Incentivizing Censorship Measurements via Circumvention

Aqib Nisar, Aqsa Kashaf, Ihsan Ayyub Qazi, Zartash Afzal Uzmi*
LUMS, Pakistan

ABSTRACT

We present C-Saw, a system that measures Internet censorship by offering data-driven censorship circumvention to users. The adaptive circumvention capability of C-Saw incentivizes users to opt-in by offering small page load times (PLTs). As users crowdsource, the measurement data gets richer, offering greater insights into censorship mechanisms over a wider region, and in turn leading to even better circumvention capabilities. C-Saw incorporates user consent in its design by measuring only those URLs that a user actually visits. Using a cross-platform implementation of C-Saw, we show that it is effective at collecting and disseminating censorship measurements, selecting circumvention approaches, and optimizing user experience. C-Saw improves the average PLT by up to 48% and 63% over Lantern and Tor, respectively. We demonstrate the feasibility of a large-scale deployment of C-Saw with a pilot study.

CCS CONCEPTS

• **Social and professional topics** → **Censorship**; *Network access control*; • **Security and privacy** → *Human and societal aspects of security and privacy*; • **Networks** → *Network measurement*;

ACM Reference Format:
Aqib Nisar, Aqsa Kashaf, Ihsan Ayyub Qazi, Zartash Afzal Uzmi. 2018. Incentivizing Censorship Measurements via Circumvention. In *SIGCOMM '18: SIGCOMM 2018, August 20–25, 2018, Budapest, Hungary*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3230543.3230568

*Aqsa Kashaf is currently with CMU and Aqib Nisar with USC.

SIGCOMM '18, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230568

1 INTRODUCTION

Internet censorship has become increasingly pervasive with nearly 70 countries restricting Internet access to their citizens [38]. It can have a substantial impact on various stakeholders in the Internet ecosystem (e.g., users, content providers, and ISPs) [43] and thus, has drawn considerable interest from systems researchers towards building *censorship measurement systems* [26, 27, 37, 48–50]—that aim to ascertain *what* is blocked, *where* it is blocked, *how* it is blocked, and *when* it is blocked?—as well as *circumvention systems* that aim to bypass censorship [10, 19, 20, 29, 35].

Unfortunately, existing censorship measurement and circumvention systems are designed *independently*, which results in their individual designs to have limited capabilities. For example, building an effective circumvention system requires understanding the capabilities of censors (e.g., what is blocked and how?), which continually evolve over time [52]. However, existing circumvention systems (e.g., Tor [29], Lantern [10], and uProxy [19]) are not driven by such measurements and thus cannot adapt to the deployed blocking mechanism used by censors. This leads to *one-size-fits-all* solutions that are either ineffective (e.g., against some type of blocking) or inefficient, leading to high page load times (PLTs)[1] and thus degrading user experience [22, 24]. On the other hand, an effective censorship measurement system requires deployment of geographically distributed probe points—possibly volunteers—who find little to no incentive, beyond altruism, to help collect continuous measurements [37, 50].

In this paper, we call for *consolidating* censorship measurements and circumvention in a single platform to address the limitations of individual systems. To this end, we propose C-Saw, a system that gathers continuous censorship measurements through crowdsourcing and uses these measurements to offer data-driven circumvention by being *adaptive* to the type of blocking, leading to improved circumvention performance. This, in turn, incentivizes users to opt-in and contribute measurements. As more users crowdsource, the measurement data gets richer, enabling C-Saw to build a comprehensive database of blocked URLs along with the blocking mechanisms used by censoring ISPs. This allows C-Saw to offer even better circumvention capabilities for

[1] Akhoondi et al. [22] reported that Tor can increase latency by more than 5× compared to the direct path.

improving user experience and also opens up the possibility of crafting *new* circumvention techniques.

The design of C-Saw offers three key benefits: First, unlike existing measurement systems (e.g., [37, 50]), it provides *incentives* for users to opt-in by offering fast access to filtered content, which can facilitate large-scale deployment. Second, censorship measurement platforms (e.g., [26, 37, 58]) rely on target lists of URLs to test for censorship. However, relying on such lists can limit the *scope* of measurements because (i) URLs outside the list cannot be tested and (b) such lists are often inaccurate or only partially known [58]. C-Saw obviates the need for having *target lists* of URLs to test for censorship. Third, it enables *ethical* measurements by making it easy to obtain *informed user consent* [18, 30]. C-Saw achieves this property by measuring censorship for *only* those URLs that a user actually visits.

The C-Saw system comprises a *local database* for storing the blocking status of URLs that a user visits, a *global database* for storing crowdsourced measurements of *censored* URLs from all C-Saw users, and a *client-side proxy* that uses these measurements to provide adaptive circumvention via one or more methods (e.g., Tor and Domain Fronting [36]).

To facilitate *real-time* detection of content manipulation (e.g., substituting content or replacing a webpage with a *block page*, informing the user that the desired page cannot be accessed) by censors, C-Saw proxy issues *redundant requests* for URLs—one via the direct path and the others using one or more circumvention paths—and compares the received responses using a two-phase algorithm that achieves both fast detection as well as high accuracy. This also makes it easier for C-Saw to distinguish censorship from network problems (e.g., high delay or packet loss) without hurting user performance.

In designing C-Saw, we address several practical challenges. First, to limit the memory footprint of the database on client machines, we propose a URL aggregation scheme that uses the structure of URLs, their blocking status, and the blocking type to aggregate URLs. Second, clients on multihomed networks can experience degraded performance. We propose a mechanism that addresses this challenge by detecting multihoming and adapting the circumvention approach. Finally, to limit the impact of malicious clients on crowdsourced measurements, we present a simple voting mechanism and allow validation of URLs by individual clients.

Despite the benefits that C-Saw offers, combining censorship measurements and circumvention in a single platform brings about its own challenges with respect to security and user privacy. First, users who contribute censorship measurements may be identified by a censor, which may endanger users in repressive countries. To address this challenge, we anonymize users by carrying censorship reports over the Tor network and do not store any Personally Identifiable Information (PII) (e.g., IP addresses) that could lead back to the user. Second, users circumventing censorship may expose themselves to risks, especially in repressive regimes. To manage the potential risks, C-Saw allows consenting users to stay anonymous by using only those circumvention methods that provide anonymity.

We implement C-Saw, with all its features, using GitHub's electron framework [4] and carry out its evaluation in comparison with other circumvention systems. Our evaluation shows that C-Saw reduces the average PLT by up to 48% and 68% over Lantern and Tor, respectively. We carried out a real-world deployment of C-Saw and conducted a pilot study comprising 123 users. The collected measurements reveal blocking of CDN servers, which was not observed in earlier studies of censorship in Pakistan [43, 44]. We also found that for the majority of censored domains, a block page was returned. The second most common type of mechanism was DNS blocking. Finally, we show how C-Saw was able to effectively measure the recent blocking of Internet services, including Twitter and Instagram, from November 25-28, 2017 across different ASes in Pakistan.

Tradeoffs. The scope and extent of measurements collected by C-Saw depends on the population of C-Saw users. As a result, C-Saw may find it difficult to (a) measure unpopular websites and (b) measure censorship at timescales of *particular* interest to some stakeholders. These challenges can be addressed by allowing testing of specific target URLs [26]. However, in designing C-Saw, we take an explicit stance to work within a design space that makes it easy to obtain *informed user consent* without reducing the effectiveness of the system to measure censorship at scale [30]. We consider this to be an acceptable tradeoff.

Our work makes the following contributions[2]:

- We present a case study of distributed censorship that highlights opportunities for improving circumvention performance through censorship measurements (§2.3).
- We design (§3 and §4), implement (§6), and evaluate (§7) C-Saw, a system that combines censorship measurements and circumvention in a single platform.
- We conduct a pilot study, which shows that C-Saw can effectively measure the blocking mechanisms used by censors in the deployment region (§7.4).

