2. Upon receiving an RNL through a trusted link $l_i$, node $x$
   does the following:
   (a) $x$ adds itself to the incoming RNL based on $D(x,l_i)$.
       Since this decision takes into account $l_i$, a node might
       add itself to an RNL received through some of its trusted
       links, but not others. This choice will be stable both
       within and across epochs.
   (b) 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.
   (c) $x$ then propagates the RNL through one of its trusted
       links, choosing the one, $l_o$, that precedes $l_i$ in
       the consistent hashing keyspace. $x$ also saves the
       propagated RNL locally.
```

Figure 5: RNL Computation.

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. These paths are are recomputed at each *epoch*, 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 [57]. 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. 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* are propagated through the same deterministic set of trusted links during an epoch. Further, it also minimizes changes between epochs when new trusted links are added to the social overlay.

Figure 4 provides an example of how *RNLs* are constructed and Figure 5 provides the pseudocode. When a node is connected to the rest of the network by a single trusted link, it will forward any incoming *RNL* back across the same link, possibly adding itself as shown in Figure 6.

In addition to *RNL* propagation, nodes also periodically perform random walks to obtain unbiased samples of network characteristics such as average uptime, average node degree, and network size. We borrow techniques from prior work [5, 24, 30], which show that a node can perform

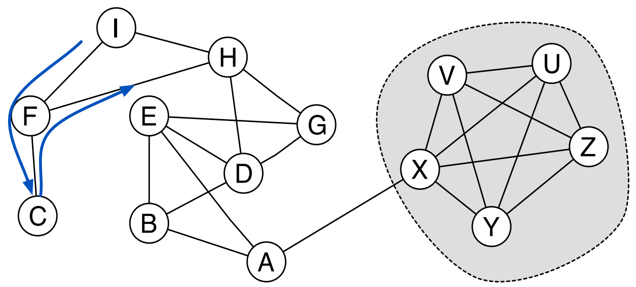

Figure 6: Example of RNL propagation. Node F has hashed its neighbors as $I \rightarrow C \rightarrow H$. Advertisements from I are passed on to C. Since C has no other friends, it will send the same list (possibly adding itself) back to F.

a random walk of length r, where r is a crude overestimate of the network diameter (say 30) in order to arrive at a random node in the network. Repeated random walks yield samples from different nodes and an average of the reported values is used as local estimates of expected node uptime and node degree. We also use the *random increasing walk* technique proposed in [5] to obtain an estimate of the network size. This technique constrains random walks to progress only from lower node ids to higher node ids, with the number of hops taken by the walk providing an estimate of the network size. Again, repeated invocations of the random walk is used to obtain an accurate estimate.

3.3.2 Policies

The *RNL* propagation mechanism outlined above allows us to propagate node identities through random paths in the social network. We now outline how we can configure the mechanism to achieve a few desired properties. Our default policy is as follows: We configure the number of relays propagated through each *RNL*, k, based on the estimated size of the network (n) and the expected probability that a typical node is online in the system (f). Our desired outcome is to have each node be connected to at least c online nodes in the system. Studies have shown that as long as $c \geq 3$, the random overlay formed by untrusted links forms a well-connected network with low diameter [6]. Each node connects to the c/f furtheest nodes in the propagating *RNL* in order to account for node uptimes. Thus, we set the number of relays stored in the propagating *RNL*, k, to be set to $log(n)+c/f$, such that the furthest c/f nodes are random nodes within the network [7,16,60]. Further, consider the following connection policy for establishing untrusted links. Upon receiving an *RNL*, let a node x with l_x trusted links connect to the $c/(f*l_x)$ farthest relays propagated by a *RNL* and establish untrusted links with each of these relays. This policy leads to improvements in availability and performance, and also mitigates bottlenecks.

Enhanced availability: Since a node x would receive l_x such messages through each of its links, it will obtain c/f untrusted links overall, each to nodes which are about $log(n)$ hops away. In expectation, c of these c/f untrusted links are likely to be active at any instant in time, thus providing greater path diversity to the node.

Reduced path lengths: The scheme outlined above also reduces path lengths. 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 [7,16,60]. This implies that the additional links can be modeled as random links, with the associated benefit of bounding the diameter of the augmented overlay [6][3].

Mitigating bottlenecks: The scheme described above also improves the number of links that cross any cut of the network graph. *RNL* messages propagate $k = log(n)+c/f$ 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.

