MassBrowser: Unblocking the Censored Web for the Masses, by the Masses

Milad Nasr, Hadi Zolfaghar, Amir Houmansadr, and Amirhossein Ghafari
College of Information and Computer Sciences
University of Massachusetts Amherst
{milad, hadi, amir, ghafari}@cs.umass.edu
Webpage: https://massbrowser.cs.umass.edu

***Abstract*—Existing censorship circumvention systems fail to offer reliable circumvention without sacrificing their users' QoS and privacy, or undertaking high costs of operation. We have designed and implemented a censorship circumvention system, *MassBrowser*, whose goal is to offer *effective censorship circumvention* to a large mass of censored users, with a *high quality of service (QoS)*, *low cost of operation*, and *adjustable privacy protection*. Towards this, we have made several key decisions in designing our system. First, we argue that circumvention systems should *not* bundle strong privacy protections (like anonymity) with censorship circumvention. Additional privacy properties should be offered to the users of circumvention systems as optional features which can be enabled by specific users or on specific connections (perhaps by trading off some QoS). Second, we have engineered MassBrowser by combining various state-of-the-art circumvention techniques to ensure strong censorship resilience at a very low cost of operation (i.e., $0.0001 per censored client per month when deployed at a large scale). In particular, MassBrowser aims at increasing the collateral damage of censorship by employing a "mass" of normal Internet users, from both censored and non-censored areas, to serve as circumvention proxies. Also, MassBrowser uses various techniques, like CDNBrowsing, to optimize the loads on circumvention proxies.**

We have built and deployed MassBrowser as a fully operational system with end-user GUI software for major operating systems. Our system has been in the beta release mode for over a year with hundreds of invited users from major censored countries testing it on a daily basis.

I. Introduction

The Internet plays a crucial role in today's social and political movements by facilitating the free circulation of speech, information, and ideas; democracy and human rights throughout the world critically depend on preserving and bolstering the Internet's openness. Consequently, repressive regimes, totalitarian governments, and corrupt corporations regulate, monitor, and restrict the access to the Internet, which is broadly known as Internet *censorship*. The techniques commonly used to enforce censorship include IP address blocking, DNS hijacking, and TCP content filtering [41], [25], [39], [65] to block access to certain destinations or to prevent certain forms of content from being transmitted. To ensure compliance and to detect undercover political/social activists, repressive regimes additionally utilize advanced networking tools, including deep packet inspection (DPI), to prevent the use of the censorship circumvention technologies by their citizens [81], [16], [36], [37].

To restore the openness of the Internet, researchers have designed and deployed an arsenal of tools [25], [66], [14], [42], [47], [78], [33], [77], [8], [84], [32], [53] to help users bypass censorship. Such tools, known as *circumvention systems*, deploy a variety of techniques ranging from IP indirection to onion routing to traffic obfuscation [39], [65].

Key shortcomings of existing systems: Unfortunately, existing circumvention systems fail to offer reliable, low-cost circumvention without sacrificing their users' QoS or privacy. Specifically, existing systems suffer from one or all of the following weaknesses: **(1) Easily blocked:** A majority of in-the-wild circumvention systems, including Tor, Lantern, Psiphon, and VPNs, work by setting up *proxy* servers outside the censorship regions, which relay traffic for censored users. Unfortunately, the proxies are implemented in a way that are easily blockable by the censors, e.g., due to using a small set of IP addresses that can get enumerated and blacklisted by the censors [81], [61], [79], [16]. **(2) Costly to operate:** To resist proxy blocking by the censors, recent circumvention systems have started to deploy proxies on shared-IP platforms such as CDNs [45], App Engines [26], and Cloud Storage services [7], a technique broadly referred to as *domain fronting* [20]. This mechanism, however, is prohibitively expensive [46] to be used at large scale. **(3) Poor QoS:** Proxy-based circumvention systems like Tor and its variants [33], [44], [70] are infamous for their low quality of service (e.g., very high latencies and low bandwidths). This is primarily due to the imbalance between the bandwidth demand from censored users versus the bandwidth provided by the proxies (e.g., Tor's ≈ 6500 relays need to proxy traffic for around two million daily users [64], while some users leverage Tor for bandwidth-extensive applications like BitTorrent. **(4) Lack of user-adjustable privacy:** Existing circumvention systems do not give users much control on their privacy protection while using such systems. On one hand, some circumvention systems like Tor bundle strong privacy protections like anonymity with circumvention, causing huge degradations to QoS and therefore scaring away

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.23340
www.ndss-symposium.org

typical Internet users. On the other hand, VPNs and one-hop proxy-based systems provide weak privacy protections to their users regardless of specific privacy needs of different users. **(5) Hard to deploy:** Modern circumvention systems proposed in academia are impractical to be used at large scale due to various reasons. For instance, decoy routing systems [32], [84], [38] require wide adoption by Internet ISPs, and tunneling systems [33], [35], [44], [70] can be disabled by third-party service providers they use for tunneling.

Our contributions: In this paper, we present the MassBrowser circumvention system, which aims at addressing the discussed shortcomings of existing circumvention solutions. Towards this goal, we base our design on the *separation of properties (SoP) principle*: *the key feature targeted by an effective circumvention system should be blocking resistance, and other features such as anonymity and browsing privacy should be provided as optional features to the users.* We argue for the SoP principle based on the real-world observation [10], [11], [71], [72], [21] that the majority of censored users are solely interested in blocking resistance, e.g., to be able to access blocked news articles and be able to communicate through blocked social networks, but for the *majority* of the censored users properties like anonymity are not a concern. This is evident by the fact that "public" VPNs, "public" HTTP proxies, and centralized circumvention systems like Lantern [40] and Psiphon [58] are the most popular among censored users in China and Iran [71], [72] (when compared to privacy-preserving alternatives like Tor) despite the fact that they provide no anonymity or browsing privacy [11]. MassBrowser users can enable stronger privacy protections for specific (e.g., more sensitive) connections by compromising on the QoS of those specific connections.

The SoP principle enables us to optimize the performance of MassBrowser around blocking resistance, and to offer features like anonymity and browsing privacy as options to the users. We will demonstrate how basing our design on the SoP enables us to overcome the circumvention shortcomings discussed above. Note that, unlike VPNs and one-hop proxy circumvention systems, MassBrowser does not neglect privacy protection. Instead, it hands the control over the privacy-usability tradeoff to the users.

In addition to separating circumvention from add-on privacy protections, we make several decisions on the technical approach of MassBrowser to optimize its resilience, QoS, and cost of operation. First, MassBrowser leverages censored users in various regions to help each other bypass censorship by implementing client-to-client proxying. This is motivated by our measurements showing that users in various censored regions are subject to different censorship regulations. Second, MassBrowser uses volunteer Internet users living in non-censored areas to proxy traffic for censored users. Third, we have implemented advanced TCP/UDP NAT traversal mechanisms into MassBrowser, allowing censored clients to connect to other clients and volunteer proxies behind NAT. This provides strong censorship resilience due to the high collateral damage of blocking NATed Internet users. Finally, MassBrowser combines several techniques including CacheBrowsing [30], selective proxying, and Domain Fronting [20] to optimize the QoS of circumvention connections while minimizing its operational costs. As shown in Section VI-C, we estimate the total cost of deploying MassBrowser to be no more than *$0.0001 per active client per month* once deployed at large scale.

Deployment: MassBrowser has been under active development and testing for over two years. We have implemented cross-platform (Mac, Windows, and Linux) end-user GUI software for novice clients and volunteers. Our client and volunteer software is written in NodeJS in approximately 50K lines of code. We have also implemented a browser bundle which contains a customized Firefox browser pre-configured to work out of the box.

MassBrowser's backend services, which we refer to as the Operator, is written in Python (approximately 10K lines of code). The Operator runs a range of services essential to the reliable operation of MassBrowser, from strategic pairing of clients and proxies, to monitoring the reachability and health of various parts of the system, to measuring the performance of MassBrowser in censored countries.

Our system is currently in the beta release mode, and we have been continuously testing and improving its performance based on feedback from hundreds of invited volunteer clients from various censored countries, including China, Turkey, and Iran. Joining our system is currently invitation-based only, and we expect to open the project to the public soon (pending a security code audit by a third-party organization, Subgraph[1]). Our software as well as the source code can be obtained from https://massbrowser.cs.umass.edu.

Paper's Organization: The rest of this paper is organized as follows. We start by overviewing existing circumvention solutions and their weaknesses in Section II. In Section III, we introduce the core ideas used in the design MassBrowser, and discuss how these ideas help MassBrowser overcome the challenges of prior circumvention systems. We discuss the technical decisions we made in designing MassBrowser in Section IV, and present MassBrowser's implementation details in Section V. Finally, we provide various performance evaluations in Section VI and discuss MassBrowser's privacy guarantees in Section VII. The paper is concluded in Section VIII.

II. Background on Circumvention Systems

Internet censorship is undoubtedly the biggest threat to the freedom of speech, ideas, and information across the globe [23]. To help censored users regain open access to the Internet, researchers and practitioners have designed and deployed an arsenal of tools known as *circumvention systems* [25], [66], [14], [42], [47], [78], [33], [77], [8], [84], [32], [53], [25], [39], [65]. Censorship authorities utilize their censorship technology to prevent the use of such censorship circumvention technologies by their citizens [81], [16], [36], [37], i.e., they block circumvention systems. In the following, we overview the major classes of circumvention systems and their weaknesses.

Proxy-based Systems The most common approach used by circumvention systems is to run network proxies outside the censorship region, and use them to relay the traffic of

[1] https://subgraph.com/about-us/index.en.html

censored users to censored Internet destinations. Many in-the-wild circumvention systems such as Tor [15], Psiphon [58], Lantern [40], and VPN services [56], [52] deploy circumvention proxies in different ways to help censored users. Most circumvention systems [58], [67], [52], [40] use simple, single-hop proxies, while others [15], [57] use more complex models for proxy deployment. Tor, in particular, has implemented various *pluggable transports* [57], [53] to further hinder blocking by obfuscating the characteristics of Tor traffic.

Domain Fronting Domain fronting [20] is a blocking-resistant approach for setting up circumvention proxies. In this approach, the circumvention proxy is hosted on shared-IP infrastructures such as content delivery networks (CDNs), App Engines, and Cloud Computing services. Therefore the domain-fronted proxy will share its IP address with other, oblivious services making any censorship attempt susceptible to collateral damage. For instance, blocking a domain-fronted proxy hosted on a CDN requires the censors to block all the web content served by that CDN. CloudTransport [7] is an older variation of domain fronting, in which proxies are run over shared cloud storage services. Recently, several major content providers, including CloudFlare [12], Google [27], and Amazon [3], have started to disable or interfere with domain fronting, presumably in the fear of losing their market inside censored countries.

CacheBrowsing CacheBrowsing [30], [85] is a technique to fetch CDN-hosted censored content directly from CDN edge servers with no need to use circumvention proxies. To do so, various bootstrapping mechanisms are used to enable a censored client to locate the CDN edge servers hosting her censored content of interest. CacheBrowsing is significantly cheaper [30], [46] than domain fronting since the CDN expenses are paid by the publishers of the censored content, not the circumvention operators. On the other hand, CacheBrowsing has a more limited scope as it can only be used to unblock certain censored content, i.e., those hosted on CDNs. In this paper, we leverage CacheBrowsing as a technique to optimize load on circumvention proxies, but not as a standalone circumvention system.