2 BACKGROUND AND MOTIVATION

Internet censorship is enforced using a *centralized* or a *distributed* infrastructure. In the former case, all censored traffic

[2] An early workshop version of the paper made the case for C-Saw [45]. However, it did not provide a concrete system design, a practical implementation, evaluation, and a real-world deployment.

of the same type sees the same kind of blocking. For example, Iran and South Korea have been shown to exercise centralized censorship. In decentralized censorship, individual ISPs deploy filtering techniques independently and as a result, may block the same type of content differently. Several countries exercise distributed censorship including China, Vietnam, and UAE [15, 38].

2.1 Web Censorship Techniques

A censor has a variety of choices to carry out web filtering by intercepting a user request at various levels of the protocol stack. We now discuss some of the common censorship techniques employed by censors.

Web censorship can take place when a client performs an initial DNS lookup. At this point, a censor can manipulate DNS responses that will leave the client either without a resolution or a resolution that is not correct. A list of various DNS tampering techniques can be found in the literature [15, 20, 43, 45].

A censor may also perform *IP Blocking* in which the IP address in a packet is compared against a blacklist. On finding a match, a censor either drops the packet or resets the connection [20].

When the client attempts to establish a HTTP connection, the censor can intercept the HTTP GET request and match the resource path and 'Host' field in the header against a blacklist of URLs and keywords. In case of a match, the censor can reset the TCP connection, drop the HTTP request, or redirect the client to a block page.

Requests to many services (e.g., Facebook) are generally through secure Transport Layer Security (TLS) protocol connections. While such connections are encrypted, censors may still monitor certain fields that are sent in plaintext. For example, censors often detect and block on the Server Name Indication (SNI) field in the TLS handshake header [32].

We refer the reader to [20] for a detailed survey of blocking techniques used by the censors.

2.2 Circumvention Approaches

A client can circumvent censorship in a variety of ways [10, 19, 29, 35]. Following are some popular circumvention techniques and tools.

Public DNS Servers. To bypass certain types of DNS blocking (e.g., DNS hijacking), clients can use public DNS servers. In case of DNS injection, where a censor intercepts DNS *responses* and fakes the resource records, approaches like Hold-On can be used [31].

Domain Fronting (DF). In case of HTTP/TLS blocking, clients can use DF (if supported by the destination server), a technique for hiding the endpoint of a connection while communicating with censored hosts [36]. In a normal client-server interaction, the destination server name appears within the DNS query (plaintext), the TLS SNI extension (plaintext), and the HTTP Host header (encrypted). With DF, the DNS query and SNI carry the name of a front-end server (which is not blocked by the censor), while the HTTP Host header (which is encrypted and thus hidden from the censor), carries the name of the intended backend server (the blocked destination). For example, google.com acts as a front-end server for the youtube.com backend destination.

Virtual Private Networks (VPNs). Many users in censored regions use VPNs to connect to proxy servers outside the censored region to access content. They are usually blocked via ports, IP addresses, or deep packet inspection.

Tor. While Tor was initially designed as an anonymity tool but in recent years, it has also become a popular circumvention tool. Tor is usually able to circumvent almost all kinds of blocking but fails in regions that block addresses of Tor bridges [33, 59].

Lantern and uProxy. Lantern [10] uses a network of HTTPS proxy servers and a client software that allows users to discover and use these servers to bypass censorship. Unlike Tor, Lantern does not provide anonymity and focuses more on performance and availability. uProxy [19] also leverages trust relationships but runs as a browser extension.

2.3 A Censorship Case Study

To quantify opportunities for improving circumvention performance through censorship measurements, we conduct a case study on distributed censorship in Pakistan. Our key insight is that censors possess different capabilities for blocking traffic based on the availability of financial and human capital and these differences can be leveraged to improve circumvention performance [52].

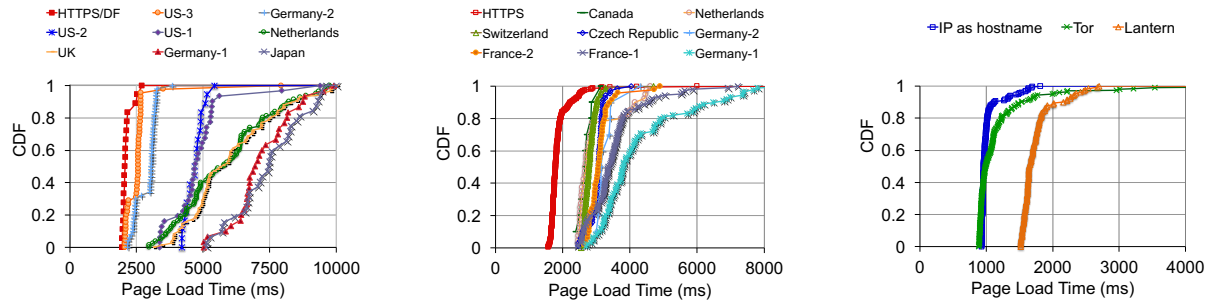
Dataset and methodology. Our dataset was collected from a University campus as well as residential networks in two cities within the censored region[3]. The University connects to the Internet via two of the largest ISPs in Pakistan (referred to as ISP-A and ISP-B from now on). The dataset was collected by sending HTTP/HTTPS requests for different blocked websites. The University site we use for performing these tests does not itself censor the type of content being tested. While we focus on YouTube, we also consider anti-religious and pornographic blocked content inside Pakistan.

Insights about censors. The analysis of our dataset reveals two key insights regarding how ISPs enforce censorship in Pakistan: (a) blocking mechanisms can differ *across* ISPs and (b) blocking mechanisms can differ *across* URLs even within

[3]This dataset was collected between February, 2015 and September, 2015. While YouTube was unblocked in Pakistan in January, 2016 [16] but thousands of porn, religious, and political websites remain blocked.

Website/Categories	ISP-A	ISP-B
YouTube	HTTP Blocking → Redirected to a block page	1) DNS Blocking → Resolved to a local host in ISP-B 2) HTTP/HTTPS Blocking → Request dropped
Rest (Social, Porn, Political,..)	HTTP Blocking → Redirected to a block page	HTTP Blocking → Block page via iframe

Table 1: Comparison of filtering mechanisms used by ISP-A and ISP-B, both of which are located in Pakistan.



(a) HTTPS/DF vs. Static Proxies **(b) HTTPS vs. Tor** **(c) Lantern vs. "IP as hostname"**

Figure 1: Comparison of (a) static proxies located around the world with HTTPS/Domain-Fronting, (b) HTTPS vs. Tor with different exit relay locations for fetching the YouTube homepage (≈360 KB in size) and (c) Lantern with "IP as hostname" (i.e., using the IP address of the blocked website in the URL as opposed to the hostname) for a porn website with ≈50 KB page size. Note that these experiments were performed on a University campus. Each figure shows results for 200 back-to-back runs.

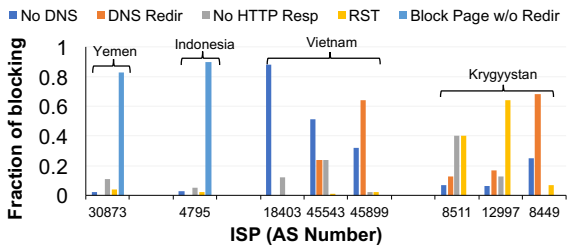


Figure 2: Fraction of blocking types, across ISPs in different countries, measured using the ONI dataset [15, 38]. `No DNS` refers to cases in which no DNS response was received for a censored page, `DNS Redir` when a user is redirected to a different IP, `No HTTP Resp` when no HTTP response is received, `RST` when a TCP reset is received and concluded to be due to blocking, and `Block Page w/o Redir` when a block page is received without DNS redirects.

Static Proxy Location	Avg. Ping Latency (ms)
UK	228
Netherlands	172
Japan	387
(US-1, US-2, US-3)	(329, 429, 160)
(Germany-1, Germany-2)	(309, 174)

Table 2: Comparison of ping latencies to different static proxies from our measurement location. The ping latency to YouTube from the same location was 186ms.

an ISP. In particular, we observed that ISP-A was carrying out HTTP-level blocking, whereas ISP-B blocked both HTTP and HTTPS traffic (see Table 1). In addition, ISP-B was also observed to be carrying out DNS blocking. This is known as *multi-stage blocking*, which is usually carried out to load balance traffic across filtering devices.