Balanced load: We configure the decision process $D(x,l_i)$ to avoid creating untrusted links to high degree nodes. These nodes, with degree over c/f, forward *RNLs* but never add themselves. This avoids hotspots, prevents discovery and blocking of high-degree nodes, and allows for the propagation of nodes with lower degrees.

Configurable security: Any node can choose a security policy of never adding its identity to *RNLs* or connecting to nodes propagated by *RNLs*. This security setting allows users in regions that practice strong censorship to prioritize their anonymity over availability and performance.

3.3.3 Analysis

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

Limited size of RNLs: With our default policy outlined above, a mole with a single attack edge will discover the identity of $log(n)+c/f$ relays by examining its incoming *RNL*. Further, it can generate an *RNL* of its own, containing just Sybils, and propagate this across the attack edge. Unsuspecting low-degree nodes would then connect to the attacker and its Sybils. The number of such victims is also $log(n) + c/f$ since these low-degree nodes would then replace the attacker nodes in the *RNL* after processing it. Overall, each attack edge yields a total of $2*(log(n)+c/f)$ relays to the attacker. As an example, consider the scenario in Figure 6. The mole X would only receive the $log(n)+c/f$ items forwarded by A, and could receive up to $log(n)+c/f$ additional connections upon forwarding a malicious list of items back to A. The attacker would

[3]It has been observed that the fast mixing property might not hold for some fraction of the nodes in social networks [34]. These nodes would see smaller reductions in path lengths, but they will still benefit from improvements in availability as long as the uptimes of its untrusted links are not correlated.

receive no additional information from the presence of Sybils U, V, Y, and Z, as they don't have additional attack edges. In other words, a censor's ability to find participants will be limited by the number of attack edges it controls.

Stability in RNL propagation paths: Propagation paths are computed using local decisions that are remain stable over time. Thus repeated invocations of the *RNL* propagation mechanism does not reveal any additional relays than earlier *RNL* messages from the same epoch.

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.

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[4], 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*[5]. 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, 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.

The traffic cost of exit node announcements is minimal until the network becomes large. With 10,000 exit nodes updating at 10 minute intervals, the per-friend traffic cost of maintaining routes to exit nodes would be around 3kbps. Once the overlay grows large enough for this traffic to be noticeable, the internal DHT can be used to manage traffic by coordinating aggregation and regional preferences for announcements.

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.

3.5 Internal Overlay DHT

Unblock includes a DHT as a rendezvous service for locating the current IP address of peers when a node rejoins the overlay [20]. 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. Other security-focused DHT designs, such as Whanau [25] and membership-concealing overlay networks [52], address these requirements using a Byzantine-resistant algorithm across all members of a social network. We use a much simpler design, leveraging the exit node routes that already exist in the system design.

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

[5]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 either directly or through the Unblock overlay.

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[6], 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* [8], 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 an adversary can mount Sybil attacks and lower DHT availability [51]. 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 [8]. While our design incurs higher overhead and larger routing tables than membership-concealing overlays [52], this overhead was already necessary for relaying user traffic to exit nodes. By leveraging this primitive, we can ultimately use a simpler design with similar security properties.

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: 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.

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 neighbors public key, 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 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 hosts to identify whether a given node is running Unblock[7], or distinguish the protocol 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. While our existing transport uses SSL and DTLS, we can also tunnel within existing obfuscation systems to disguise our traffic as other protocols, like Skype or HTTPS [49, 54].

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 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. Unsigned exit node announcements are subjected to strict rate limiting by each node. In order to limit sybil attacks on the directory server, we require computational puzzles to be solved as part of the certification

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

[7]This probing technique is used by China to identify hidden Tor bridge nodes.

request. Because sybil exit nodes can perform a variety of availability attacks (i.e. selectively blackhole traffic), the directory server periodically checks for bad behavior from exit nodes. In the future, we could also introduce a reputation system that allows users to keep track of poorly-performing exit nodes. Individual nodes can also request additional random exit nodes from the directory service in order to detect whether it is being actively attacked.

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.

3.7 Overlay Performance

A usable system needs to provide an acceptable level of performance for typical interactive browsing. We believe the choice of protocol mechanism dramatically influence the viability of overlay transport. We examine several mechanisms, including: 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, and show these mechanisms make the Unblock protocol efficient for web browsing.

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[8]. 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 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.

[8]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 [2, 15, 39].