Protocol Tunneling Several circumvention proposals suggest to tunnel traffic through popular Internet services that are unlikely to be entirely blocked by the censors. For instance, FreeWave [33] tunnels circumvention traffic through VoIP services like Skype, and CovertCast [44] tunnels traffic through video streaming services. Alternatively, Rook [70] and Castle [29] tunnel traffic through gaming applications, and Sweet [35] tunnels through email communications. To block a tunneling circumvention system, the censors will need to block the oblivious service being used for tunnel, which has significant collateral damage to the censors [9]. On the downside, tunneling circumvention systems offer impractical QoS (e.g., high latencies and low bandwidth) due to the limitations imposed by their hosting services.

Decoy Routing Decoy routing aims at defeating IP address blocking by integrating circumvention software into the routing infrastructure [84], [32], [38], [50]. In decoy routing, censorship circumvention is implemented with help from a number of friendly Internet autonomous systems, called *decoy ASes*. Each decoy AS modifies some of its routers (e.g., its border routers) such that they deflect the Internet traffic of censored users to the blocked Internet destinations requested by the users. By design, decoy routing defeats IP address blocking, however, it is prone to particular routing-based blocking attacks known as RAD [62], [34], [49]. Requiring deployment by a number of in-the-wild ISPs is a major obstacle to the real-world deployment of decoy routing systems.

TABLE I. WEAKNESSES OF MAJOR TYPES OF CIRCUMVENTION SYSTEMS

Category	Easily blocked	Costly	Poor QoS	Deployability
Proxy-Based	●	◐	●	○
Domain Fronting	○	●	○	○
CacheBrowsing	○	○	●	○
Tunneling	○	◐	●	◐
Decoy Routing	○	◐	○	●

A. *Weaknesses of Existing Systems*

Here, we summarize the main weaknesses of existing circumvention systems, as summarized in Table I:

1) Easy to block: Proxy-based circumvention systems, which encompass the majority of in-the-wild systems like Tor, Psiphon, and VPN services [52], [56] can easily get blocked by the censors who enumerate their limited, small set of proxy IP addresses [81], [61], [79], [16]. The censors can also use more advanced techniques like traffic analysis and active probing to block various kinds of circumvention systems [81], [16], [61], [31], [24], [62].

2) Costly to operate: As introduced earlier, domain fronting aims at resisting IP address filtering by setting up proxies on shared-IP platforms such as CDNs, App Engines, and Cloud services. However, due to the prohibitively high costs of domain fronting [46], domain fronting is not used for circumvention proxying at scale, and recent proposals suggest to use domain fronting only for circumvention signaling, but not for proxying [63]. Several protocol tunneling systems [7] similarly need to some pay service providers for using their service, and decoy routing services require large investment in order to be deployed by Internet ISPs [49].

3) Poor QoS: Proxy-based circumvention systems like Tor suffer from low quality of services (e.g., high latencies) due to high congestion on the proxies. Various factors contribute to such congestion, most importantly the small number of proxies compared to clients, as well as the use of circumvention system by many clients for accessing bandwidth-extensive content such as copyright infringed multimedia content. Tunneling circumvention systems like FreeWave [33], Sweet [35], and CoverCast [44] offer low bandwidth and high latencies to the clients as they are constrained by the quality of service of their host services. CDNBrowsing systems [30], [85] offer good latencies but can only be used to browse specific types of censored websites.

4) Lack of user-adjustable privacy: Existing circumvention systems do not give users much control on their privacy protection while using such systems. On one hand, some circumvention systems like Tor bundle strong privacy protections like anonymity with circumvention, causing huge degradations to QoS and therefore scaring away typical Internet users.

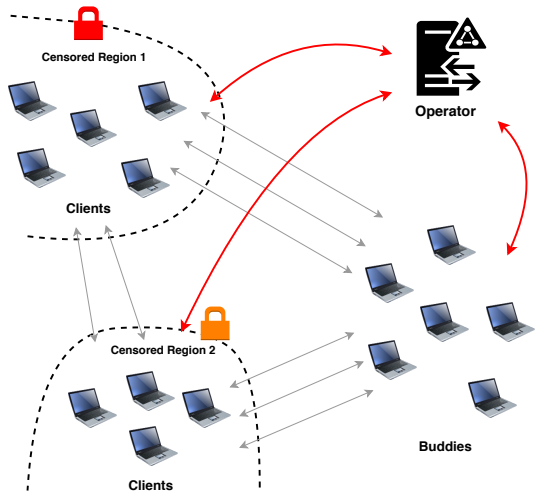

Fig. 1. The main setting of MassBrowser.

On the other hand, VPNs and one-hop proxy-based systems provide weak privacy protections to their users regardless of specific privacy needs of different users.

5) Hard to deploy: Some of the circumvention systems proposed in the literature are impractical to be used at large scale, despite offering reasonable blocking resistance and QoS. For instance, decoy routing systems [32], [84], [38], [50] require wide adoption by Internet ISPs, and tunneling systems [33], [35], [44] can be trivially disabled by the third-party service providers they use for tunneling.

III. Sketch of our Approach

Figure 1 shows the high-level setting of MassBrowser. MassBrowser is a volunteer-run proxy-based system: it leverages normal Internet users with access to the free Internet to proxy censored web traffic for censored users. It also uses censored users to proxy traffic for other censored users who face different censorship restrictions (e.g., those who reside in different censoring countries). We will refer to the censored clients of MassBrowser as *Clients*, and we will use *Buddies* to refer to MassBrowser's proxies. Note that a Buddy can be either a volunteer (non-censored) party or a censored client of MassBrowser.

The central component of MassBrowser is a hard-to-block *Operator* service that oversees and enforces MassBrowser's key functionalities, particularly, by strategically matching Clients to Buddies based on the preferences of Clients and Buddies as well as their geographic locations, their available bandwidth, and other conditions. As will be evaluated, Operator can handle very large numbers of users with tiny operational costs by making minimal communications.

We will provide further details about these components and their interactions throughout the paper. Next, we present the key ideas behind the design of MassBrowser.

A. MassBrowser's Threat Model

We assume that MassBrowser Clients are located inside censoring regions. The censorship authorities monitor the Internet communications of the censored Clients, and are able to block or interfere with any connection from such Clients to Internet destinations. Censors are also able to act as Clients or Buddies in order to gain information about the system and to disrupt the system to the best of their ability. However, we assume that censors are not capable of tampering with users' devices (e.g., installing monitoring softwares on their devices), as this will disable any privacy-enhancing tool.

We assume the players in our system to be *rational*. A rational censor tries to minimize the costs and collateral damages incurred by its actions, such as interfering with benign, popular Internet services. Buddies are rational in that they are willing to help censored users as long as this does not pose any risks to themselves. For example, a Buddy will not let Clients use her device to deploy network attacks (e.g., port scan, sending spam email) or to access controversial destinations that will get the Buddy in trouble.

We also assume that the *censors do not penalize normal users for the sole act of using a circumvention software or accessing an unblocked destination*, unless the websites accessed are directly related to major criminal offenses. Although using circumvention tools is considered illegal in many censoring countries, penalizing Internet users merely for using a circumvention software has been extremely rare in most countries [10]. For example, as of 2017, Facebook has over 17 million users from Iran accounting for over 20% of the population [5], despite it having been blocked for more than 8 years. Our threat model assumes that the censored clients are aware of, and accept the (negligible) risks of using a circumvention software.

B. Separation of Properties

We rely on the *separation of properties (SoP) principle* in order to overcome the practical shortcomings of existing circumvention solutions. The SoP principle states that *the key feature targeted by a circumvention system must be blocking resistance, and additional properties such as anonymity and browsing privacy should be provided as optional features to the users.* The SoP principle is based on the real-world observation [10], [11], [71], [72], [21] that the majority of censored users are solely interested in blocking resistance, e.g., to be able to access blocked news articles or to be able to communicate through blocked social networks; however, the *majority* of the censored users are not seeking properties like anonymity [83]. Our claim is supported by the ostensible popularity of "public" VPNs, "public" HTTP proxies [71], [72], [83], [82] and centralized circumvention systems like Lantern [40] and Psiphon [58], in contrast to privacy-focused solutions such as Tor. For instance, an estimated 31% of Chinese users use VPN services [83] compared to Tor's only 2 millions daily users globally.

The SoP principle allows us to run single-proxy circumvention connections, which improves the QoS-cost tradeoff. Also, the principle allows us to restrict the use of our circumvention proxies to accessing censored content *only*. This not only reduces congestion on the proxies (therefore improving the QoS-cost tradeoff), but also increases the potential number of volunteer proxies by significantly reducing the legal consequences of running circumvention proxies, which has been a

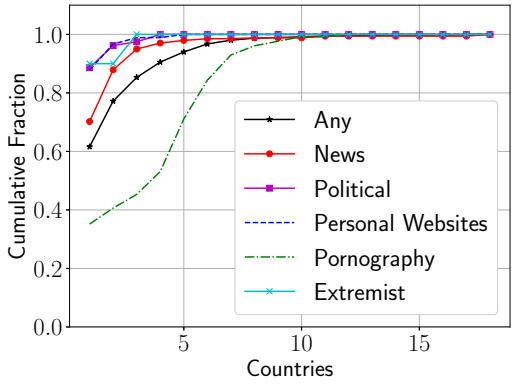

Fig. 2. The cumulative distribution of the number of countries that a website is blocked in, for different content types.

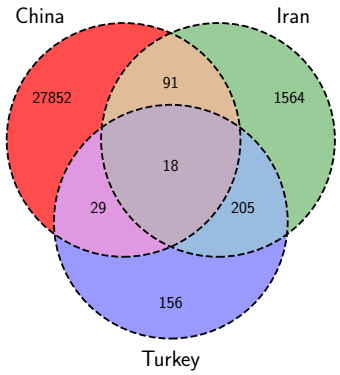

Fig. 3. Comparing website blacklists in three major censorship regions.

major issue for general purpose circumvention systems like Tor [6], [4].

Our system provides *user-adjustable privacy*, unlike existing circumvention systems. Specifically, VPNs and single-hop circumvention systems like Psiphon and Lantern provide no privacy protection, and Tor provides strong privacy for all users, across all connections. In MassBrowser, however, a client can choose to protect his sensitive connections by tunneling them through an integrated interface with Tor (and therefore only pay a QoS cost for those sensitive connections).

C. Client-to-Client Proxying

The key to the resilience and QoS of any volunteer-based circumvention system like ours is to have a balanced ratio of proxying capacity to circumvention bandwidth demand. A key technique we use in MassBrowser is having censored clients help other censored clients by proxying their traffic. What enables us to do so is *the difference in censorship blacklists across different censorship regions*, as demonstrated in various measurement studies [51], [54], [55].

To demonstrate the extent of usability of client-to-client proxying, we use the measurement data provided by ICLab [51] and GreatFire [81]. For each website, we compute the number of countries that the website is blocked in. As shown in Figure 2, *the majority of censored websites are only blocked in one or two countries*. This is specially more apparent for political and news content, as censorship rules on such content are heavily dependent on geographic regions. Therefore, an Iranian MassBrowser client can help a Chinese MassBrowser client to access webpages blocked in China, and vice versa. Figure 3 illustrates the intersection of blocked domains in three major censoring countries of China, Iran, and Turkey. We believe that using clients-to-client proxying is a major step towards balancing the ratio of circumvention proxies to circumvention traffic, and therefore improving circumvention QoS. Also, as we will discuss later, using clients/volunteers for proxying also offers strong censorship resilience properties by increasing the collateral damage of censorship.