Blocking mechanisms across the world. Such heterogeneity in blocking mechanisms has also been observed in several other countries including Thailand, UAE, Burma, and South Korea [38]. Figure 2 shows the fraction of blocking types across different ISPs in Yemen, Indonesia, Vietnam, and Kyrgyzstan. Observe that DNS censorship (e.g., via dropping of DNS requests or responses, redirects to a private IP address or to the address of a proxy that delivers a block page) and HTTP blocking (e.g, by dropping the HTTP GET request, delivering a block page, or injecting a TCP reset) is common, however, their distribution varies across ISPs and countries.

Fine-grained censorship measurements can reveal such differences in blocking mechanisms, which can be used to select the least overhead circumvention strategy for a given filtering mechanism.

Insights about circumvention. We carried out measurements over several weeks to study PLTs under *direct* circumvention mechanisms (e.g., using HTTPS in ISP-A which blocks only HTTP traffic, or using domain fronting in ISP-B to unblock HTTPS traffic) as well as with *indirect* approaches that use relays (e.g., Lantern). For these experiments, we focus on the PLTs of the YouTube homepage. As our results exhibited a similar trend across weeks, we report only a subset of the results.

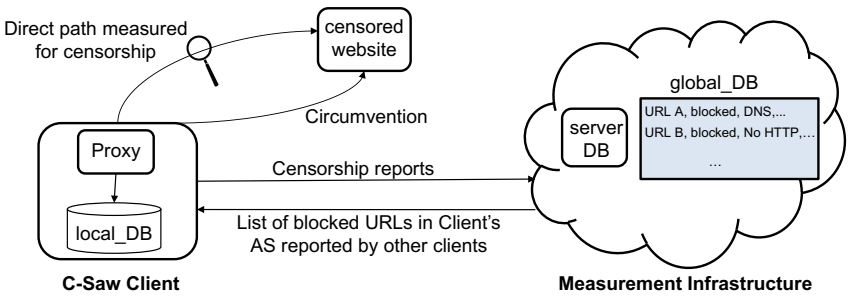


Figure 3: C-Saw components and their interaction.

Comparison with Static Proxies. Users often access blocked URLs using static proxies that are spread throughout the world. We observed that PLTs under a direct method (i.e., using HTTPS/DF at ISP-B) were significantly better than under static proxies located in US, Europe, and Asia as shown in Figure 1a. The average ping latencies[4] are shown in Table 2. Observe that some proxies (e.g., Germany-1, UK, and Japan) resulted in PLTs that varied widely across runs suggesting either real-time on-path congestion or high load at the proxy.

Comparison with Tor. Tor operates by establishing a circuit—which comprises three relays: entry, middle, and exit nodes—and changes it over time (usually every 10mins unless the circuit fails). Thus, we collected and isolated measurement results for every unique circuit. We recorded the location of the exit relay used by Tor across the measurement runs. We observed that in most cases, using HTTPS instead of Tor for accessing YouTube resulted in significantly lower PLTs (see Figure 1b). This is because Tor's use of multiple relays often results in much longer paths, which inflates PLTs[5].

Comparison with Lantern. We now compare the performance of Lantern (another proxy-based circumvention approach) with directly using the IP address as hostname in the URL of a blocked porn page (size ≈50KB) to bypass keyword filtering. Unlike Tor, Lantern does not provide anonymity and focuses more on availability [36]. Observe that Lantern results in ≈1.5x longer PLTs compared to the "IP as hostname" approach (see Figure 1c). This happens because Lantern leverages trust relationships when choosing relays. As a result, traffic can go through longer paths compared to the direct approach.

Summary. These results show that different circumvention techniques can lead to *widely* different PLTs. Thus, fine-grained censorship measurements can reveal differences in blocking mechanisms, which can be used for improving circumvention performance.

[4]These do not include the latency from the proxies to YouTube.

[5]Anonymity is not always required or desirable especially when it comes at the cost of performance as there exist countries (e.g., Pakistan) where there are no legal implications on users who bypass censorship.

3 C-SAW'S DESIGN RATIONALE

We now present C-Saw's key design goals and principles, an overview of the system, and the threat model.

Design Goals. Motivated by the insights from our case study, we set forth the following design goals for C-Saw:

- G1–*Scalable Measurements with User Consent:* The system should allow collection of fine-grained censorship measurements from large number of users with their consent.
- G2–*Adaptive Circumvention:* The system should be able to dynamically adapt the circumvention approach based on the type of filtering to minimize PLTs.

In addition, a practical and usable solution should obey the following constraints:

Constraint-1: The system should *not require* a set of target URLs to be tested for censorship.

Constraint-2: The system should preserve the privacy of users contributing censorship measurements.

Design Principles. The above design goals lead to the following design principles for C-Saw.

- To collect diverse and continuous measurements, a system should have built-in *incentives*. C-Saw offers small PLTs as an incentive and obtains user consent to measure only those URLs that a user visits in accordance with their natural browsing habits.
- To achieve high circumvention performance, the system should be able to (a) quickly determine the blocking mechanism and (b) adapt the circumvention strategy to choose the one with the least overhead.

We realize these design principles in C-Saw by combining measurements and circumvention in a single platform. Of course, performance is not the only incentive for users, however, we view it as a useful incentive as it directly impacts user experience and engagement.

C-Saw Overview. A high-level view of C-Saw's design, shown in Figure 3, highlights its three key components: (a) a client-side proxy that comprises a measurement module and

Field	Description
URL	Requested URL by the user
AS Number	Autonomous system number
T_m	Time at which the URL was measured
Status	blocked, not-blocked, or not-measured
Stage-1 Blocking	Blocking experienced at stage 1
...	...
Stage-k Blocking	Blocking experienced at stage k
Global Posted	Was this URL posted to the global_DB?

Table 3: Fields in the local_DB.

a circumvention module, (b) a local database (local_DB) on the client's machine for storing information about URLs that the client visits, and (c) a global database (global_DB) along with a co-located server (server_DB) for storing system-wide measurements of censored URLs from all C-Saw clients.

Initialization. When users install C-Saw, the client-side proxy registers itself with the global_DB by first asking the client to solve a "No CAPTCHA reCAPTCHA"; a new reCAPTCHA API that uses a risk analysis engine and adaptive CAPTCHAs to keep automated software from creating large number of fake accounts [17]. During this phase, the server_DB sends a unique ID to the user for sending future updates to the global_DB. These updates include information about *only* blocked URLs. Finally, the proxy downloads information about all URLs that are blocked from the client's AS.

Workflow. After initialization, all URL requests are automatically routed through the local C-Saw proxy. The measurement module within the proxy first consults the local_DB to ascertain the blocking status of a URL. It only consults the local copy of global_DB[6] if the URL is found to be unblocked in the local_DB. Based on the response, the direct path and/or some circumvention path are are used for fetching the URL. Whenever the measurement module processes a *new* URL, it adds an entry to the local_DB. Furthermore, each client periodically sends its measurements to the global_DB.

Threat Model. We assume an adversary can block, modify, or reject a web connection at any time but is unwilling to block all web traffic (possibly due to collateral damage). The adversary may attempt to (a) block clients' access to the global_DB, (b) distort censorship measurements stored in the global_DB by contributing false measurements, and (c) identify users who are contributing measurements to the global_DB. C-Saw considers all these aspects in its design. We assume that an adversary cannot gain unauthorized access to the local_DB or the global_DB and that users trust the C-Saw proxy itself (e.g., the proxy cannot be subverted by an adversary to proxy user traffic through malicious nodes).

[6]Note that this copy only contains information about blocked URLs that have the same AS as the client.

Field	Description
T_p	Time when the update was posted
UUID	Universal unique identifier

Table 4: The global_DB contains fields in the local DB as well as these additional fields.

4 C-SAW DETAILED DESIGN

We now present the design of C-Saw's components and describe their interactions.

4.1 Local Database

Each record in the local_DB contains a URL (used as an index), the autonomous system (AS) number at the time of measurement, the time when the URL was last measured (T_m), its blocking status, type of blocking experienced (we add multiple fields to track multi-stage blocking), and whether the latest update has been posted to the global_DB (see Table 3). The blocking status of a URL can be either `blocked` or `not-blocked` if the URL was *previously* measured and is set to `not-measured` if it was either never measured before or its record in the local_DB expired based on the system timer.