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, temporarily slowing or stopping incoming flow. This design is similar to mechanisms used in ATM networks [23], which suggest that some queue must be allowed to form to fully utilize the bottleneck node [46].

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 [55]).

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.

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 [14, 56] or those systems where there is a distinction between users of the system and relays that constitute the transport infrastructure (e.g., Tor).

Our prototype Unblock implementation is built on top of OneSwarm, an existing social-network based overlay aimed at peer-to-peer file sharing [20]. 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, and across an existing deployment.

We do have to ensure that the incentives for Unblock are aligned with users of the underlying system. For instance, there should be benefits for existing users to serve as relays, considering that Unblock users would not necessarily participate in the underlying network. Interestingly, by establishing trusted or untrusted links with Unblock users, connectivity increases, which in turn translates to more diversity of overlay paths between overlay users. In other words, it suffices that Unblock users are passive transport relays as opposed to active participants.

4 Evaluation

In this section, we present experiments that evaluate Unblock. Currently, the Unblock extension has been enabled by a small set of users on top of OneSwarm. We do not have access to the topology of the user base, making it difficult to quantify the performance and robustness of Unblock using that 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 [32, 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* [29]. 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 the desired threshold of active connections. This restriction protects high-degree nodes from being overloaded and restricts disclosure of high-value nodes to censors.

Figure 7(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 $3/f$. This parameter setting implies that each active node, in expectation, will have three 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 [18, 41, 47]).

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 with varying numbers of exit nodes. We obtained results that were qualitatively similar. For example, adding 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 7(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[9]. As expected, we found a linear relationship between the number of attack edges and the number of honest nodes exposed to moles. More importantly, Figure 7(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 mes-

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

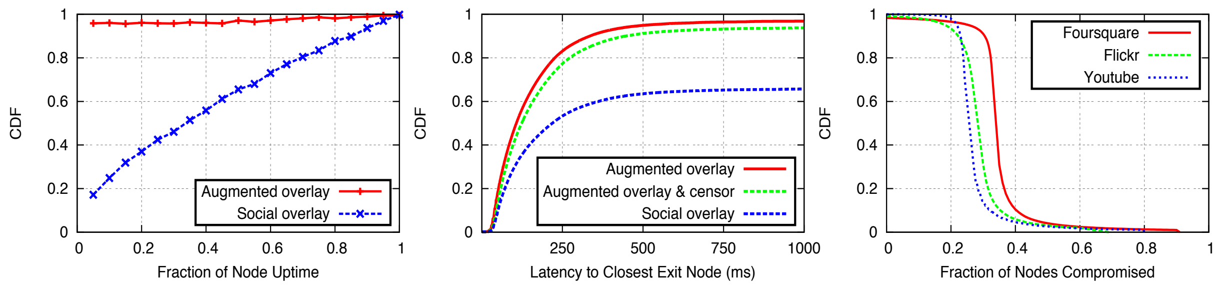

Figure 7: (a) Fraction of nodes with paths to exit nodes on the Youtube social network dataset for varying node uptimes and with 10% of the nodes being exit nodes. (b) Impact of untrusted links on latency to exit nodes when 50% of users are online. (c) Fraction of nodes with paths to exit nodes under adversarial attacks on availability.

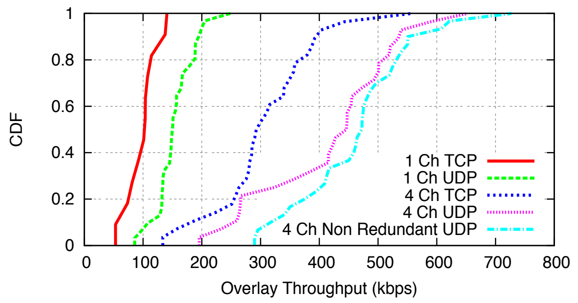

Figure 8: Throughput performance of Unblock. UDP performance improves with more paths until the endpoints are bandwidth limited. Non Redundant represents throughput when packets are only sent once, at the cost of latency.

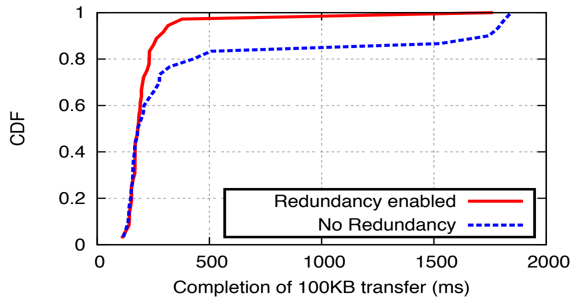

Figure 9: Latency performance within the overlay. With redundancy, latency suffers less from slow/flaky paths.