D. Leveraging Volunteer Proxies

MassBrowser complements client-to-client proxying by employing volunteer Buddies who live in non-censored areas (e.g., to proxy content commonly blocked across various censoring countries). We use various techniques (discussed in Section IV-B) to encourage a wide adoption by volunteers. Note that, we are not the first to suggest to use volunteers in deploying circumvention proxies. In the following, we compare MassBrowser to alternative circumvention systems that also use volunteer proxies.

uProxy [67]: uProxy (currently, deprecated [67]) is another proposal to use volunteer Internet users as proxies for censored users. uProxy's original design [69] used the WebRTC protocol to connect a censored user to a volunteer proxy with an installed Chrome plugin. The uProxy project lately shifted towards using Shadowsocks [68] for connecting users to servers. uProxy did not use any central operator as in MassBrowser; instead, a uProxy censored user was supposed to know a friend outside the censorship region to act as her proxy. That is, uProxy would enable clients to set up "private" proxies, very much similar to private VPNs. We believe that this is not a scalable solution, as many censored users do not have close friends with access to the free Internet to help them.

FlashProxy [22], [19]: FlashProxy (currently, deprecated [22]) suggested to use volunteer websites to recruit ephemeral proxies. The volunteer website would load a particular JavaScript on each of its visitors, turning them into ephemeral proxies for censored clients. Even though a FlashProxy volunteer website would present a banner to its visitors informing them of the process, the visitors had no way to opt out except by refraining from visiting that website. We believe that high-visitor websites are unlikely to become volunteers as this may decrease their visitors. Additionally, the censors may retaliate by simply censoring (or even attacking) the volunteer websites.

Snowflake [63]: Snowflake is the successor of the FlashProxy project and uses some of the core communication protocols

of uProxy [69], e.g., its WebRTC communication schemes. Similar to FlashProxy, Snowflake converts the visitors of some volunteer websites into circumvention proxies by loading a JavaScript. Therefore, we argue that a major challenge to Snowflake is adoption by volunteer websites: a volunteer website may get the target of censorship or cyberattacks by the censors, and therefore we do not expect adoption by major websites. Note that deployment by low-visitor websites does not help since the number of the proxies is proportional to the number of the visitors to the volunteer websites. Also, similar to Flashproxy, users in Snowflake have no way to opt out except by refraining from visiting the volunteer websites. By contrast, in MassBrowser we use Internet users to *knowingly* and voluntarily proxy traffic for censored users. Also, we use a hard-to-block central entity (the Operator) to strategically matchmake clients and volunteer proxies. MassBrowser implements various traffic optimization techniques and selective proxying to encourage volunteer proxying by respecting their preferences.

VPNGate [52]: VPNGate is a network of volunteers running VPN software open to the public. The VPNGate system maintains the list of all volunteer VPNs, and publishes the list on its webpage [75] for the interested clients. Unfortunately, VPNGate does not employ effective mechanisms to resist blocking, and therefore it is trivially blockable by the censors. The VPNGate website contains fake VPN IP addresses to prevent the censors from blacklisting the VPN IPs in bulk, however, the censors can easily identify and ignore such fake IPs by trying to connect to them through VPN protocols. In fact, the majority of VPNGate proxies appear to be currently blocked in China [73], [74]. By contrast, in MassBrowser a blocking resistant Operator component establishes the connections between clients and proxies, preventing the censors from enumerating the proxies. Even if the censors enumerate MassBrowser's Buddy IPs, they can not block them without collateral damage as such IPs are NATed IPs with ephemeral port numbers, i.e., they change their port numbers for *every connection*. Additionally, MassBrowser deploys traffic obfuscation to defeat traffic analysis, while VPNGate's VPN traffic is trivially detectable at the network layer. As another distinction, MassBrowser employs various selective proxying techniques to optimize traffic load on volunteer proxies.

E. How MassBrowser Addresses Circumvention Issues

Here we summarize how MassBrowser addresses the major circumvention issues discussed in Section II-A. This will be further expanded later on.

1) Blocking resistance: As discussed earlier, proxy enumeration is the most common technique used by the censors to block circumvention systems. Proxy enumeration is feasible in practice due to two reasons; first, the small number of proxy IP addresses used by typical circumvention systems enables the censors to enumerate all the IPs within a short interval [81]. Second, typical circumvention proxies use *dedicated* IP addresses that once identified can be blocked with no collateral damage. Domain fronting defeats IP blocking by using shared IP addresses, however is prohibitively expensive as a scalable solution.

MassBrowser deploys a large number of proxies run by normal Internet users—either censored clients or non-censored volunteers. Therefore, the number of relay IPs of MassBrowser scales with the number of its users. Also, as the relays are run by normal Internet users, they are expected to frequently change IP addresses and use shared NATed IPs. Therefore, blocking such moving targets can impose significant collateral damage to the censors.

2) Cost of operation: Similar to (the prohibitively expensive) domain fronting [20] and CloudTransport [7] systems, MassBrowser makes use of shared IP addresses to defeat IP enumeration. By contrast, MassBrowser is significantly cheaper to operate as the voluminous circumvention traffic is proxied through censored clients and volunteer proxies. Also, while MassBrowser's Operator is implemented as a domain-fronted service to resist blocking, it only costs MassBrowser an estimated *$0.001 per active client per month* due to the small volume of its signaling traffic.

3) QoS: MassBrowser combines several complimentary techniques to offer a high QoS. First, it leverages CacheBrowsing [30] to minimize the traffic load on the proxies. Second, being based on the SoP principle, MassBrowser uses single-hop proxies for its connections (for the majority of the users who do not demand anonymity), and restricts the use of proxies to censored content. Third, as discussed above, the number of proxies in MassBrowser scales with the number of its users.

4) User-adjustable privacy: Existing circumvention solutions either provide weak privacy on all connections (e.g., as in VPNs, Lantern, Psiphon) or provide strong privacy on all connections and for all users (e.g., as in Tor). By contrast, MassBrowser allows its users to adjust their privacy protection for different connections based on their needs, offering a more usable privacy-QoS tradeoff. Specifically, by default, MassBrowser connections are established through one-hop (fast) MassBrowser relays; however, a client can choose to tunnel some or all of her connections through MassBrowser's Tor interface. MassBrowser's Tor interface is run by some Buddies that act as Tor bridges [14]. Therefore, MassBrowser's Buddy software can be used as a pluggable transport [57] by Tor bridges. We evaluate MassBrowser's cost of operation when used as a Tor pluggable transport, showing that it is *drastically cheaper* than meek [45], while both offering similar blocking resistance properties (both meek and MassBrowser aim at increasing the censors' collateral damage by making use of shared IP addresses). We will provide a more detailed privacy discussion in Section VII.

5) Deployment feasibility: Unlike approaches like decoy routing systems [32], [84], [38] and tunneling systems [33], [35], [44], MassBrowser does not require cooperation/deployment from third-party Internet operators. Also, while MassBrowser's Operator is hosted as a domain-fronted service, it can be deployed using any low-bandwidth, high-latency covert communication mechanism [35], [33] if domain fronting is widely disabled [27], [3]. Our system is currently in beta release mode with user-friendly GUI software for both volunteers and clients.

IV. Technical Design Decisions

In this section, we discuss our design decisions aimed at designing MassBrowser as a reliable circumvention system

with practical a tradeoff between QoS, cost of operation, and user privacy.

A. Blocking Resistance

We use the following core techniques to provide strong blocking resistance in MassBrowser.

Use of shared, dynamic proxy IPs to resist IP enumeration: As MassBrowser proxies are run by normal Internet users (either censored clients or non-censored volunteers), blocking them is costly and prone to collateral damage. First, a typical Buddy will most likely have a NAT IP address, therefore sharing a public IP address with other users/services in the same network. For instance, a Buddy connecting from a coffee shop will share a public IP with other users in the area (we will describe how MassBrowser enables connections despite NAT). Additionally, a typical Buddy will frequently change IP addresses, e.g., by moving across networks, amplifying the collateral damage. Second, the number of MassBrowser's relay IPs scales with the number of its users as it deploys client-to-client proxying. Also, we have employed various social engineering techniques to attract a large number of volunteer proxies.

Traffic Obfuscation and Encryption: All MassBrowser communications are encrypted to prevent deep-packet inspection. Specifically, MassBrowser deploys traffic obfuscation mechanisms to remove protocol fingerprints and prevent censors from detecting MassBrowser traffic based on traffic characteristics like packet timings and sizes.

Domain Fronting the Operator: MassBrowser's Operator runs as a domain fronted service [20]. As discussed earlier, a domain fronted service runs behind a network infrastructure with shared IPs (e.g., CDNs), therefore blocking it will cause significant collateral damage to the censors. Although domain fronting is a relatively expensive technique, the costs of domain fronting MassBrowser's Operator is very low due to the small volume of the control traffic generated by the Operator, as shown in Section VI-C. Note that while MassBrowser's Operator is hosted as a domain-fronted service, it can be deployed using *any* low-bandwidth, high-latency covert communication mechanism [35], [33] if domain fronting is widely disabled in the wild [27], [3]. Furthermore, with the ongoing adoption of TLS1.3 [59] and encrypted SNI [60] by CDNs (e.g., Cloudflare and Google CDN), domain fronting can be performed with no need to modifying the SNI field of TLS connections (therefore it can not be disabled by CDN providers who support encrypted SNI).

B. Optimizing Cost and QoS

As discussed earlier in Section III, blocking resistant circumvention systems suffer from either low QoS or high cost of operation (or both). We argue that the main reason for the poor QoS/high cost of existing circumvention systems is the extreme disproportion between available proxying throughput and the bandwidth demand from censored clients. We therefore take the following two complimentary approaches to alleviate such disproportion.

Optimizing load on proxies through selective proxying: We use the following techniques to minimize the traffic load on MassBrowser proxies.

a) Whitelisting censored content only: Existing circumvention tools like Tor and VPNs tunnel *all* network traffic of a censored client through circumvention proxy, including censored and non-censored content. This is done in Tor to provide anonymity on all connections, but even non-anonymous tools like VPNs, Lantern, and Psiphon tunnel all traffic through circumvention proxies for the ease of operation. We believe that this is one of the key reasons constituting to high bandwidth pressure on in-the-wild circumvention proxies (causing their low QoS). We evaluated the list of top bandwidth-consuming domains provided to us by a major non-anonymous circumvention tool[2] for the day of Feb 21, 2008. Our evaluation finds that 48% of the proxied traffic belongs to websites that are *not* censored in Iran (total proxied traffic is 3.56 TB).

Tunneling non-censored content through a circumvention system not only puts additional burden on the proxies, it also lowers the quality of service for most of the non-censored websites, e.g., a Chinese user will have to access a (non-censored) China-based website through a US-based proxy, therefore increasing the latency. Basing our design on the SoP principle, we restrict the use of MassBrowser Buddies to censored-content only. Therefore, our Client software only proxies censored content through Buddies and retrieves non-censored content directly with no proxy, and the Buddies deploy whitelists to proxy only censored content. Needless to say, a MassBrowser client can divert her privacy-sensitive (but not censored) connections to MassBrowser's Tor interface.

b) CacheBrowsing: MassBrowser uses a recent circumvention technique called CacheBrowsing [30], [85] to further minimize the load on the proxies. In CacheBrowsing, a client directly fetches a censored object hosted on CDN from the hosting CDN's edge servers, without using proxies. However, a limitation of CacheBrowsing is that it can only retrieve censored content hosted on a CDN and accessible through HTTPS,[3] therefore it can not be used as a standalone circumvention system. We integrate CacheBrowsing into MassBrowser's client software. That is, a MassBrowser client will fetch the CDN-hosted censored content directly from CDNs using the CacheBrowsing technique, and only use MassBrowser Buddies for the censored content not hosted on CDNs. Based on our analysis, this saves 41% of bandwidth on the Buddies for Alexa top 1000 websites.

c) Strategic proxy assignment: MassBrowser's Operator is responsible for matchmaking Clients and Buddies. The Operator considers various factors, including the regions, bandwidths, and the preferences of the clients and relays in pairing them together.