4.2 Global Database

The global_DB contains all fields in the local_DB and two additional fields (see Table 4): (a) time when an update is posted (T_p) and (b) a server assigned universal unique identifier (UUID)—which is a cryptographic hash of the current system time—to allow users to post updates for measurements they report. The UUID also allows consumers of measurements to perform user-centric analytics (e.g., number of users reporting measurements from a certain AS). To protect user privacy, we do not store IP addresses in the global_DB.

To benefit from crowdsourced measurements reported by *other* users on the *same* AS, clients periodically download the list of blocked URLs from the global_DB. This obviates the need for clients to locally measure *every* URL for censorship, which reduces overhead and enables faster access to blocked content. Note that efficiently pushing large lists to end-users is common, e.g., Google Safe Browsing provides lists of URLs that contain malware or phishing content and is used by many browsers including Chrome and Firefox [6].

The global_DB and the server_DB can be hosted on one or more cloud platforms. Many cloud platforms already provide dynamic scaling of resources to handle high load (e.g., the auto scaling feature in Amazon EC2 [1] and Microsoft Azure [9] provides this service), ensure high availability in case of failures, and offer DDoS mitigation services (e.g., Amazon's AWS shield [2]).

```
Input  : URL
result = check_local_DB(URL);
/* if URL is not in DB                                   */
if result == not-measured then
    send_redundant_requests ( );
    result = measure_direct_path ( );
/* if URL is in DB but blocked                           */
else if result == blocked then
    circumvent based on the returned approach;
    randomly do;
    (a) send_request_direct_path ( );
    (b) result = measure_direct_path ( );
/* if URL is in DB but not blocked                       */
else
    send_request_direct_path ( );
end
add_to_local_DB (URL, result);
/* Each URL entry expires after t secs                   */
```

Algorithm 1: C-Saw Measurement Module

4.3 C-Saw Proxy

The C-Saw proxy resides on a client's machine and implements the *measurement* and *circumvention* modules.

4.3.1 Measurement Module. All URL requests from a user first go through the measurement module, which performs four key functions: (1) it fetches the URL record from the local_DB, (2) forwards either one request or multiple redundant requests—depending on the blocking status—to the circumvention module. The latter facilitates real-time block page detection, (3) measures common forms of Web censorship (e.g., DNS blocking, IP blocking, and HTTP/HTTPS blocking) and writes the results to the local_DB, and (4) periodically sends updates about blocked URLs to the global_DB. Algorithm 1 describes the flow of tasks in this module.

Detection algorithm. The measurement module implements an in-line blocking detection algorithm for requests sent on the direct path. Figure 4 shows a flowchart for detecting common forms of blocking [15, 38]. Note that we declare a URL to be *blocked* when (1) we receive a response from the circumvention path but no response (or receive an anomalous response) from the direct path or (2) the returned page is flagged as a block page.

Redundant requests. There are two key challenges that impact user experience when detecting censorship.

- *High detection times for censored URLs:* If an unmeasured URL is actually blocked, it may take a long time before blocking can be detected (e.g., due to TCP timeouts in case of TCP/IP blocking), thereby hurting user response times. Table 5 shows the blocking detection times for censored URLs in Pakistan that experience different types of blocking. Observe that the detection times can be as high as 32.7s.
- *Detecting real-time content manipulation:* Detecting content manipulation (e.g., replacing a web page with a block page or substituting content) in real-time is challenging due to the dynamic nature of content, which may vary based on region of access or due to personalization [37, 42].

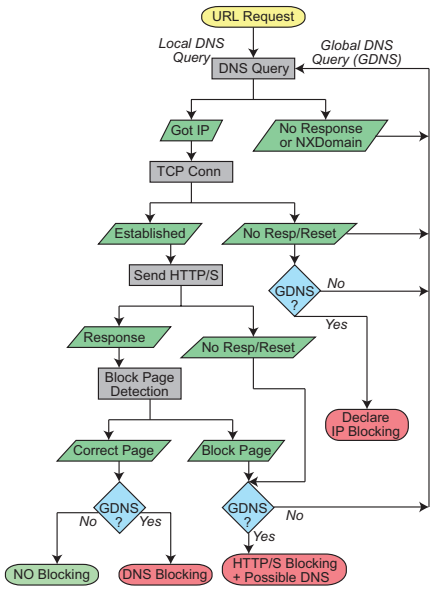


Figure 4: Detecting different types of blocking.

Blocking Type	Avg. Detect. Time(s)
TCP/IP	21
DNS (Response: "Server Failure")	10.6
DNS (Response: "Server Refused")	0.025
HTTP (Block Page)	1.8
TCP/IP + DNS	32.7

Table 5: The average time for detecting different types of blocking including TCP/IP, two types of DNS blocking, HTTP, and multi-stage (i.e., DNS blocking followed by IP blocking). Each result is the average of 50 runs.

We address these challenges in C-Saw by making use of *redundant requests*—one through the direct path and the other through one or more circumvention paths—for URLs having the `not-measured` status.[7]

Reducing response times with selective redundancy. Redundant requests can improve response times when the *direct path* is censored, in which case the user is served the version delivered from the circumvention path. While redundant requests offer improvements when the CPU load is low, they may degrade performance at high loads [28, 39, 55]. This

[7] To avoid multiple writes, HTTP POST requests are not duplicated.

poses a problem for pages with a large number of embedded links. To address this challenge, we use *selective redundancy* i.e., for `not-blocked` URLs, we do not measure the direct path for censorship. While this may increase detection times if the URL becomes censored before its record expires, we think this is an acceptable tradeoff, especially considering that blocking events happen on long time scales.

Real-time detection of content manipulation. To enable fast detection of content manipulation, C-Saw uses a 2-phase algorithm. In the first phase, C-Saw attempts to detect a block page by just examining the direct path response using a heuristic based on HTML tags [42]. If the page is *not* suspected to be a block page, it is served to the user, without waiting for a response from the circumvention path. If the page is deemed a block page, we go to the next phase. In the second phase, we compare the sizes of the returned webpages—one received from the direct path (i.e., block page) and the other from the circumvention path (i.e., actual page), similar to [42]. Such an approach has been shown to be quite effective. The 2-phase approach reduces block page detection times for the common case and achieves high accuracy in other cases at the cost of higher detection times.

Using existing datasets of block pages from 47 ISPs across the globe [3, 13], we find that C-Saw is able to accurately classify ~80% pages as block pages in the first phase, without any false positives (i.e., normal pages being classified as block pages). In case there are false positives, C-Saw simply waits for the second phase. In case of a false negative (i.e., a block page classified as a normal page) at the first step, the block page will be served to the user. However, this is instantly corrected once response from the circumvention path is received, by issuing a page refresh to the browser.

The use of redundant requests also helps in differentiating censorship from network problems (e.g., high delay or packet loss). For instance, if a response is received from the circumvention path but not from the direct path, the latter is more likely to be a censorship event rather than a network problem because both the paths share at least a subset of network resources (e.g., access link) and a problem in them will likely affect both the paths.

Low overhead vs. resilience to false reports. C-Saw clients periodically download the list of blocked URLs from the global_DB. Relying *solely* on these crowdsourced measurements presents two challenges: (i) it makes the system *vulnerable* to false reports—that can cause clients to use more costly circumvention approaches thus leading to higher PLTs—and (ii) it can reduce the *footprint* of measurements. To address these challenges, C-Saw clients measure the direct path for blocked URLs in global_DB *randomly* with probability p and independently for each URL. The value of p presents a tradeoff: $p = 0$ implies that clients solely rely on global_DB for measurements, which makes C-Saw vulnerable to false reports whereas $p = 1$ implies that clients *always* measure the direct path, which increases measurement overhead and reduces the usefulness of global_DB. Thus, p should be between 0 and 1 to achieve a balance between low overhead and resilience to false reports.

4.3.2 Circumvention Module. This module hosts different circumvention services and allows URL requests to be dispatched via the direct path, *local-fix*, static proxy, Tor, or any other available circumvention method. The local-fix is any *non-relay* based circumvention method, which varies based on deployed censorship. For example, in case of DNS blocking, the local-fix is to use a public/global DNS. In case of HTTP blocking, HTTPS is used as a local-fix (if available), and when HTTPS traffic is blocked, the local-fix is to use domain fronting (if available).