sages 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 7(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 conformed to the underlying networks imposed bandwidth rate limit of 1Mbps at each node. In Figure 8, 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 node limited the ability to transmit or receive more. We also examine the throughput of multi-path 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 9 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 performance 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 Alexa top 100 popular websites. Much of the time spent rendering a page comes from dependent resources, making network latency more important than many systems admit.

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 10 shows that Unblock has a fairly constant 2-5 second page load penalty compared with loading pages

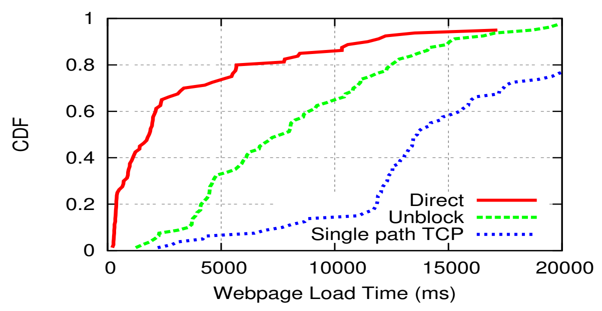

Figure 10: Web page load time across the Unblock overlay. Unblock represents load times across a three hop overlay using the optimized Unblock transport protocol. Single path TCP shows baseline load times for the same topology using per-hop TCP over a single overlay path.

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.

Performance in privacy preserving overlays is an important problem, and we note from these experiments that the transport protocol is both critical for achieving network performance, and dependent on the network overlay. An end-user based blocking resistant overlay like Unblock will gain optimal performance from a different set of protocol choices than a system built with dedicated open-access relays, and in particular the use of multi-path and redundant transfers are more important in the Unblock scenario.

5 Related Work

Providing privacy and anonymity for Internet users has been a longstanding goal of the research community, and Unblock draws on a large corpus of previous work.

Anonymous Communications: Crowds [40] provides anonymity by having an intermediary either choose a random successor or simply transmit to the destination. Tor [14] leaves the choice of relays to the source. Anonymizing networks such as Tor have been shown to be slow for reasons beyond bandwidth constraints [2]. Systems have been proposed to improve performance of these networks using congestion control [59] and network maintenance [31]. Unblock takes advantage of lessons learned from these systems, as well as years of multipath networking research [38,55]. There has also been a variety of work on anonymous self-contained darknets such as Freenet [12]. By contrast, Unblock focuses on providing unrestricted open Internet access.

Censorship resistance: Naturally, anonymizing solutions have been adapted to achieve censorship resistance [11]. A key stumbling block is that most proxy-based anonymizing solutions aren't membership concealing. The Tor developers recognized this challenge [50] and use semi-secret bridges, but they can be exposed with some effort (as we show in Section 2). Tor developers also have employed camouflage and steganography techniques, such as obfsproxy [49] and StegoTorus [54], to prevent censors from fingerprinting Tor network traffic. Membership-concealing networks have also been studied as an isolated problem [52] from censorship resistance. While these obfuscation and membership-concealing systems alone cannot provide censorship resistance, they provide useful techniques that can be applied to Unblock. Censorspoofer [53] conceals its proxies by leveraging asymmetric paths and IP spoofing to send traffic to clients. However, the system is limited by the proxy's ability to spoof addresses from within its Internet service provider, as well as limited indirect paths to contact proxies. Recently, there has been a proposal for rearchitecting the Internet to provide Tor like functionality at the network level [28]. Also related to such an approach is Telex [56], 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: Sybil defenses include strong identities, computational puzzles and bandwidth contributions to make peers prove that they are not Sybils, and leveraging social networks [25, 60]. Defenses based on social networks, such as SybilGuard [60], 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] Google Transparency Report. http://google.com/transparencyreport.

[2] ALSABAH, M., BAUER, K., GOLDBERG, I., GRUNWALD, D., MCCOY, D., SAVAGE, S., AND VOELKER, G. Defenestrator: Throwing out windows in Tor. In *PETS* (2011).

[3] APPELBAUM, J. Tor bridge nodes, 2011. Private communication.