Incentivizing volunteer proxies: We use the following approaches to increase the number of volunteer proxies. We envision a large fraction of MassBrowser Buddies to be from typical Internet users with little technical background. We therefore design a GUI-based client software for Buddies to

[2]We do not disclose their identity per their request.
[3]We call such webpages CacheBrowsable.

offer a user-friendly experience, transparency, and full, fine-grained control over what they proxy. Our Buddy GUI offers the following features.

a) Imperceptible operation: Our Buddy GUI runs imperceptibly and does not interfere with the volunteer's normal activities. The volunteer user will only need to perform a one-time installation and setup of the relay software, and may then let it operate until she needs to adjust her preferences.

b) Transparency on usage: Our Buddy GUI offers the volunteer with information on how the proxy is being used.

c) Enable relays to limit proxied bandwidth: The Buddy software enables a volunteer Buddy operator to specify how much bandwidth she is willing to donate to MassBrowser. Even a small donated bandwidth can help MassBrowser clients due to the bandwidth minimization mechanisms discussed above.

d) Enable relays to whitelist destinations: Our MassBrowser relay software enables a volunteer to proxy traffic only to Internet destinations she is comfortable with. A major set-back for volunteers is the potential legal consequences of relaying traffic to controversial destinations (such as those experienced by Tor exit relay operators [4], [6]). In MassBrowser, relays whitelist the categories of destinations they are willing to proxy traffic to, e.g., a relay can decide to relay traffic only to news websites or scientific websites.

e) Optional financial incentives: Future versions of MassBrowser may incorporate financial incentives for volunteers, either as the form of a service like Bitcoin mining by clients, or monetary compensation. We leave the investigation of incorporating such economic incentives with MassBrowser to future work.

Client to client proxying: As discussed earlier in Section III-C, censored users in various regions are subject to different censorship blacklists. Therefore, we leverage censored clients to proxy traffic for other censored clients in different regions, which improves the overall QoS of the connections by scaling the number of proxies with the number of clients. The matchmaking between the clients is performed by the Operator based on client preferences, locations, and bandwidth resources. Note that in our system, a client is able to opt out of being used as a proxy for other clients. To encourage client participation, we only allow oped-in Clients to benefit from other Clients as proxies.

C. User-adjustable privacy

We have integrated a Tor interface in MassBrowser Client and Buddy software. This enables MassBrowser clients to divert their privacy-sensitive connections to this interface, while performing their non-sensitive communications through MassBrowser's regular network (expectedly, with much better QoS). MassBrowser's Tor interface is run by volunteer Buddies who opt in to serve as Tor bridges [14] for MassBrowser Clients. We envision that existing Tor bridges can also add support for MassBrowser, as a new type of pluggable transport [57]. We will provide a more detailed privacy discussion in Section VII.

V. MassBrowser's Implementation

In this section, we discuss the implementation of MassBrowser's key technical components. We present further implementation details (including our backend services, browser bundle, GUI software, etc.) in Appendix A.

A. Connecting Users Behind NAT

As MassBrowser Clients and Buddies are regular Internet users, most of them will likely be connecting to the Internet using NATed IP addresses. Therefore, an important challenge to MassBrowser's operation is enabling communication between NATed Clients and Buddies, i.e., MassBrowser needs to deploy *NAT traversal* techniques [80], [43]. Typical NAT traversal techniques, however, may not be applicable for all transport protocols depending on the type of a peer's NAT, i.e., depending on how the underlying NAT maps local IPs to public IPs. Matthews et al. [43] perform a thorough analysis of different NAT deployments in the Internet and how NAT traversal techniques may apply to them. We categorize MassBrowser peers (i.e., Clients and Buddies) into three categories based on the type of their NATs.

TCP Reachable: These are the peers with whom it is possible to initiate a TCP connection, either directly or via some existing NAT traversal technique.

UDP Reachable: For such peers, we are not able to initiate TCP connections, but are still able to send UDP packets to them via some NAT traversal technique. These peers reside behind *Restricted NATs* as defined by Wing et al. [80].

Unreachable: Such peers are located behind NATs that prevent the use of *any* NAT traversal technique. Wing et al. [80] classify these NATs as *Symmetric NATs*.

MassBrowser's Operator serves as a STUN server to discover the NAT type of each peer. The Operator then uses the discovered NAT type of the peers to match Clients and Buddies, and to decide which party should initiate the connection, as shown in Table II. For any pair of a Client and a Buddy, they can communicate if at least one of them is reachable from behind NAT. As can be seen, when both of the peers are reachable, the Client initiates the connection. When both peers are UDP reachable, MassBrowser's software tunnels a TCP connection through an established UDP tunnel. If none of the peers are reachable, a MassBrowser connection can not be established between these peers, and therefore the Operator will *not* map an unreachable Client to an unreachable Buddy.

Note that MassBrowser's Operator *does not* deploy a TURN server; a TURN server will need to proxy the connections between (unreachable) Clients and Buddies, which is significantly expensive and bandwidth-extensive for a free circumvention system like ours. Additionally, a circumvention TURN server can easily get blocked by the censors unless it is deployed as a (prohibitively expensive) domain fronted service.

B. Assigning Buddies to Clients by the Operator

The Operator is in charge of coordinating Client and Buddy communications and providing Clients with online Buddies to

TABLE II. CONNECTION INITIATION FOR A MATCHED PAIR OF CLIENTS AND BUDDIES. IF BOTH THE CLIENT AND BUDDY ARE UNREACHABLE, THE OPERATOR WILL NOT MATCH THEM TOGETHER.

		Buddy		
		TCP-Reach	UDP-Reach	Unreach
Client	TCP-Reach	Client	Buddy	Buddy
	UDP-Reach	Client	Client	Buddy
	Unreach	Client	Client	✗

use as relays. The Operator assigns Buddies to Clients with the following considerations.

Buddy destination whitelists: Buddies can whitelist destinations they are willing to proxy traffic to based on their content types. The Operator actively maintains Buddy whitelist preferences. When a Client queries the Operator for new Buddies, the Operator will respond with Buddies that allow the intended destinations in their whitelists.

Buddy loads: The MassBrowser system is a heterogeneous network composed of machines with varying processing powers and network bandwidths. The Operator approximates a Buddy's available throughput based on the bandwidth limit set by the Buddy owner, the number of active Clients assigned to that Buddy, Buddy's reliability over time. This is used to balance the load on Buddies when assigning Buddies to new Clients.

Parties' NAT types: The Operator also considers the NAT types of the peers in matching Clients and Buddies, as described above.

Sybil attack protection: As discussed in Section VII, a censor can *not* block the Buddies that she obtains from the Operator, nor can she identify their clients (since Buddy IPs are NATed). However, a resourceful censor may overload the identified Buddies in order to consume their circumvention capacity (i.e., DoS the Buddies). Note that this will be a costly DoS attack due to the symmetry between the load on the attacker and the target. Nonetheless, our Operator can deploy standard Sybil protection mechanisms against such an expensive DoS attack. We have particularly implemented the Sybil protection mechanism of Nasr et al. [48], which uses a strategic reputation system to pair clients and proxies.

C. Selective Proxying Through Whitelisting

As discussed earlier in Section IV-B, MassBrowser deploys *selective proxying* to optimize the load on the Buddies as well as to enable them to enforce their proxying preferences. Specifically, MassBrowser inspects every network request *individually* to decide how to best handle that request; this is in contrast to standard circumvention systems that naively proxy *everything* through their circumvention proxies. To perform such per-request proxying, MassBrowser creates a *content whitelist*, which specifies how different content objects should be handled. The whitelist is maintained by the Operator and is regularly synced by all of the Buddies and Clients.

Creating the Whitelist: We create our list of censored domains using the data released by several recent censorship measurement studies, in particular, the IClab [51] and GreatFire [28], that have assembled lists of censored domains for different censorship areas (note that we exclude controversial censored webpages, such as adult content, from our list to preemptively protect our Buddies; such objects are fetched through Tor, as explained later). For each of the censored domains in our list, we identify all of its web objects and create rules for fetching each of those objects. This is done using our automated technique that crawls each censored website using the Chrome web browser and follows all of its inner links (with a depth 3), then creates regular expressions to identify all the corresponding domains/subdomains of that censored domain. For each web object, our crawler creates one of the following three rules:

- If the object is not censored, it will be assigned a `NonCensored` tag; such objects will be fetched directly through the Internet (without using a Buddy).
- If the object is censored, our crawler will check if it can be browsed through CacheBrowser [30], [85]. If yes, it will be assigned a `CacheBrowsable` tag, and it will create rules on how to fetch it from CDNs (we use the code from prior work [30], [85]).
- Finally, if the object is censored but not CacheBrowsable, it will be assigned a `MassBrowsable` tag; such objects will be fetched through a Buddy.

We perform a similar whitelisting mechanism for non-web objects that are generated by other applications. For instance, we have created rules for Telegram and Tor traffic based on the IP addresses of Telegram servers and public Tor relays.

Enforcing the Whitelist: Figure 4 demonstrates the selective proxying process performed by a MassBrowser Client for each requested object, based on the content whitelist. Note that, in order to enforce such whitelisting policies, the Client's web browser delegates DNS resolution to MassBrowser's Client software (this requires disabling the browser's DNS caching); therefore, proxy destinations must be hostnames, not IP addresses. This enables distinguishing requests to different hosts that resolve to same (shared) IP addresses. For every web request by a user, the Client software looks up the requested destination hostname in the whitelist, identifies which website the hostname belongs to, identifies whether the website is censored, and determines the content types associated with that website. If the request is censored and the Client already has an open session with a Buddy that supports the required content's type, it will use the existing connection to proxy that request. Otherwise, the Client will be assigned a Buddy who has whitelisted the category of the requested content.

Maintaining the Whitelist: Note that the Operator regularly updates its whitelists by adding/updating rules for censored websites (e.g., based on the most recent measurement data). The Operator regularly (once a day) pushes the changes of the whitelist to all Clients and Buddies, which they incorporate into their local whitelist databases.

Enforcing Buddy Preferences: Each whitelisted domain is assigned a content category as shown in Table III. Each Buddy is able to choose what categories of censored content she is willing to proxy to (as shown in Figure 5). If a Buddy disallows a category, it will not proxy any websites from that category. As explained above, a Buddy is in charge of performing DNS resolution for its Client requests; therefore the Buddy will be able to ensure that the Client is not violating the Buddy's

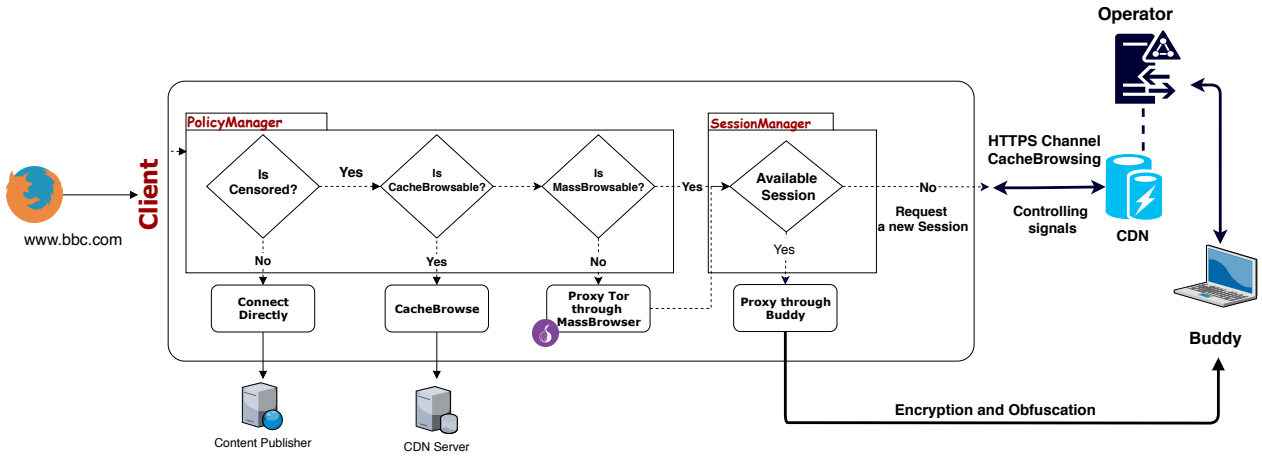

Fig. 4. Optimizing proxying load by a MassBrowser Client through selective proxying. The Client will check each requested object against the content whitelist, as introduced in Section V-C, to decide how to handle the requested object. If the requested object is non-censored, it will be obtained directly; if it is CacheBrowsable, it will be obtained from the corresponding CDN servers; otherwise, the object will be requested through a Buddy (if it is tagged as MassBrowsable), or sent through Tor if no Buddy supports that censored object.

destination restrictions.