Selecting a circumvention approach. For a `blocked` URL, C-Saw may have multiple options for circumventing censorship. We aim to select a circumvention approach that is expected to result in the smallest PLT. To this end, we always prefer local-fixes over relay-based approaches (e.g., Tor and Lantern) as the former usually have smaller path latencies. If multiple relay-based approaches can be used for circumvention, we normally choose the one that yields the smallest PLT, by way of maintaining a moving average of PLTs for each circumvention approach and URL. To accommodate the case where, over time, a circumvention approach may improve in PLTs, we use a randomly chosen circumvention approach (among possible approaches) for every $n = 5$-th access to the URL.

4.4 C-Saw Features and Optimizations

C-Saw's additional features include: (a) a modular design with user customization, (b) ability to track evolution of censorship, (c) an algorithm for reducing the size of local_DB, which can be particularly useful for memory-constrained devices in developing countries [21], and (d) the ability to manage multi-homed clients.

Modular design with user customization. C-Saw's design facilitates evolution and thus, can incorporate new censorship detection algorithms and circumvention methods via automatic software updates. Furthermore, a user can specify a desired configuration according to personal preferences such as high performance or anonymity. If a user prefers performance over anonymity, the C-Saw proxy always picks local-fixes (whenever available). If a user prefers anonymity over performance, C-Saw always chooses an anonymous circumvention approach (e.g., Tor).

URL status churn. Based on censors' policies, unblocked URLs may get censored whereas censored URLs may get whitelisted over time. C-Saw tracks these changes as follows:

- *Scenario A: Blocked → Unblocked.* This type of churn is handled by expiring the timer associated with the URL record in the local_DB. On expiration, the URL status is changed to `not-measured`. Then, as per Algorithm 1, the proxy issues redundant requests over the direct path and additional circumvention path(s), and will be able to observe white listing of the URL.
- *Scenario B: Unblocked → Blocked.* This type of churn is implicitly addressed in the design because the proxy always *measures* the direct path for blocking.

Managing the database size. C-Saw reduces the size of the local_DB, an in-memory hash table, by (1) aggregating URLs and (2) expiring stale entries. With URL aggregation, derived URLs are aggregated into a single URL, typically the domain name or the hostname. This aggregation policy varies based on the type of blocking.

(1) *HTTP blocking*: We consider three cases (a) if the base URL (e.g., `http://www.foo.com/`) is blocked, we just keep this one record in the local_DB and consider all derived URLs (e.g., `http://www.foo.com/a.html`) as blocked, (b) if a derived URL is blocked, then its base or derived URLs on the same base may or may not be blocked (e.g., censors sometimes block only specific pages [12]), thus, we make an entry for the derived URL[8], and (c) if a URL, base or derived, is found to be *uncensored*, we keep only one entry in the local_DB corresponding to the base URL. Considering cases (b) and (c) collectively requires longest prefix matching to find the correct status of a derived URL that is `blocked`.
(2) *IP, DNS, and HTTPS blocking*: A censor uses these mechanisms to filter an IP address or a hostname. For example, in case of HTTPS, a censor filters traffic based on the SNI field, which contains the hostname in clear text. Thus, if a censor uses either of these approaches, we mark the base URL to be blocked (and store a single entry in local_DB), even if we detect a derived URL as being censored.

Multi-homed clients. Multi-homing creates a challenge for C-Saw as it can increase PLTs. Consider a network that randomly maps requests to one of two ISPs, A and B. Assume A blocks URL Y but B does not. Suppose when the request for URL Y first arrives, it goes through provider A. C-Saw will mark the URL `not-blocked`. However, a subsequent request for URL Y may go through ISP B that filters Y, requiring blocking detection. For the next request, C-Saw may choose a more expensive relay-based circumvention approach. This oscillatory behavior, whereby a URL is deduced to be `not-blocked` and `blocked` can continue leading to degraded performance. C-Saw addresses this challenge by detecting multi-homing and then using a circumvention strategy commensurate with the blocking mechanism of the filtering ISP (in case only one ISP is blocking the URL) or the more strict censorship. For example, if A filters HTTPS traffic for URL Y and B does not, we use HTTP/DF for all subsequent requests for URL Y. We detect multi-homing by periodically generating requests for determining the ASN of the providers. If over short timescales, more than one ASN is returned, we mark the network to be multi-homed.

[8]We avoid measuring the homepage due to lack of user consent, which may not be practical to obtain in this case.

5 SECURITY AND PRIVACY CONSIDERATIONS

We now discuss some challenges related to user privacy, security, and the quality of collected measurements.

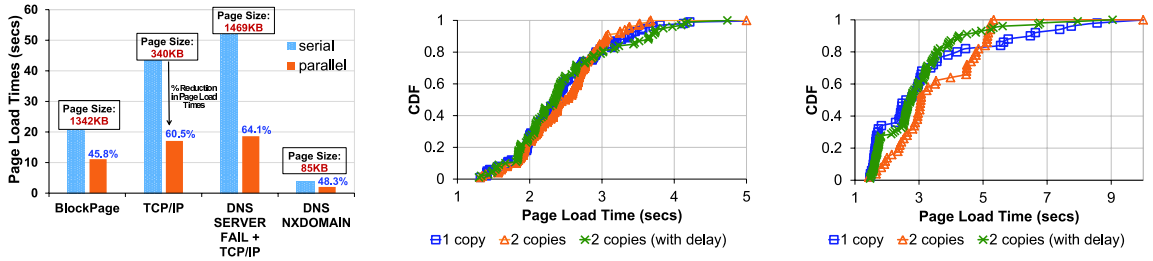
Interfering with C-Saw measurements. For any system that relies on crowdsourced measurements, malicious users may distort them by false reports [26]. Measurements in global_DB are also prone to such distortions.

C-Saw aims to address this challenge by (a) rate limiting the creation of automated fake identities from malicious users by requiring them to solve "No CAPTCHA reCAPTCHA", a new reCAPTCHA API that uses an advanced risk analysis engine and adaptive CAPTCHAs [17], before they can register as a C-Saw client and (b) using a voting mechanism to limit the impact of a single malicious client on measurements. The voting process is run by the server_DB. It assigns each client c_i one unit of vote, which it evenly spreads among all blocked URLs that it reports, i.e., it gives $v_{i,j,k}=1/d$ vote to each of the d blocked websites, where j and k are the blocked URL and client AS, respectively. To establish a confidence criteria for measurements posted in the global_DB, the server_DB maintains two statistics for each blocked URL from an AS as estimates of robustness: (i) sum of votes $s_{j,k}$ and (ii) the *total number of clients* $n_{j,k}$ voting for the given URL from an AS. A user may pay less heed to a measurement with large $n_{j,k}$ and small $s_{j,k}$ (indicating large number of reports per user) or measurements with small $n_{j,k}$ to mitigate the impact of erroneous reports. This is partly inspired by the PageRank algorithm [25].

Finally, one can also design schemes, similar to reputation systems [47, 57], for identifying individual malicious users or groups based on distinctness in behavioral patterns and revoke UUIDs of malicious users [54].

Attacks on the server_DB and global_DB. An attacker may launch a flood attack on the measurement infrastructure (e.g., the server_DB). To address this challenge, these components can be replicated across different cloud providers, which already provide dynamic scaling and offer DDoS mitigation services [2, 9]).

Blocking access to the global_DB. A global_DB with a well-known domain name can be blocked if the Tor exit



(a) Blocked Webpages (b) Small Unblocked Webpages (c) Larger Unblocked Webpages

Figure 5: Impact of redundant requests on PLTs for (a) blocked webpages with different types of blocking under the `serial` approach and the `parallel` approach, (b) small unblocked webpages (95 KB) and (c) relatively large unblocked webpages (316 KB). For these experiments, we launch 100 requests with uniformly distributed inter-arrival times from the interval [1s, 5s].

relay for sending censorship reports lies in the same region as the censor. This challenge can be addressed by using a distributed collection service, similar to OONI [14], which exposes each collector as a Tor Hidden Service and uses them to post results to the global_DB.