[4] BALL, J. Internet anti-censorship tools are being overwhelmed by demand. The Washington Post, 2012-10-21, 2012.

[5] BAWA, M., GARCIA-MOLINA, H., GIONIS, A., AND MOTWANI, R. Estimating aggregates on a peer-to-peer network.

[6] BOLLOBAS, B. *Random Graphs*. Cambridge University Press, 2001.

[7] BOYD, S., GHOSH, A., PRABHAKAR, B., AND SHAH, D. Gossip algorithms: design, analysis and applications. In *INFOCOM* (2005).

[8] CAESAR, M., CASTRO, M., NIGHTINGALE, E. B., O'SHEA, G., AND ROWSTRON, A. Virtual ring routing: network routing inspired by DHTs. In *SIGCOMM* (2006).

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

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

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

[12] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. Freenet: a distributed anonymous information storage and retrieval system. In *PETS* (2001).

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

[14] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: the second-generation onion router. In *USENIX Sec.* (2004).

[15] DINGLEDINE, R., AND MURDOCH, S. Why Tor is slow and what we're going to do about it. https://blog.torproject.org, 2009.

[16] FLAXMAN, A. D. Algorithms and models for the web-graph. In *Expansion and Lack Thereof in Randomly Perturbed Graphs*, W. Aiello, A. Broder, J. Janssen, and E. Milios, Eds. 2008.

[17] FLOSS MANUALS. https://howtobypassinternetcensorship.org.

[18] GUMMADI, K. P., DUNN, R. J., SAROIU, S., GRIBBLE, S. D., LEVY, H. M., AND ZAHORJAN, J. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP* (2003).

[19] HOWARD, P., DUFFY, A., FREELON, D., HUSSAIN, M., MARI, W., AND MAZAID, M. Opening closed regimes: What was the role of social media during the arab spring? http://pitpi.org/?p=1051, 2011.

[20] ISDAL, T., PIATEK, M., KRISHNAMURTHY, A., AND ANDERSON, T. Privacy-Preserving P2P Data Sharing with OneSwarm. In *SIGCOMM* (2010).

[21] KARLIN, J., ELLARD, D., JACKSON, A., JONES, C., LAUER, G., MANKINS, D., AND STRAYER, W. Decoy routing: Toward unblockable internet communication. In *USENIX FOCI* (2011).

[22] KELLY, S., AND COOK, S. Freedom on the net 2011: A global assessment of Internet and digital media. *Freedom House* (2011).

[23] KUNG, H. T., BLACKWELL, T., AND CHAPMAN, A. Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation and statistical multiplexing. In *SIGCOMM* (1994).

[24] LE MERRER, E., KERMARREC, A.-M., AND MASSOULIE, L. Peer to peer size estimation in large and dynamic networks: A comparative study. In *HPDC* (2006).

[25] LESNIEWSKI-LASS, C., AND KAASHOEK, M. F. Whanaungatanga: Sybil-proof distributed hash table. In *NSDI* (2010).

[26] LI, J., STRIBLING, J., MORRIS, R., KAASHOEK, F., AND GIL, T. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. *IEEE Infocom* (2005).

[27] LIANG, G. Surveying Internet usage and its impact in seven chinese cities. Tech. rep., Chinese Academy of Social Sciences, 2007.

[28] LIU, V., HAN, S., KRISHNAMURTHY, A., AND ANDERSON, T. Tor Instead of IP. In *HotNets* (Nov. 2011).

[29] MADHYASTHA, H. V., ISDAL, T., PIATEK, M., DIXON, C., ANDERSON, T., KRISHNAMURTHY, A., AND VENKATARAMANI, A. iPlane: An Information Plane for Distributed Services. In *OSDI* (2006).

[30] MASSOULIÉ, L., LE MERRER, E., KERMARREC, A.-M., AND GANESH, A. Peer counting and sampling in overlay networks: random walk methods. In *PODC* (2006).

[31] MCLACHLAN, J., TRAN, A., AND ANDYONGDAE KIM, N. H. Scalable onion routing with torsk. In *CCS* (2009).

[32] MISLOVE, A., MARCON, M., GUMMADI, K. P., DRUSCHEL, P., AND BHATTACHARJEE, B. Measurement and analysis of online social networks. In *IMC* (2007).

[33] MISLOVE, A., POST, A., DRUSCHEL, P., AND GUMMADI, K. P. Ostra: leveraging trust to thwart unwanted communication. In *NSDI* (2008).