Note that to preemptively protect our Buddies, we have excluded controversial webpages/categories (such as adult content) from the whitelist of Buddies. As shown in Figure 4, for a requested domain that is not whitelisted, MassBrowser will request it through a generic circumvention system; our current design sends such connections through Tor. Note that in such a setting, MassBrowser will act as a Tor bridge [14], so it is not impacted by the potential censorship of Tor.

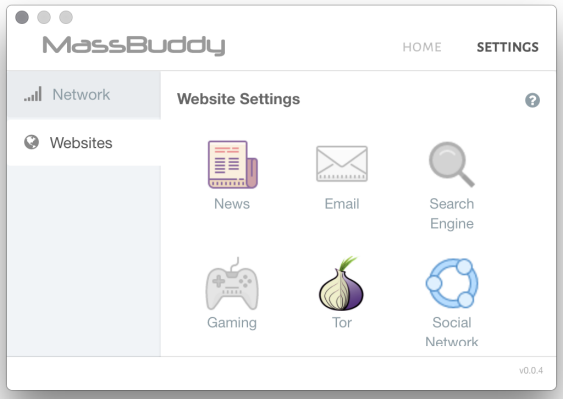

Fig. 5. The settings page in the MassBrowser Buddy software allowing the user to select it's allowed content types

The Complexity of Whitelisting: We store the whitelists as regular expressions in a hashmap data structure in which each key consists of the two right-most part of the domain name, and its corresponding value is the regular expression. Whenever a new connection arrives, the client extracts the two right-most domain name of that connection, and checks against the whitelist. If it is not found, it is considered to be non-censored. If the domain is in the list, the client will check if the whole domain matches the corresponding regular expression,

TABLE III. CONTENT CATEGORIES IN MASSBROWSER

Category	Example
News	CNN, BBC, ...
Email	Gmail, Ymail, ...
Search Engine	Google, Bing, ...
Gaming	Steam, Blizzard, ...
Social Network	Facebook, Twitter, ...
Sharing Platforms	Dropbox, Flicker, ...
Messaging	Telegram, Whatsapp, ...
Tor	Tor

which has a time complexity of $\mathcal{O}(s)$, where s is the length of the URL. Therefore, the worst case time complexity of our whitelisting method is $\mathcal{O}(s)$, but s is small for most domains.

D. Encryption and Traffic Obfuscation

In MassBrowser all of the communication between Clients and Buddies are encrypted in order to resist DPI attacks deployed by the censors. A matched pair of Client-Buddy encrypt their messages using a symmetric cipher with a shared secret key that they share through the Operator. Our implementation currently uses AES 256 for Client-to-Buddy encryption.

We also implement traffic obfuscation to protect MassBrowser's traffic against traffic analysis attacks [31], [24], [76]. Particularly, we have built a custom implementation of the *obfsproxy* [53] Tor pluggable transport tailored to work with our MassBrowser implementation. The obfuscation algorithm removes identifiable traffic patterns, making the Client-Buddy protocol look like benign peer-to-peer traffic, e.g., p2p gaming or file sharing traffic.

E. Communication Sessions in MassBrowser

We define a MassBrowser *session* to be a connection between a Client and a Buddy. Upon receiving a request from the browser, the Client checks whether the request can be handled with any of the currently active sessions the Client has, i.e., whether any of the connected Buddies will accept the request in their whitelisted categories. If no such session is found, the Client will need to ask the Operator to assign it a new session with a suitable Buddy that will accept the request.

The Operator will select a Buddy to assign to the Client and will notify both parties to establish a new session. Each session has the following attributes:

1) *Allowed content types:* This is the list of content types that the Client is allowed to obtain through this session.
2) *Shared Keys and Cipher Suite:* All communications between the Client and Buddy are encrypted with a shared key and cipher suite shared through the Operator.
3) *Obfuscation method:* In order to prevent fingerprinting attacks on the Client-Buddy communication protocol, the Operator may instruct the users to use one of the available obfuscation algorithms if the censoring region is known to deploy DPI attacks.
4) *Connection initiator:* Based on NAT type of the peers, the Operator will instruct one of the users to initiate the connection with the other using an appropriate NAT traversal technique, as described earlier.
5) *Expiration time:* Each session is only valid within a defined time period. The Client will have to ask to renew the session if he wishes to continue using it beyond the expiration time. This is to perform load balancing on Buddies over time.

The Operator will send the details of each new session to the corresponding Client and Buddy. The party who has been selected as the connection initiator will then attempt to establish a connection with the other party. The receiving party will keep the session in a list of pending sessions until either the connection is established or the session expires. Each session can only be used once, and both parties will notify the Operator once the session connection has been established.

Timeline of events: Figures 6, 7, and 8 depict the timelines of communications between Buddies, Clients, and the Operator. As shown in Figure 6, when a Buddy starts, it sends a request to a STUN server to get its remote port and IP address. Then, Buddy updates the Operator with its new IP address and port number. Next, the Operator checks if the Buddy is reachable (either using TCP or UDP). The Buddy regularly repeats these steps to make sure that the Operator is up to date. Therefore, the Operator keeps track of all of Buddies and their reachability status.

Figure 7 illustrates the procedure for requesting a new session by a Client through the Operator. Each Client preemptively requests for sessions to reduce the overall waiting times for session creation.

Finally, Figure 8 summarizes the overall timeline of communications between MassBrowser parties. When an application (e.g., a browser) sends a new request to MassBrowser, the client will check if it has a session for that request (assuming the request is for a censored content), and if so, the Client starts using that specific session for the request. If not, the Client will requests a new session for that specific content.

VI. Performance Evaluation

A. System Performance

We evaluated the performance of MassBrowser's different phases of operation. We used our own MassBrowser clients for the measurement due to ethical reasons. Also, as mentioned earlier, while the Operator uses domain fronting, it can be

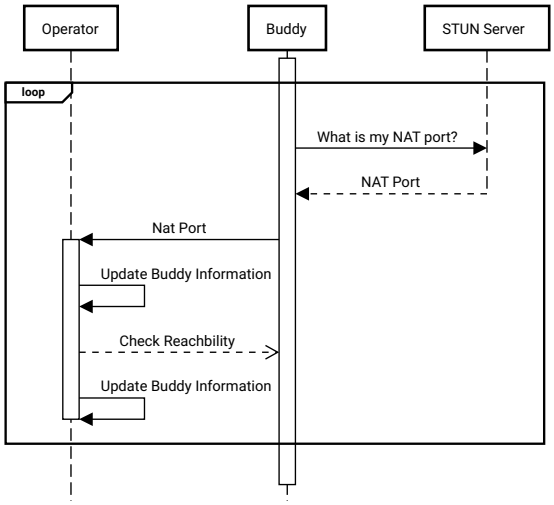

Fig. 6. Each Buddy sends status updates to the Operator to keep it updated on the Buddy's connectivity. The Buddy regularly contacts some STUN servers to check if it is behind NAT and to obtain its public ports. Then, it will send its public port information to the Operator. Also, the Operator regularly checks the reachability of each of the Buddies using their public port numbers.

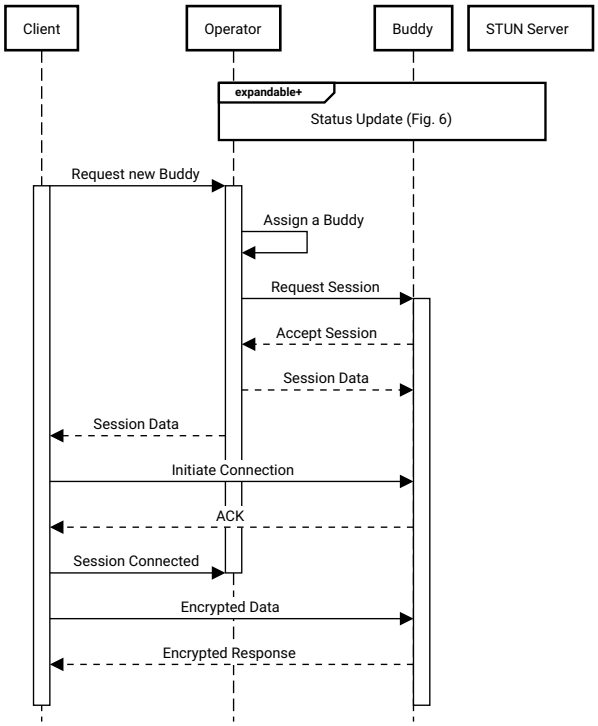

Fig. 7. Session creation procedure by a Client. When a Client decides to create a new session for a requested object, it will contact the Operator to ask for a Buddy that supports the requested object. The Operator assigns a Buddy using its relay assignment mechanism in Section V-B (e.g., based on the preferences of Buddies). Next, the Operator contacts the Buddy and requests a new session, which the Buddy can reject or accept the new session. If the selected Buddy rejects the session (e.g., is unreachable), the Operator will assign another Buddy. Once a Buddy accepts the session, the Operator will create a "session data" token that includes the cryptographic keys and protocol specifications for that session; the session data token is sent to the corresponding Client and Buddy. Finally, the Client initiates a connection to the assigned Buddy using the information in the session data token.