User privacy and resilience to detection. A censor can snoop on traffic to identify users contributing censorship measurements to the global_DB. This challenge is addressed by requiring that all measurement reports are carried over the Tor network.

6 IMPLEMENTATION

We implement C-Saw as a local proxy running on a client's machine. C-Saw is built using GitHub's electron framework [5] and as a result, can be used across platforms and works with all popular browsers. Our implementation provides support for (i) measuring common forms of censorship including DNS filtering and HTTP/HTTPS blocking and (ii) detecting block pages by issuing redundant requests. C-Saw's circumvention module implements all local fixes, optimizations (including URL aggregation), as well as provides support for Tor and Lantern as relay-based circumvention approaches. For our evaluation, the global_DB was hosted on MongoLab [11]—a cloud database-as-a-service platform—and server_DB was hosted on Heroku [8].

7 EVALUATION

We now present the evaluation of C-Saw, which focuses on[9]: (a) impact of *redundant requests* on PLTs, (b) effectiveness of *URL aggregation*, (c) performance comparison with Tor and Lantern, and (d) analysis of measurements from a deployment study.

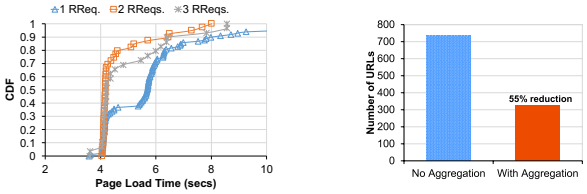
7.1 Impact on Page Load Times

We now evaluate the impact of redundancy on PLTs.

- *Impact on blocked URLs.* We consider two approaches for generating web requests: `serial` and `parallel`. In the `serial` approach, a request is first sent on the direct path and *after* blocking is detected, we send the same request via Tor for circumvention. Note that this approach may reduce the accuracy of real-time block page detection as discussed in Section 4.3. In the `parallel` approach, two parallel copies (one via the direct path and the other via Tor) of a request are sent and the faster of the two responses is shown to the user. Figure 5a shows the PLTs for webpages that experience different types of blocking under the `serial` and `parallel` approaches. Observe that the `parallel` approach provides 45.8%–64.1% reduction in PLTs. This happens because the blocking detection times can often be a significant fraction of the actual PLT.
- *Impact on unblocked URLs.* To evaluate the impact on uncensored URLs, we consider a small webpage (95KB) and a relatively large unblocked webpage (316KB). Figure 5 shows the CDF of PLTs under a single request "1 copy", redundant requests "2 copies", and redundant requests with an added delay of 2s "2 copies (with delay)" between generation of redundant requests[10]. For these experiments, we launch 100 web requests whose inter-arrival times are uniformly distributed between 1s and 5s. For the small webpage, "2 copies (with delay)" performs similar to the "1 copy" case. Observe that delaying generation of a redundant request improves median latency at the cost of higher tail latency. For the larger webpage, "2 copies (with delay)" performs much better than "2 copy" because staggering redundant request helps in reducing client load [55].

[9] All experiments involving censored webpages were carried out from a University campus in Pakistan.

[10] Note that if we get a response for a webpage from the direct path within 2s, we do not send a request on Tor.



(a) (b)

Figure 6: Impact of (a) increasing the number of redundant requests and (b) URL aggregation.

p	Median PLT (s)
0	5.6
0.25	6.9
0.5	7.5
0.75	8.1

Table 6: Impact of p on median PLT.

How many redundant requests are sufficient? We setup an experiment in which we send out one, two, and three duplicate requests for an uncensored webpage. All redundant requests are sent via *separate* Tor circuits and we measure the *minimum* PLT across all duplicate requests. We find that increasing redundancy from one to two improves the median PLT by ~30% (see Figure 6a). While increasing the number of redundant requests to three does not improve the median PLT, it *increases* the 95th percentile PLT by 17%. We attribute this to increased load on the client.

Choosing the value of p. C-Saw clients measure the direct path for *blocked* URLs (reported via global_DB) with probability p. Table 6 shows the median PLT for different values of p involving an experiment where Tor is used as a circumvention approach. Observe that measuring the direct path increases the median PLT. We recommend a value of $p < 0.25$ to achieve a balance between overhead and resilience to false reports. Note that blocked URLs that require a local-fix do not incur this overhead because they use the direct path, which is measured by default without generating any extra traffic.

Different circumvention approaches. The PLTs seen by C-Saw users depend on the specific circumvention method being used. We now evaluate the impact of using Lantern as the relay-based circumvention method in C-Saw and perform comparison when Tor is used. Figure 7c shows the CDF of PLTs with Lantern and Tor in case of multi-stage blocking (IP blocking and DNS blocking). Observe that Lantern significantly outperforms Tor, which can be attributed to Tor's anonymity feature, which introduces overhead.

7.2 Impact of URL aggregation

Next, we study the impact of aggregation. We launch requests for Alexa top 15 websites in Pakistan and track the number of records stored in local_DB *without* aggregation. Later, we enable aggregation and repeat the experiment. Figure 6b shows that aggregation provides ~55% reduction in the number of records in local_DB by storing only one record for URLs whose base URL is found to be unblocked. These savings can be particularly useful for memory-constrained mobile devices common in developing countries [21].

7.3 Comparison with circumvention tools

To analyze how C-Saw's performance compares with other circumvention methods, we evaluate C-Saw (w/ Tor), Lantern, and Tor in isolation. Figures 7a and 7b show the CDF of PLTs with C-Saw, Lantern, and Tor for a blocked (undergoing DNS blocking) and unblocked URL, respectively. Observe that C-Saw significantly outperforms Lantern and Tor due to its adaptive circumvention approach that applies a local-fix. Lantern first detects blocking and then always uses a *relay-based* circumvention for blocked pages. Tor always uses *multiple relays* for circumvention. Observe that for the unblocked webpage, C-Saw outperforms these approaches as well because it simply uses the direct path.

7.4 Deployment Study

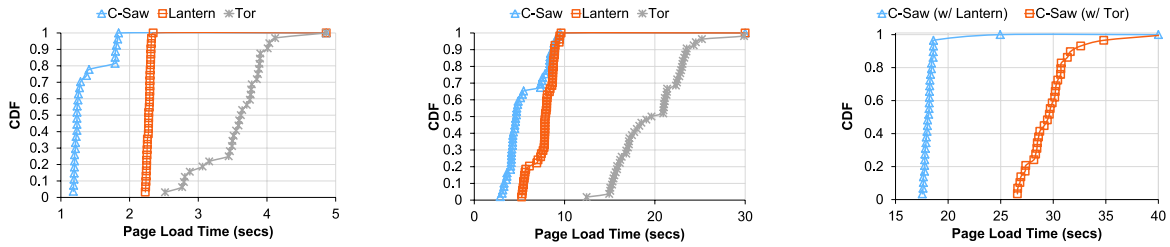
We evaluate the feasibility of deploying C-Saw based on a prototype implementation and a pilot study within a controlled user base. We released C-Saw to 123 consenting users and collected censorship measurements for three months. These included users behind residential, enterprise, and University networks in Pakistan. The users were carefully informed about C-Saw but were not given any list of blocked websites they needed to visit.

Table 7 shows some statistics we farmed from the measurements collected by the global database. We obtained the following insights based on our deployment:

- The users visited 420 blocked domains and accessed them through 16 different ASes.
- We find that for a majority of URLs, a block page was returned. The second most common type of filtering was DNS blocking.
- We found blocking of CDN servers, which was not observed in earlier studies of censorship in Pakistan [44, 45]. C-Saw enables such detection because it receives the censored page from the circumvention path and then sends each request for a CDN on the direct path, which it measures for censorship by default.

7.5 C-Saw in the Wild

Due to protests in the capital city of Pakistan, several Internet services, including Twitter and Instagram, were blocked between November 25-28, 2017. Several C-Saw users attempted to use these services. Here is a snapshot of the measurements we collected:



(a) Blocked page—*DNS Blocking* (b) Unblocked Webpage (c) C-Saw with {Lantern,Tor}

Figure 7: CDF of PLTs of C-Saw in comparison with Lantern and Tor for (a) blocked webpage (*DNS blocking*), (b) unblocked webpage, and (c) C-Saw with Tor and C-Saw with Lantern.