[34] MOHAISEN, A., YUN, A., AND KIM, Y. Measuring the mixing time of social graphs. In *IMC* (2010).

[35] MOTOYAMA, M., MCCOY, D., LEVCHENKO, K., VOELKER, G. M., AND SAVAGE, S. Dirty Jobs: The Role of Freelance Labor in Web Service Abuse. In *USENIX Sec.* (2011).

[36] OPENNET INITITATIVE. Internet filtering country profiles. opennet.net.

[37] R. CLAYTON AND S. MURDOCH AND R. N. M. WATSON. Ignoring the great firewall of China. In *PETS* (2006).

[38] RAICIU, C., HANDLY, M., AND WISCHIK, D. Coupled congestion control for multipath transport protocols. *IETF RFC 6356* (Oct. 2011).

[39] REARDON, J., AND GOLDBERG, I. Improving Tor using a TCP-over-DTLS tunnel. In *USENIX sec.* (2009).

[40] REITER, M. K., AND RUBIN, A. D. Crowds: anonymity for Web transactions. *ACM Trans. Inf. Syst. Secur.* (1998).

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

[42] ROBERTS, H., ZUCKERMAN, E., YORK, J., FARIS, R., AND PALFREY, J. 2010 circumvention tool usage report. The Berkman Center, 2010.

[43] ROBERTS, H., ZUCKERMAN, E., YORK, J., FARIS, R., AND PALFREY, J. International bloggers and Internet control: Full survey results. The Berkman Center, 2011.

[44] SCELLATO, S., MASCOLO, C., MUSOLESI, M., AND LATORA, V. Distance matters: Geo-spacial metrics for online social networks. In *WOSN* (2010).

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

[46] SHALUNOV, S., HAZEL, G., IYENGAR, J., AND KUEHLEWIND, M. Low extra delay background transport (LEDBAT). In *IETF Internet Draft* (2006).

[47] STUTZBACH, D., AND REJAIE, R. Understanding churn in peer-to-peer networks. In *SIGCOMM* (2006).

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

[49] TOR PROJECT. Obfsproxy. https://torproject.org/projects/obfsproxy.

[50] China blocking Tor. https://blog.torproject.org, Mar. 2010.

[51] URDANETA, G., PIERRE, G., AND VAN STEEN, M. A survey of DHT security techniques. *ACM Computing Surveys 43*, 2 (2011).

[52] VASSERMAN, E. Y., JANSEN, R., TYRA, J., HOPPER, N., AND KIM, Y. Membership-concealing overlay networks. In *CCS* (2009).

[53] WANG, Q., GONG, X., NGUYEN, G. T. K., HOUMANSADR, A., AND BORISOV, N. Censorspoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing. In *CCS* (2012).

[54] WEINBERG, Z., WANG, J., YEGNESWARAN, V., BRIESEMEISTER, L., CHEUNG, S., WANG, F., AND BONEH, D. StegoTorus: A camouflage proxy for the Tor anonymity system. In *CCS* (2012).

[55] WISCHIK, D., RAICIU, C., GREENHALGH, A., AND HANDLEY, M. Design, implementation and evaluation of congestion control for multipath TCP. In *NSDI* (2011).

[56] WUSTROW, E., WOLCHOK, S., GOLDBERG, I., AND HALDERMAN, J. A. Telex: Anticensorship in the Network Infrastructure. In *USENIX sec.* (2011).

[57] XIE, Y., YU, F., ARCHAN, K., GILLUM, E., GOLDSZMIDT, M., AND WOBBER, T. How dynamic are IP addresses. In *SIGCOMM* (2007).

[58] Access to YouTube blocked until further notice because of non-islamic videos. http://en.rsf.org/pakistan-youtube-access-unblocked-after-27-02-2008,25889.html.

[59] YU, F., GOPALAKRISHNAN, V., LEE, D., AND RAMAKRISHNAN, K. K. Nemor: A congestion-aware protocol for anonymous peer-based content distribution. In *IEEE P2P* (2011).

[60] YU, H., KAMINSKY, M., GIBBONS, P. B., AND FLAXMAN, A. D. SybilGuard: defending against Sybil attacks via social networks. *SIGCOMM* (2006).

[61] ZAFARANI, R., AND LIU, H. Social computing data repository at ASU. http://socialcomputing.asu.edu, 2009.