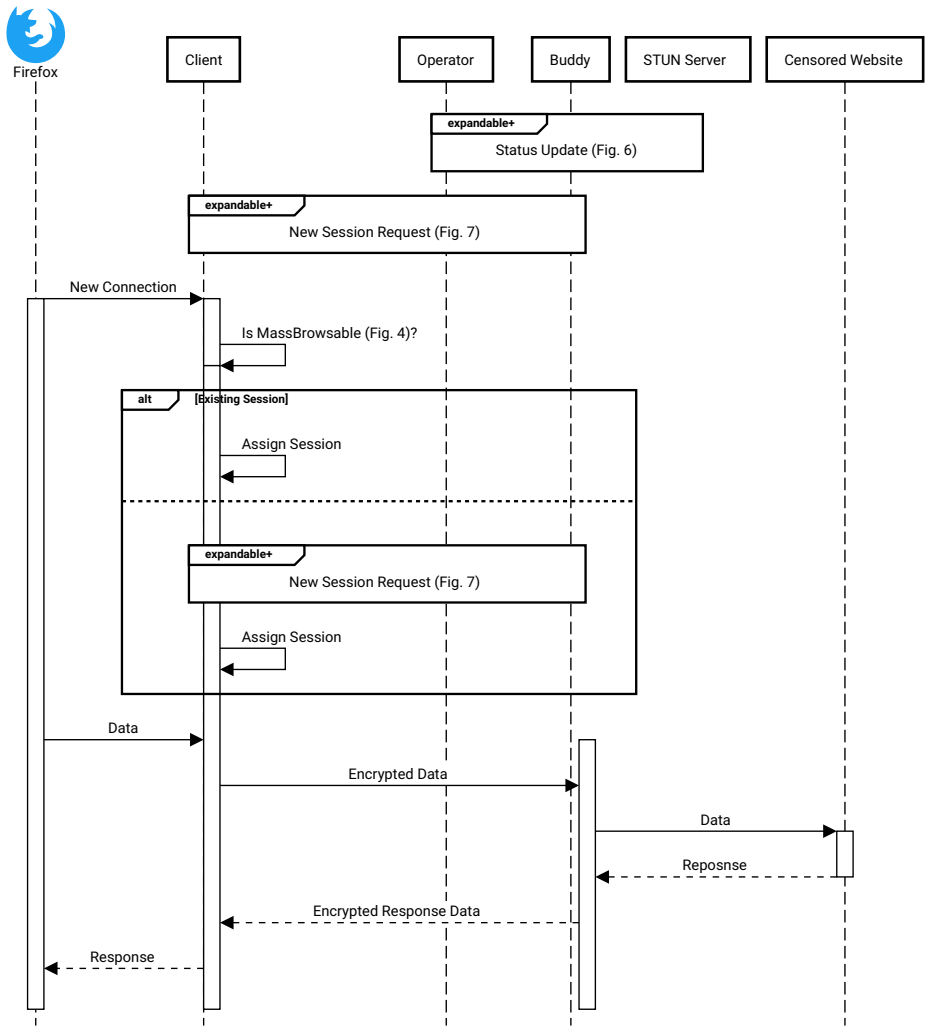

Fig. 8. A typical MassBrowser communication session between a Client and a Buddy. Upon receiving an object request (from a web browser in this example), the Client will check against the content whitelist (Section V-C) to decide how to handle that object. If the requested object is MassBrowsable (as defined in Section V-C), the Client will check if there already exists a session that can be used to fetch that object, otherwise she will request the Operator for a new session (Figure 7). ("alt" refers to the different alternatives)

deployed using any high-latency boostrapping mechanism. To demonstrate that, we artificially increase the latency of the Operator's responses in our performance measurement.

Software Boot Up: Whenever a user starts the Client, it connects to the Operator to get the latest whitelisting rules and updates. We ran 100 Clients and measured the average boot up time. The average boot up latency for the very first run of Client is about 20 seconds (with a standard deviation 5 seconds), while future boot ups takes an average of 4 seconds (with a standard deviation of 0.5 second). To demonstrate the possibility of deploying the Operator using high-latency channels (e.g., SWEET [35]), we artificially increased the response time of the Operator to 5 minutes. And expected, this slowed down the client's start up accordingly. However, it did not break nor interfered with the operation of the Client software.

Session Creation: A key phase of MassBrowser's operation is the session creation process. Each Client will preemptively create a general session, but whenever the client software receives a connection to a destination not supported by the existing session, the client will create a new session. In our measurements, each session creation takes about 0.7 second on average (with a standard deviation of 0.07 second). To demonstrate the possibility of using a high-latency channel for session creation, we artificially increased the response time of Operator to 1 minute. Even with such a large latency, Client is able to create sessions without breaking down.

Data Communication: MassBrowser uses a custom protocol over TCP/UDP for the communications between Clients and Buddies. We measure the throughput of this protocol using our own Client and Buddy machines, connected through a 1Gbps network: the throughput is 30MB/s over TCP and 10MB/s over UDP.

Operator's Capacity: Our current implementation of Operator is deployed on AWS using a 't2.large' instance and a 't2.medium' instance. We used the Apache HTTP server

benchmarking tool[4] to evaluate our current (light-weight) deployment of Operator against 1,000,000 clients (100 concurrent clients), showing an average response time of 400ms without any failed connections.

B. Buddy Bandwidth Contribution

Our analysis of the top 1000 Alexa website homepages [2] finds the average size of each webpage to be 2.4 MB. We found 41% of the generated traffic by these pages to be CacheBrowsable, which is very promising for MassBrowser regarding load optimization (note that most of the CacheBrowsable webpages are partially CacheBrowsable [85], therefore MassBrowser needs to proxy only the non- CacheBrowsable components). Therefore, in order to load a typical page through MassBrowser the client will only need to proxy an estimated 1.4 MB through the Buddies. The Akamai State of the Internet Connectivity Report [1] estimates the Internet bandwidth of an average user living in the United States in 2017 to be 18.7 Mbps. Assuming volunteers will provide MassBrowser with 25% of their unused bandwidth, an average Buddy in the United States will contribute 4.7 Mbps when not using the Internet, which translates into a page load every 2.5 seconds. Also, recall that in MassBrowser, the bandwidth of Buddies is solely used for loading censored content.

C. Costs of Operation

Ensuring low operational cost is one of the primary design goals of MassBrowser. The (bulky) circumvention traffic of MassBrowser clients is handled by volunteer Buddies. Therefore, the only operational cost of MassBrowser is imposed by running the Operator. Recall that the Operator is deployed as a domain fronted service, i.e., hosted on a CDN, in order to allow unblockable access to the censored users. In this section, we show that while domain fronting is known to be prohibitively expensive for proxying [46], it imposes little costs on MassBrowser as it is only used for its control traffic.

There are three factors that contribute to the Operator's operational costs:

1) Number of Client-Requested Sessions Per Day: Each session established between a Client and a Buddy is capable of serving any volume of traffic to different destinations as long as they satisfy the content type restrictions imposed by the Buddy. Therefore, it is unlikely that a Client will require more than a few active sessions at any given time. Our evaluation of a typical Client shows that 20 sessions per day is sufficient for typical web browsing.

2) Size of Session Objects: Upon creation of a new session between a Client and a Buddy, the Operator will need to exchange some protocol messages to the two parties. The exchanged information is composed of a 500 byte fixed-size segment containing details about the IP addresses, ports, NAT types, connection initiator, secret key, and the session expiration date, along with a variable-size segment listing the content types that will be accepted on the session (each content type takes 12 bytes). Therefore, the overall traffic load on Operator for each session is ≈ 1000 bytes.

[4] https://httpd.apache.org/docs/2.4/programs/ab.html

3) Size of the Webpage Database: The Operator maintains a database containing information on how to browse different censored websites supported by the Buddies. While the number of such unique domains for every website could be high, the database stores the domains in regex format, combining groups of similar domains with identical censorship information into single entries. The majority of the websites have at most 50 entries in Operator's database; given that each entry is around 1KB, each website will use at most 50KB in the database.

Based on these factors, we estimate Operator's operational costs, which is hosted over the Amazon AWS.

Cost of Running the Operator Servers: We estimated every user to request 20 sessions per day. For 10M users this requires 200M requests which would amount to an average of 2000 requests per second. Four AWS EC2 *a1.metal* instances, costing at about $0.408 an hour (at the time of writing), will be sufficient for handling this load of requests generated by 10,000,000 users. *The monthly cost will amount to $0.00011 per user.*

Cost of Deploying on CDNs: We have hosted the Operator on the Amazon Cloudfront CDN. Amazon Cloudfront charges based on the volume of traffic, and the locations of the CDN edge servers used. Note that Operator's communications with Clients are *not* latency sensitive; therefore, it suffices for the Operator to use a cheap CDN service (we use a service with $0.01 per GB). As estimated above, each user will request 600 sessions per month, for which the Operator will need to send 600 KB of control data to the Clients; this costs *$0.00006 per user each month*. The user will also need to synchronize her local database with Operator, resulting in a one-time 50 KB data transfer for each supported website, which costs $0.0000005 per user for every website.

Comparing costs with meek: Meek [45] is a Tor pluggable transport that relays Tor traffic through domain fronted proxies to evade censorship. In order to operate, meek must proxy all of the users' traffic through CDN servers. As a result, unlike MassBrowser the costs of operating meek is proportional to the client's bandwidth usage. As we saw in the previous analysis, we estimate the cost for a MassBrowser user with 600 sessions per month to be $0.00006 each month using Amazon Cloudfront CDN, *regardless of the types of the websites browsed* (e.g., video streaming, news, etc.). If we assume each session to be just for one website load and each website to have an average of 2.4 MB (as we measured), then the same client using meek over Amazon Cloudfront CDN will cost $600 * 0.0024 * 0.01 = \$0.014$, which is over *200 times* the cost of the user on MassBrowser. Note that in real life each session will be used to browse multiple websites and may require higher traffic (e.g., for video streaming), therefore, the cost gap will be even greater in favor of MassBrowser.

D. MassBrowser as a Tor Transport

As mentioned before, MassBrowser can be used as a Tor pluggable transport, i.e., a Client who needs anonymity can connect to a Buddy who whitelists Tor traffic. We measured the time to load the top 100 Alexa websites with Tor, using MassBrowser as a bridge for Tor, and using MassBrowser without Tor (experiments were performed from the same vanatge points

TABLE IV. AVERAGE PAGE LOAD LATENCIES FOR DIFFERENT WEBSITE OVER TOR, MASSBROWSER AS A TOR BRIDGE, AND MASSBROWSER ALONE.

Website	Tor (s)	MassBrowser + Tor (s)	MassBrowser (s)
Google.com	19.6	20.3	2.6
Youtube.com	27.3	25.6	6.3
Facebook.com	27.4	30.4	6.6
Baidu.com	7.5	10.1	1.7
Wikipedia.com	29.5	22.3	1.1

in Iran). We browsed each website 50 times over each setting and computed the average time to load the websites. Table IV presents the load times for different websites. On average *loading each website on Tor takes more than 16 seconds longer than using MassBrowser*. Using MassBrowser as a Tor bridge does not significantly change the load times compared to using Tor with no pluggable transport; therefore, *MassBrowser's added latency on Tor is negligible*, making MassBrowser a suitable plug for Tor bridges.

E. Effect of Churn

We expect MassBrowser to be resistant to Buddy churn. This is because each Buddy runs a stand-alone software application (as opposed to web sessions in systems like FlashProxy [19]), and therefore the churn times are in the order of several hours[5] (e.g., when the users restart) rather than minutes. Since the primary use of MassBrowser is web browsing, such long churn times will have little impact on short-lived HTTP/HTTPS sessions.

VII. DISCUSSION OF SECURITY AND PRIVACY GUARANTEES

In this section, we discuss the privacy guarantees of MassBrowser's components.

A. Client Privacy

A. Privacy against Buddies A MassBrowser Buddy imposes the same privacy threats to its Clients as a network observer, e.g., an ISP, on regular Internet users.

Anonymity against Buddies: As discussed earlier, providing client anonymity is not a design goal for MassBrowser based on the SoP principle. Therefore, a Buddy can learn the destinations being accessed by her connected Clients —this is similar to how a typical network observer (like an ISP or a transit AS) can learn browsing patterns of typical Internet users. Note that, like a normal Internet user, a MassBrowser client needing anonymity can use an anonymity system like Tor—through MassBrowser —(i.e., by connecting to Buddies that support Tor).

Confidentiality from Buddies: A Buddy will not be able to see its Clients' communication content for HTTPS destinations, which includes the majority of services hosting sensitive user data like social networking websites and search engines. A Buddy, however, will be able to see a Client's communication content to an HTTP destination, similar to how an ISP observes the HTTP traffic of its users. A MassBrowser Client can opt to use MassBrowser for HTTPS websites.

Surveillance by censor-run Buddies: A powerful organization that runs numerous Buddies for user surveillance is not different than a nation state or ISP wiretapping through Internet routers. Real-world observations over the years have shown that censoring governments tend to not penalize their users for the sole act of circumventing censorship. The risk is much less for MassBrowser Clients as, by design, MassBrowser Buddies do not allow connection to controversial websites with potential legal consequences (for such websites, the clients will need to use Tor through MassBrowser).