No. of users	123
No. of unique blocked URLs accessed	997
No. of unique blocked domains accessed	420
No. of unique ASes	16
Distinct types of blocking observed	5
No. of URLs experiencing DNS blocking	376
No. of URLs experiencing TCP connection timeout	114
No. of URLs for which a block page was returned	475
No. of unique updates	1787

Table 7: Insights from our deployment study.

- Twitter was found blocked at 13:31 on Nov 25, 2017 from AS 38193 (Response: HTTP_GET_TIMEOUT)
- Twitter was found blocked at 13:32 on Nov 25, 2017 from AS 17557 (Response: HTTP_GET_BLOCKPAGE)
- Instagram was found blocked at 4:51 on Nov 26, 2017 from AS 38193 (Response: DNS blocking)
- Instagram was found blocked at 9:06 on Nov 26, 2017 from AS 59257 (Response: DNS blocking)
- Instagram was found blocked at 9:31 on Nov 26, 2017 from AS 45773 (Response: DNS blocking)

The above snapshot reveals interesting insights, which includes: (i) Different ASes were blocking Twitter differently and (ii) both Twitter and Instagram were blocked by different ASes. Our deployment experience suggests that C-Saw is effective at measuring censorship. Whether performance is a sufficient incentive for adoption, remains to be seen.

8 DISCUSSION

Ethics. There are well-known concerns regarding the ethics of censorship measurements w.r.t. consent, privacy, and safety, and include questions such as: *Are users aware of which URLs they, or someone on their behalf, may be visiting and consent to it? How are users apprised of risks? To what extent can users be implicated for traffic that leaves their machine towards a censored website?* When designing C-Saw, we took explicit steps to abide by the ethical guidelines set out by the Menlo Report [30]. For instance, C-Saw incorporates explicit user consent in its design by measuring *only* those URLs that users access with their knowledge. With C-Saw, users can choose to stay anonymous by using only those circumvention methods that provide anonymity. To protect the identity of users contributing measurements, C-Saw does not store IP addresses and uses Tor for sending measurement reports. Finally, while it is important to apprise users of potential risks, it is challenging to ascertain the actual *degree* of risk when using measurement or circumvention tools [40, 41, 60].

Why not use Tor for measurements? Tor's design makes it challenging to *detect* censorship and provide adaptive circumvention. As Tor is geared towards providing anonymity, it *always* uses relays for accessing *every* URL making it hard to measure censorship on the *direct path*. C-Saw's design allows new circumvention methods to be seamlessly integrated into its framework. This would not be possible if a design with a *specific* circumvention mechanism is used.

Robustness of C-Saw. C-Saw uses Tor as one possible circumvention strategy. However, Tor exits can be easily blacklisted [51], which raises concerns about C-Saw's robustness. While using Tor bridges and pluggable transports makes it more challenging to block Tor, there is an arms race between Tor and some censors (e.g., China) [53]. Our hope is that C-Saw can ride on Tor's successes in achieving blocking resistance with features like pluggable transports. However, it is useful to highlight that censors in several countries are neither as resourceful nor motivated as the censors in countries like China and Iran. Second, Tor is just one of the many circumvention mechanisms that C-Saw relies on. New circumvention approaches can be readily incorporated into its framework.

Fingerprintability of C-Saw. Suppose a censor shuts down circumvention mechanism X, and as a result, C-Saw users migrate to mechanism Y. This flocking behavior may make it easier for censors to fingerprint users who switch to a particular (anonymous) circumvention mechanism. The fingerprinting effectiveness, however, would depend on a number of factors including the number of C-Saw users within an

AS, circumvention strategies being used[11], and the type of blocked URLs being accessed (because blocking mechanisms may differ across URLs). Similarly, C-Saw's use of redundant requests may also seem like a fingerprintable behavior. However, C-Saw uses redundant requests *selectively* (i.e., only when a URL has not been measured before or randomly for blocked URLs) and may use different circumvention mechanisms depending on the blocked content. In some cases, the source IP address visible to the censors may also be different (e.g., in case of Tor and VPNs). Moreover, as redundant requests are commonly used by WAN optimizers, they may be challenging to fingerprint. However, we leave a detailed analysis of such fingerprinting approaches to future work.

Can C-Saw work with mobile users? If a user moves and its AS changes, C-Saw will fetch censorship measurements for the new AS from the global_DB. As measurements from the global_DB are fetched periodically, C-Saw will automatically adapt to user mobility.

C-Saw's data usage. C-Saw uses redundant requests, which increase data usage and may be a concern in developing countries. To limit this increase, C-Saw uses redundancy only selectively (e.g., when a URL has not been measured before or randomly with probability p for blocked URLs). This significantly limits additional data usage in the common case. To further optimize data usage, the value of p can be lowered in developing regions albeit at the cost of reduced resilience to false reports.

Server-based filtering. If a server or content provider filters content in a geographical region (e.g., Google filters content in some countries based on government requests [7]), C-Saw can (a) access such censored content using one of its circumvention services and (b) detect such type of filtering.

Non-web filtering. In this work, we focused on web censorship. In the future, it would be interesting to explore non-web filtering (e.g., messaging, voice, and video applications, such as Whatsapp, and UDP blocking).

9 RELATED WORK

Censorship measurement tools. Existing tools for measuring censorship, such as OONI [37] and Centinel [27], try to recruit users who are either willing to host a measurement device or install a measurement software. However, their widespread deployment remains a challenge due to lack of user incentives. CensMon [50] used PlanetLab nodes but was deployed only for a short time. Iris [49] does not require diverse vantage points but only measures DNS manipulation. Augur [48] uses TCP/IP side channels but focuses on identifying IP-based disruptions only. Encore uses cross-origin requests to measure censorship but cannot determine the *type* of blocking [26].

Circumvention tools. Flash proxy [35] creates many short-lived proxies to outpace the censor's ability to block them. Infranet [34] is designed to conceal traffic that would otherwise be blocked within seemingly normal HTTP traffic. Telex [61] allows tagging normal TLS streams cryptographically so that an ISP-level router may redirect it to a blocked destination. Unlike Infranet, unblocked Web sites do not need to participate in or know about circumvention. Lantern [10] and uProxy [19] leverage trust relationships for choosing proxy servers for circumvention. Alkasir [23] uses a proxy server located in a non-censored region to provide circumvention and requires manual reports from users to enable circumvention. LASTor [22] improves the latency performance of Tor. Astoria [46] provides higher level of security against AS-level attackers than Tor. A survey of existing circumvention techniques can be found in [20].

10 CONCLUSION

Collecting continuous and reliable censorship measurements is challenging due to lack of incentives for user participation. At the same time, circumvention systems lack insights about censor capabilities, which continually evolve over time, leading to inefficient or ineffective circumvention. We develop C-Saw, a system that addresses these challenges by combining censorship measurements with circumvention. Our evaluation shows that C-Saw is effective at reporting measurements and leveraging fine-grained knowledge about filtering mechanisms to improve user-perceived performance.

Acknowledgements. We thank our shepherd, James Mickens, and the anonymous SIGCOMM reviewers for their valuable feedback. We also thank Vern Paxson, Fahad Rafique Dogar, and Zafar Ayyub Qazi for fruitful discussions and their feedback on this work.

REFERENCES

[1] [n. d.]. Auto Scaling. https://aws.amazon.com/autoscaling/.
[2] [n. d.]. AWS Shield. https://aws.amazon.com/shield/.
[3] [n. d.]. citizenLab. https://github.com/citizenlab/blockpages.
[4] [n. d.]. Electron. http://electron.atom.io/.
[5] [n. d.]. GitHub. https://github.com/.
[6] [n. d.]. Google Safe Browsing. https://developers.google.com/safebrowsing/.
[7] [n. d.]. Google Transparency Report. https://www.google.com/transparencyreport/.
[8] [n. d.]. Heroku. https://www.heroku.com/.
[9] [n. d.]. How to auto scale a cloud service. http://bit.ly/1sICRVF.
[10] [n. d.]. Lantern. https://getlantern.org/.
[11] [n. d.]. mongolab. https://mongolab.com/.
[12] [n. d.]. ONI Research Profile: Yemen. http://opennet.net/research/profiles/yemen.
[13] [n. d.]. ooni-block. https://github.com/hellais/blockpages.