Identification by censors who know Buddies: The Buddies obtained by a censoring client from Operator can not be used to learn any information about the Clients who use these Buddies. This is because different Clients connecting to the same Buddy will make connections through different IP address and port combinations due to NAT.

B. Privacy against Operator Unlike traditional circumvention tools like Psiphon, Anonymizer, and Lantern, in MassBrowser the Operator of the circumvention system is separate from the proxying parties. Therefore, the Operator is not able to observe Client traffic. The Operator can only learn the categories of content a Client is willing to access.

B. Buddy Privacy

Privacy against Clients A Client using a Buddy will only learn the (ephemeral) *NATed* IP address of that Buddy, but no other information. As Client-Buddy assignments are performed by the Operator, a Client can not choose the Buddy to connect.

Privacy against Operator The Operator will have access to a Buddy's preferences such as her whitelisted content types and specified bandwidth limits. A Buddy's IP address will also be exposed to the Operator, however similar to the Clients, this is the *NAT* IP address of the Buddy, which is also visible to any other web service the Buddy connects to on the Internet.

C. Security and Reliability

Outright blocking of large IP domains If the censors are willing to disrupt unknown network protocols, they can block all suspected/unknown IP addresses, which will significantly impact the operation of any proxy-based circumvention system like MassBrowser.

Taking over the Operator Like other centralized circumvention systems, if the censors are able to take control of the Operator, the will be able to shut it down and learn the IP addresses of the active Clients and Buddies.

Blocking the Operator's IP In MassBrowser, the Operator has the same protection against IP blocking as domain fronting systems. That is, being run on public CDNs, the censors will have to block a whole CDN in order to block the Operator.

Blocking the Buddies' IP The IP enumeration techniques that censors practice against traditional circumvention systems like Tor [81], [16] will *not* work against MassBrowser Buddies. This is because the censors can only obtain the NAT IPs of

[5] https://www.statista.com/statistics/736727/worldwide-teen-average-online-time-devices/

the Buddies; blacklisting such IPs will have similar collateral damage as blocking domain fronting systems.

Traffic fingerprinting All MassBrowser communications between Clients and Buddies are obfuscated (and encrypted) using a tailored variant of obfsproxy [53] to prevent known traffic fingerprinting attacks [31], [24], [76]. The key difference of our tailored protocol with obfsproxy is that we omit the handshaking process of obfsproxy, since the Operator communicates the information needed to bootstrap the sessions (through the session data tokens in Figure 7) to the Clients and Buddies. Removing the handshaking process improves resilience to known fingerprinting attacks on obfsproxy [76]. All Client traffic to the Operator is protected with domain fronting.

DoS attacks through censor Sybils As discussed in Section IV-A, a censor who obtains a Buddy from the Operator can not block that Buddy, nor can he identify the Buddy's clients. However, a resourceful censor may overload the obtained Buddies to consume their available circumvention capacities. Note that such an attack is not a strong DoS attack, as the load on the attacker and victim is symmetric (asymmetry is the key property of real-world DoS attacks). Nonetheless, our Operator deploys standard Sybil protection mechanisms as explained earlier.

TLS interception by the censors If the censors are able to intercept the TLS communications, all circumvention systems including MassBrowser will be insecure.

Compromising a Client's local certificate As described before, each Client will use a locally created certificate to intercept and optimize MassBrowser connections. If an adversary is able to somehow obtain the private key of this certificate for a specific Client (e.g., through installing a malware on the target device), he will be able to intercept all connections or disrupt communications of that specific Client.

Faulty Buddies A large number of faulty/misbehaving Buddies will negatively impact the usability of MassBrowser to the clients. A Buddy can be faulty due to various reasons such as changes in firewall filters or NAT mappings that are not yet synced with the Operator, poor network connectivity, or being owned by malicious parties. To prevent this, our measurement backend periodically establishes connections through the Buddies to assess their reachability and remove faulty Buddies from the system.

VIII. Conclusions

In this paper, we presented the design and deployment of the MassBrowser censorship circumvention system. MassBrowser is a volunteer-run circumvention system, and its goal is to provide effective censorship circumvention to a large mass of censored users, with a high quality of service (QoS), low cost of operation, and adjustable privacy protection. Towards this, MassBrowser separates circumvention from privacy protection, allowing it to optimize the system around circumvention. MassBrowser has been deployed as a fully operational system with end-user GUI software for major operating systems, and it is currently available to early-adopters through invitations. The code and software are available online https://massbrowser.cs.umass.edu/.

Acknowledgements

We would like to thank our shepherd Carmela Troncoso and anonymous reviewers for their extremely helpful comments. We also would like to thank many people who provided feedback on this project since it started, in particular, Roger Dingledine, Philipp Winter, and Matthew Finkel (Tor project), Nicholas Merlino (UMass Amherst), Matthew Wright (RIT), and others who preferred to remain anonymous.

The MassBrowser project has been under active development since 2017. During this time, many people have supported the project by contributing to its code at different levels or by testing our software. Specially, we are extremely thankful to (and proud of) several UMass undergraduate students who helped the project by contributing to the development or testing of the project (but no substantial technical contributions); this includes (alphabetically) Derek Costigan (Undergraduate Honors Thesis, 2016), Joseph Lew (Undergraduate Honors Thesis, 2017), Matthew Hickey (Undergraduate Honors Thesis, 2019), Matthias Didong (Undergraduate exchange student, 2019), and Milo Cason-Snow (Undergraduate Honors Thesis, 2020).

The project has been generously supported by the National Science Foundation (NSF CAREER grant CNS-1553301) and the Open Technology Fund. Milad Nasr has been supported by a Google PhD Fellowship in Security and Privacy.

References

[1] Akamai, "State of the Internet Connectivity Report, Q1 2017," https://www.akamai.com/us/en/about/our-thinking/state-of-the-internet-report/global-state-of-the-internet-connectivity-reports.jsp.

[2] Alexa, "Top Websites," https://www.alexa.com/topsites.

[3] "Amazon threatens to suspend Signal's AWS account over censorship circumvention," https://signal.org/blog/looking-back-on-the-front/, May 2018.

[4] "Dark net raids were overblown by police," http://www.bbc.com/news/technology-29987379.

[5] M. Azali, "Infographic: Facebook Usage Statistics in Iran," http://techrasa.com/2017/08/16/infographic-facebook-usage-statistics-iran/.

[6] "Access Now and EFF Condemn the Arrest of Tor Node Operator Dmitry Bogatov in Russia," https://goo.gl/nNmP86.

[7] C. Brubaker, A. Houmansadr, and V. Shmatikov, "CloudTransport: Using Cloud Storage for Censorship-Resistant Networking," in *Privacy Enhancing Technologies*, 2014.

[8] S. Burnett, N. Feamster, and S. Vempala, "Chipping Away at Censorship Firewalls with User-Generated Content," in *USENIX Security*, 2010.

[9] "China's GitHub Censorship Dilemma," http://mobile.informationweek.com/80269/show/72e30386728f45f56b343ddfd0fdb119/.

[10] "Penalties For Using VPN In Various Countries," https://www.vpnunlimitedapp.com/blog/penalties-for-using-vpn/.

[11] "Everyone's Guide to By-Passing Internet Censorship for Citizens Worldwide," http://www.nartv.org/mirror/circ_guide.pdf.

[12] "Another Strike Against Domain Fronting," https://wills.co.tt/1746/another-strike-against-domain-fronting, February 2017.

[13] "Amazon Cloudfront CDN," https://aws.amazon.com/cloudfront.

[14] R. Dingledine and N. Mathewson, "Design of a Blocking-Resistant Anonymity System," https://svn.torproject.org/svn/projects/design-paper/blocking.html.

[15] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *USENIX Security*, 2004.

[16] "Ten Ways to Discover Tor Bridges," https://blog.torproject.org/blog/research-problems-ten-ways-discover-tor-bridges.

[17] "Django Web Framework," https://www.djangoproject.com/.

[18] "Electron Framework," https://electronjs.org/.

[19] D. Fifield, N. Hardison, J. Ellithorpe, E. Stark, D. Boneh, R. Dingledine, and P. Porras, "Evading censorship with browser-based proxies," in *Privacy Enhancing Technologies*. Springer, 2012.

[20] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant Communication through Domain Fronting," in *Privacy Enhancing Technologies*, 2015.

[21] "Where can I find an up to date list of free US proxies?" https://www.quora.com/Where-can-I-find-an-up-to-date-list-of-free-US-proxies.

[22] "FlashProxy," http://crypto.stanford.edu/flashproxy/.

[23] "Freedom on the Net 2017," https://freedomhouse.org/sites/default/files/FOTN_2017_Final.pdf, Freedom House, 2017.

[24] J. Geddes, M. Schuchard, and N. Hopper, "Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention," in *ACM CCS*, 2013.

[25] "Defeat Internet Censorship: Overview of Advanced Technologies and Products," http://www.internetfreedom.org/archive/Defeat_Internet_Censorship_White_Paper.pdf, 2007.

[26] "GoAgent proxy," https://code.google.com/p/goagent/.

[27] "Google disables "domain fronting" capability used to evade censors," https://arstechnica.com/information-technology/2018/04/google-disables-domain-fronting-capability-used-to-evade-censors/, 2018.

[28] "GreatFire Analyzer," https://en.greatfire.org/analyzer.

[29] B. Hahn, R. Nithyanand, P. Gill, and R. Johnson, "Games without frontiers: Investigating video games as a covert channel," in *European Security and Privacy (EuroS&P)*. IEEE, 2016.

[30] J. Holowczak and A. Houmansadr, "CacheBrowser: Bypassing Chinese Censorship without Proxies Using Cached Content," in *ACM CCS*, 2015.

[31] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The Parrot Is Dead: Observing Unobservable Network Communications," in *IEEE Security and Privacy (S&P)*, 2013.

[32] A. Houmansadr, G. Nguyen, M. Caesar, and N. Borisov, "Cirripede: Circumvention Infrastructure Using Router Redirection with Plausible Deniability," in *ACM CCS*, 2011.

[33] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer, "I Want My Voice to Be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention," in *NDSS*, 2013.

[34] A. Houmansadr, E. Wong, and V. Shmatikov, "No Direction Home: The True Cost of Routing Around Decoys," in *NDSS*, 2014.

[35] A. Houmansadr, W. Zhou, M. Caesar, and N. Borisov, "SWEET: Serving the Web by Exploiting Email Tunnels," in *Privacy Enhancing Technologies*, 2013.

[36] "Iran Reportedly Blocking Encrypted Internet Traffic," http://arstechnica.com/tech-policy/2012/02/iran-reportedly-blocking-encrypted-internet-traffic.

[37] "How Iran Censors The Internet," http://www.popsci.com/technology/article/2013-03/how-iran-censors-internet-infographic.

[38] J. Karlin, D. Ellard, A. Jackson, C. Jones, G. Lauer, D. Mankins, and W. Strayer, "Decoy Routing: Toward Unblockable Internet Communication," in *FOCI*, 2011.

[39] S. Khattak, T. Elahi, L. Simon, C. Swanson, S. Murdoch, and I. Goldberg, "SoK: Making sense of censorship resistance systems," *Privacy Enhancing Technologies*, 2016.

[40] "Lantern," https://getlantern.org/.

[41] C. Leberknight, M. Chiang, H. Poor, and F. Wong, "A Taxonomy of Internet Censorship and Anti-censorship," http://www.princeton.edu/~chiangm/anticensorship.pdf, 2010.

[42] M. Mahdian, "Fighting Censorship with Algorithms," in *Fun with Algorithms*, 2010.

[43] P. Matthews, R. Mahy, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," 2010.