[11] Tor performs load balancing by weighting relay selection in proportion to each relay's perceived bandwidth [56].

[14] [n. d.]. OONIB Specification. https://github.com/TheTorProject/ooni-spec/blob/master/oonib.md#20-collector.

[15] [n. d.]. OpenNet Initiative. https://opennet.net/.

[16] [n. d.]. PTA orders ISPs to unblock YouTube immediately. http://tribune.com.pk/story/1029791/pta-orders-isps-to-unblock-youtube-immediately/.

[17] [n. d.]. reCAPTCHA. https://www.google.com/recaptcha/intro/index.html.

[18] [n. d.]. The Belmont Report - Ethical Principles and Guidelines for the protection of human subjects of research. http://ohsr.od.nih.gov/guidelines/belmont.html.

[19] [n. d.]. uProxy. https://www.uproxy.org/.

[20] Giuseppe Aceto and Antonio PescapÃľ. 2015. Internet Censorship detection: A survey. *Computer Networks* 83, 0 (2015), 381 – 421.

[21] Sohaib Ahmad, Abdul Lateef Haamid, Zafar Ayyub Qazi, Zhenyu Zhou, Theophilus Benson, and Ihsan Ayyub Qazi. 2016. A View from the Other Side: Understanding Mobile Phone Characteristics in the Developing World. In *ACM IMC*.

[22] Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. 2014. LASTor: A Low-latency AS-aware Tor Client. *IEEE/ACM Trans. Netw.* 22, 6 (Dec. 2014), 1742–1755.

[23] Walid Al-Saqaf. 2016. Internet Censorship Circumvention Tools: Escaping the Control of the Syrian Regime. *Media and Communication* 4, 1 (2016).

[24] Mashael Alsabah and Ian Goldberg. 2016. Performance and Security Improvements for Tor: A Survey. *ACM Comput. Surv.* 49, 2, Article 32 (Sept. 2016), 36 pages.

[25] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.* 30, 1-7 (April 1998), 107–117.

[26] S. Burnett and N. Feamster. 2015. Encore: Lightweight Measurement of Web Censorship with Cross-Origin Requests. In *ACM SIGCOMM*.

[27] centinel [n. d.]. Centinel. https://github.com/iclab/.

[28] Jeffrey Dean and Luiz AndrÃľ Barroso. 2013. The Tail at Scale. *Commun. ACM* 56 (2013), 74–80.

[29] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-generation Onion Router. In *USENIX Security Symposium*.

[30] D. Dittrich and E. Kenneally. 2012. *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Technical Report. U.S. Department of Homeland Security.

[31] Haixin Duan, Nicholas Weaver, Zongxu Zhao, Meng Hu, Jinjin Liang, Jian Jiang, Kang Li, and Vern Paxson. 2012. Hold-On: Protecting Against On-Path DNS Poisoning. In *Securing and Trusting Internet Names*.

[32] D. Eastlake. [n. d.]. RFC 6066: Transport Layer Security (TLS) extensions: Extension definitions, Jan. 2011.

[33] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *ACM IMC*.

[34] Nick Feamster, Magdalena Balazinska, Greg Harfst, Hari Balakrishnan, and David Karger. 2002. Infranet: Circumventing Web Censorship and Surveillance. In *USENIX Security Symposium*.

[35] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Roger Dingledine, Phil Porras, and Dan Boneh. 2012. Evading Censorship with Browser-Based Proxies. In *PETS*.

[36] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *PETS* (2015).

[37] A. Filasto and J. Appelbaum. 2012. OONI: Open Observatory of Network Interference. In *FOCI*.

[38] Phillipa Gill, Masashi Crete-Nishihata, Jakub Dalek, Sharon Goldberg, Adam Senft, and Greg Wiseman. 2015. Characterizing Web Censorship Worldwide: Another Look at the OpenNet Initiative Data. *ACM Trans. Web* 9, 1, Article 4 (Jan. 2015), 29 pages.

[39] Ali Musa Iftikhar, Fahad Dogar, and Ihsan Ayyub Qazi. 2016. Towards a Redundancy-Aware Network Stack for Data Centers. In *ACM HotNets*.

[40] Ben Jones, Roya Ensafi, Nick Feamster, Vern Paxson, and Nick Weaver. 2015. Ethical Concerns for Censorship Measurement. In *Ethics in Networked Systems Research*.

[41] Ben Jones and Nick Feamster. 2015. Can Censorship Measurements Be Safe(r)?. In *ACM HotNets*.

[42] Ben Jones, Tzu-Wen Lee, Nick Feamster, and Phillipa Gill. 2014. Automated Detection and Fingerprinting of Censorship Block Pages. In *ACM IMC*.

[43] Sheharbano Khattak, Mobin Javed, Syed Ali Khayam, Zartash Afzal Uzmi, and Vern Paxson. 2014. A look at the consequences of Internet Censorship through ISP Lens. In *ACM IMC*.

[44] Zubair Nabi. 2013. The Anatomy of Web Censorship in Pakistan. In *FOCI*.

[45] Aqib Nisar, Aqsa Kashaf, Zartash Afzal Uzmi, and Ihsan Ayyub Qazi. 2015. A Case for Marrying Censorship Measurements with Circumvention. In *ACM HotNets*.

[46] Rishab Nithyanand, Oleksii Starov, Adva Zair, Phillipa Gill, and Michael Schapira. 2016. Astoria: AS-aware relay selection for Tor. In *NDSS*.

[47] Jeffrey Pang, Ben Greenstein, Michael Kaminsky, Damon McCoy, and Srinivasan Seshan. 2009. Wifi-reports: Improving Wireless Network Selection with Collaboration. In *MobiSys*.

[48] Paul Pearce, Roya Ensafi, Frank Li, Nick Feamster, and Vern Paxson. 2017. Augur: Internet-Wide Detection of Connectivity Disruptions. In *IEEE Symposium on Security & Privacy*.

[49] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. 2017. Global Measurement of DNS Censorship. In *USENIX Security Symposium*.

[50] A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis. 2011. A Web Censorship Monitor.. In *FOCI*.

[51] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. 2017. Characterizing the Nature and Dynamics of Tor Exit Blocking. In *USENIX Security Symposium*.

[52] Michael Carl Tschantz, Sadia Afroz, Anonymous, and Vern Paxson. 2016. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *IEEE Symposium on Security and Privacy*.

[53] M. C. Tschantz, S. Afroz, Anonymous, and V. Paxson. 2016. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *2016 IEEE Symposium on Security and Privacy*.

[54] Bimal Viswanath, Muhammad Ahmad Bashir, Muhammad Bilal Zafar, Simon Bouget, Saikat Guha, Krishna Gummadi, Aniket Kate, and Alan Mislove. 2015. Strength in Numbers: Robust Tamper Detection in Crowd Computations. In *ACM COSN*.

[55] Ashish Vulimiri, Philip Brighten Godfrey, Radhika Mittal, Justine Sherry, Sylvia Ratnasamy, and Scott Shenker. 2013. Low Latency via Redundancy. In *ACM CoNEXT*.

[56] Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. 2013. An Empirical Evaluation of Relay Selection in Tor. In *NDSS*.

[57] Kevin Walsh and Emin Gün Sirer. 2006. Experience with an Object Reputation System for Peer-to-peer Filesharing. In *NSDI*.

[58] Z. Weinberg, M. Sharif, J. Szurdi, and N. Christin. 2017. Topics of Controversy: An Empirical Analysis of Web Censorship Lists. In *PETS*.

[59] P. Winter and S. Lindskog. 2012. How the Great Firewall of China is Blocking Tor. In *FOCI*.

[60] J. Wright, T. De Souza, and I. Brown. 2011. Fine-Grained Censorship Mapping: Information Sources, Legality and Ethics. In *FOCI*.

[61] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. 2011. Telex: Anticensorship in the Network Infrastructure. In *USENIX Security Symposium*.