[44] R. McPherson, A. Houmansadr, and V. Shmatikov, "CovertCast: Using Live Streaming to Evade Internet Censorship," in *Privacy Enhancing Technologies*, 2016.

[45] "meek Pluggable Transport," https://trac.torproject.org/projects/tor/wiki/doc/meek.

[46] "[tor-project] Summary of meek's costs, March 2017," https://lists.torproject.org/pipermail/tor-project/2017-April/001097.html.

[47] H. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "SkypeMorph: Protocol Obfuscation for Tor Bridges," in *ACM CCS*, 2012.

[48] M. Nasr, S. Farhang, A. Houmansadr, and J. Grossklags, "Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory." in *NDSS*, 2019.

[49] M. Nasr and A. Houmansadr, "GAME OF DECOYS: Optimal decoy routing through game theory," in *ACM CCS*. ACM, 2016.

[50] M. Nasr, H. Zolfaghari, and A. Houmansadr, "The Waterfall of Liberty: Decoy Routing Circumvention that Resists Routing Attacks," in *ACM CCS*, 2017.

[51] A. Niaki, S. Cho, Z. Weinberg, N. Hoang, A. Razaghpanah, N. Christin, and P. Gill, "ICLab: A Global, Longitudinal Internet Censorship Measurement Platform," *IEEE Security & Privacy*, 2020.

[52] D. Nobori and Y. Shinjo, "VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls," in *NSDI*, 2014.

[53] "A Simple Obfuscating Proxy," https://www.torproject.org/projects/obfsproxy.html.en.

[54] P. Pearce, R. Ensafi, F. Li, N. Feamster, and V. Paxson, "Toward Continual Measurement of Global Network-Level Censorship," *IEEE Security & Privacy*, vol. 16, no. 1, 2018.

[55] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson, "Global Measurement of DNS Manipulation," in *USENIX Security*, 2017.

[56] V. Perta, M. Barbera, G. Tyson, H. Haddadi, and A. Mei, "A glance through the VPN looking glass: IPv6 leakage and DNS hijacking in commercial VPN clients," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 1, 2015.

[57] "Tor: Pluggable Transports," https://www.torproject.org/docs/pluggable-transports.html.en.

[58] "Psiphon," http://psiphon.ca/.

[59] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3 (August 2018)," RFC 8446, Tech. Rep.

[60] E. Rescorla, K. Oku, N. Sullivan, and C. Wood, "Encrypted Server Name Indication for TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-esni-04, July 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-04

[61] "How Governments Have Tried to Block Tor," https://svn.torproject.org/svn/projects/presentations/slides-28c3.pdf.

[62] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper, "Routing around decoys," in *ACM CCS*, 2012.

[63] "SnowFlake Pluggable Transport," https://github.com/keroserene/snowflake.

[64] "Tor Metrics," https://metrics.torproject.org/.

[65] M. Tschantz, S. Afroz, V. Paxson *et al.*, "SoK: Towards Grounding Censorship Circumvention in Empiricism," in *IEEE Security and Privacy (S&P)*. IEEE, 2016.

[66] "Ultrasurf," http://www.ultrareach.com.

[67] "uProxy," https://www.uproxy.org/.

[68] "uProxy's Shadowsocks version," https://github.com/uProxy.

[69] "uProxy's WebRTC version," https://github.com/UWNetworksLab/uProxy-p2p.

[70] P. Vines and T. Kohno, "Rook: Using video games as a low-bandwidth censorship resistant communication platform," in *ACM Workshop on Privacy in the Electronic Society*. ACM, 2015.

[71] "How does your VPN speed Measure against other VPNs in China?" https://cc.greatfire.org/en.

[72] "Top 110 Free Proxy Sites – Best Free Proxy Servers List 2017," https://www.alltechbuzz.net/top-best-free-proxy-sites-servers-2016/.

[73] "SoftEther VPN User Forum: VPN Gate servers blocked in China," http://forum.vpngate.net/viewtopic.php?f=11&t=42498.

[74] "SoftEther VPN User Forum: Can Not Be Used In China," http://forum.vpngate.net/viewtopic.php?f=11&t=38298.

[75] "VPNGate: VPN Server List," http://www.vpngate.net/en/.

[76] L. Wang, K. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton, "Seeing Through Network-Protocol Obfuscation," in *ACM CCS*, 2015.

[77] Q. Wang, X. Gong, G. Nguyen, A. Houmansadr, and N. Borisov, "CensorSpoofer: Asymmetric Communication Using IP Spoofing for Censorship-Resistant Web Browsing," in *ACM CCS*, 2012.

[78] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "StegoTorus: A Camouflage Proxy for the Tor Anonymity System," in *ACM CCS*, 2012.

[79] T. Wilde, "Knock Knock Knockin' on Bridges' Doors," https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors, 2012.

[80] D. Wing, P. Matthews, R. Mahy, and J. Rosenberg, "Session traversal utilities for NAT (STUN)," 2008.

[81] P. Winter and S. Lindskog, "How the Great Firewall of China Is Blocking Tor," in *FOCI*, 2012.

[82] J. Wolff, "VPN Usage Around the World," https://cdn2.hubspot.net/hubfs/304927/Downloads/VPN-Usage-Around-the-World-Infographic.pdf, Global Web Index, 2017.

[83] ——, "The Internet Censor's Dilemma," http://tiny.cc/vfm0iz, March 2018.

[84] E. Wustrow, S. Wolchok, I. Goldberg, and J. Halderman, "Telex: Anticensorship in the Network Infrastructure," in *USENIX Security*, 2011.

[85] H. Zolfaghari and A. Houmansadr, "Practical censorship evasion leveraging content delivery networks," in *ACM CCS*, 2016.

APPENDIX

A. *MassBrowser's Implementation Details*

We have fully implemented MassBrowser as an end-user software, and it is currently in the beta release state with early-adopters from around the world evaluating it. Joining our system is currently invitation-based only, and we expect to open the project to the public soon (pending code review by a third-party organization). Our current implementation of MassBrowser supports Mac, Windows, and Linux operating systems. We have hosted an anonymized version of our code at https://massbrowser.cs.umass.edu. In the following we give details of our system implementation.

1) The Operator server: MassBrowser's Operator is a suite of backend services that runs various services essential to the reliable operation of MassBrowser, including strategic pairing of clients and proxies, monitoring the reachability and health of various parts of the system, and measuring the performance of MassBrowser in censored countries. We only use our own clients to perform our measurements. We have coded the Operator mostly in Python with the Django web framework [17] (approximately 10K lines of code). We have hosted our Operator server on Amazon CloudFront CDN [13], therefore it is a domain-fronted service and can not be blocked. Our Operator's API is accessible through both standard HTTP requests and WebSockets, though we refrain from using WebSocket connections for Clients in order to prevent introducing protocol fingerprints.

As previously mentioned, the Operator maintains a database of supported websites along with per-region censorship and CacheBrowsing information for all domains in the websites. To do so, the Operator has a *probing* component that regularly crawls the supported websites to identify domains and update its information.

Also, we have taken various measures to ensure the scalability and reliability of our backend services. In particular, we run multiple redundant servers for each component of Operator, and our 24/7 health monitoring system makes regular assessment of the status of system services, and sends email notifications in case of issues.

Finally, we have implemented a management console (with a snapshot in Figure 9) allowing us to manually monitor and configure different components of the system.

2) Buddy Software: We have coded our Buddy software in Javascript ES6 using NodeJS with a graphical user interface developed with the Electron framework [18] (approximately 50K lines of code). In addition to the GUI interface, our Buddy software is also available as a command-line application for expert volunteers. The Buddy actively maintains a WebSocket connection to the Operator, and will be notified of newly created sessions on this channel.

The Buddy software allows volunteers to have full transparency and control over their desired settings including bandwidth limits, destination whitelists and Client blacklists (Figure 5 displays a snapshot of a Buddy volunteer configuring her destination whitelists through the GUI). The Buddy software runs with minimal interference from the user. It is able to run in the background while providing an easily accessible switch for disabling the Buddy's activities on the users demand.

3) Client Software: We have implemented our Client software with NodeJS with an Electron based GUI (approximately 50K lines of code). A client application, e.g., a web browser, can connect to the Client software via a SOCKS proxy. On the first run, the Client software will walk the user through a setup wizard which will assist them in configuring their preferred browsers to use MassBrowser. The current implementation of Client software provides a setup wizard walking a client through connecting her web browser with MassBrowser. Figure 10 displays our Client setup wizard. The MassBrowser Client software requires to see each individual request, even when encrypted with TLS. In the normal case, the proxied TLS requests would not be visible to the Client software since it does not own the website certificates. To enable the interception of TLS connection by Client, the setup wizard adds a *locally created root certificate* to the client's browser during the initial setup. Note that the root certificate does not leave the client's computer, and therefore the client is secure as long as she does not share the certificate with others (Figure 10 shows how the user is informed during the setup). Client uses this certificate to "locally" man-in-the-middle MassBrowser's TLS connections to perform load

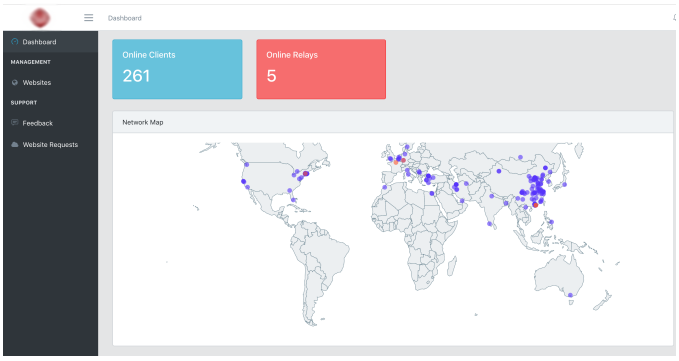

Fig. 9. Our management console

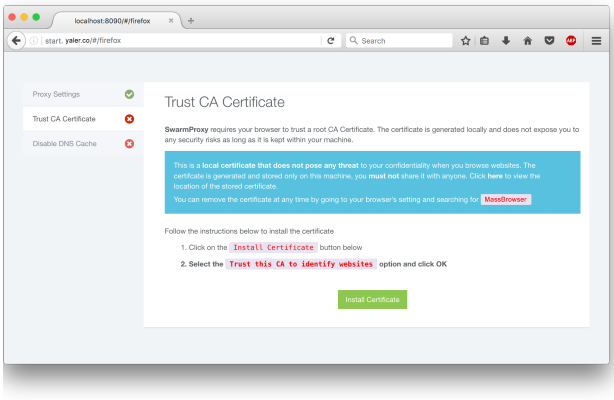

Fig. 10. The Client setup wizard to connect client's web browser to MassBrowser's client software.

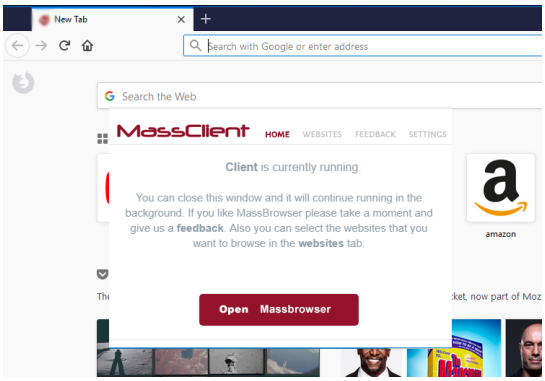

Fig. 11. The Client browser bundle comes with a pre-configured Firefox browser, which is ready to use out of the box.

optimizations like CacheBrowsing.

In addition to the client setup wizard, we have implemented a *client browser bundle* for clients. The bundle comes with a preconfigured, customized Firefox browser, and is ready to use out of the box. Figure 11 shows a snapshot of the browser bundle.