sciendo

Proceedings on Privacy Enhancing Technologies ; 2020 (4):321–335

Benjamin VanderSloot*, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J. Alex Halderman, Nikita Borisov, and Eric Wustrow

Running Refraction Networking for Real

Abstract: Refraction networking is a next-generation censorship circumvention approach that locates proxy functionality in the network itself, at participating ISPs or other network operators. Following years of research and development and a brief pilot, we established the world's first production deployment of a Refraction Networking system. Our deployment uses a high-performance implementation of the TapDance protocol and is enabled as a transport in the popular circumvention app Psiphon. It uses TapDance stations at four physical uplink locations of a mid-sized ISP, Merit Network, with an aggregate bandwidth of 140 Gbps. By the end of 2019, our system was enabled as a transport option in 559,000 installations of Psiphon, and it served upwards of 33,000 unique users per month. This paper reports on our experience building the deployment and operating it for the first year. We describe how we overcame engineering challenges, present detailed performance metrics, and analyze how our system has responded to dynamic censor behavior. Finally, we review lessons learned from operating this unique artifact and discuss prospects for further scaling Refraction Networking to meet the needs of censored users.

1 Introduction

National-governments are deploying increasingly sophisticated systems for Internet censorship, which often take the form of deep-packet inspection (DPI) middleboxes located at network choke-points [5]. At the same time, popular circumvention techniques, such as domain fronting and VPNs, are becoming harder to deploy or more frequently blocked [15]. There is an urgent need to field more advanced circumvention technologies in order to level the playing field.

One proposed new circumvention approach, Refraction Networking, has the potential to fill this need, and has been developed in the form of several proposed protocols [2, 7, 9, 13, 16, 17, 20, 27, 28] and other research [3, 10, 14, 19, 25] over the past decade. It works by deploying technology at ISPs or other network operators that observes connections in transit and provides censorship circumvention functionality. Though promising in concept, deploying Refraction Networking in the real world has faced a number of obstacles, including the complexity of the technology and the need to attract cooperation from ISPs. Other than a one-month pilot that our project conducted in 2017 [8], no Refraction implementation has ever served real users at ISP scale, leaving the approach's practical feasibility unproven.

In this paper, we describe lessons and results from a real-world deployment of Refraction Networking that we have operated in production for over a year and that is enabled as a transport in more than 559,000 installations of the popular Psiphon circumvention tool for PC and mobile users. Building on our 2017 pilot, the deployment is based on a high-performance implementation of the TapDance protocol [8]. It operates from stations installed at Merit Network, a mid-sized ISP, that observe an average of 70 Gbps of aggregate commodity traffic from four network locations, each of which individually processes a peak of 10–40 Gbps. The system has served up to 500 Mbps of proxied traffic to circumvention users.

Building and running this deployment required solving complex technical, operational, and logistical challenges and necessitated collaboration among researchers, network engineers, and circumvention tool developers. This reflects a non-trivial challenge to Refraction Networking systems: Refraction Networking cannot function in isolation from the rest of the Internet, and so its success depends on close interactions between the Internet operator and Internet freedom communities.

In order to serve users well and meet requirements set by our ISP and circumvention tool partners, we worked within the following technical constraints:

***Corresponding Author: Benjamin VanderSloot:** University of Michigan, E-mail: benvds@umich.edu
Sergey Frolov: University of Colorado Boulder, E-mail: sergey.frolov@colorado.edu
Jack Wampler: University of Colorado Boulder, E-mail: jack.wampler@colorado.edu
Sze Chuen Tan: University of Illinois, Urbana-Champaign, E-mail: stan13@illinois.edu
Irv Simpson: Psiphon, E-mail: i.simpson@psiphon.ca
Michalis Kallitsis: Merit Network, E-mail: mgkallit@merit.edu
J. Alex Halderman: University of Michigan, E-mail: jhalderm@eecs.umich.edu
Nikita Borisov: University of Illinois, Urbana-Champaign, E-mail: nikita@illinois.edu
Eric Wustrow: University of Colorado Boulder, E-mail: ewust@colorado.edu

- Each TapDance station could use only 1U of physical rack space at one of the ISP's uplink locations.
- All stations at the ISP would need to coordinate to function as a single distributed system.
- The deployment had to operate continuously, despite occasional downtime of individual stations.
- We had to strictly avoid interfering with the ISP's network operations or its customers systems.
- The deployment had to achieve acceptable network performance to users in censored environments.

In this paper we describe our experience meeting these requirements and the implications this has for further deployment of Refraction Networking.

In addition, we analyze data from four months of operations to evaluate the system's performance. This four-month period reflected typical behavior for our ISP partners and concluded with a significant censorship event that applied stress to infrastructure of our circumvention tool deployment partner. It shows that our deployment's load is affected by censorship practices, and that it was able to handle the spike in utilization effectively. During the censorship event, we provided Internet access to more users than at any previous time, and the system handled this load without measurable degradation in quality of service or generating excessive load on decoy websites, as reflected by the opt-out rate.

Our final contribution is a discussion of lessons we learned from building and operating the deployment that can inform future work on Refraction Networking and other circumvention technologies. We identify two particular areas—decoy site discovery and reducing station complexity—where further research and development work would be greatly beneficial.

We conclude that Refraction Networking can be deployed continuously to end-users with sufficient network operator buy-in. Although attracting ISP partnership remains the largest hurdle to the technology's practical scalability, even with relatively limited scale, Refraction can meet a critical real-world need as a fall-back transport that can provide service when other, lighter-weight transports are disrupted by censors.

The remainder of this paper is structured as follows. In Section 2, we discuss existing techniques for and use of Refraction Networking. We then describe our deployment's architecture in Section 3. In Section 4, we quantify the performance of our deployment using data from the first four months of 2019. We end the discussion with a comparison to existing decoy routing schemes. In Section 5, we draw lessons from our deployment experience. Finally, we conclude in Section 6.

2 Background

Refraction Networking (previously known as "decoy routing") is an anticensorship strategy that places circumvention technology at Internet service providers (ISPs) and other network operators, rather than at network endpoints. Clients access the service by making innocuous-looking encrypted connections to existing, uncensored websites ("decoys") that are selected so that the connection travels *through* a participating network. The client covertly requests proxy service by including a steganographic tag in the connection envelope that is constructed so that it can only be detected using a private key. At certain points within the ISP's network, devices ("stations") inspect passing traffic to identify tagged connections, use data in the tag to decrypt the request, proxy it to the desired destination, and return the response as if it came from the decoy site. To the censor, this connection looks like a normal connection to an unblocked decoy site. If sufficiently many ISPs participate, censors will have a difficult time blocking all available decoy sites without also blocking a prohibitive volume of legitimate traffic [24].

Refraction Networking was first proposed in 2011. Three independent works that year—Telex [28], Curveball [16] and Cirripede [13]—all proposed the idea of placing proxy "stations" at ISPs, with various proposals for how clients would signal the ISP station. For instance, Curveball used a pre-shared secret between the client and station, while Telex and Cirripede used public-key steganography to embed tags in either TLS client-hello messages or TCP initial sequence numbers. Without the correct private key, these tags are cryptographically indistinguishable from the random protocol values they replace, so censors cannot detect them.

However, all of these first-generation schemes required *inline blocking* at the ISP; that is, the station needed to be able to stop packets in individual tag-carrying TCP connections from reaching their destination. This lets the station pretend to be the decoy server without the real decoy responding to the client's packets. While this makes for a conceptually simpler design, inline blocking is expensive to do in production ISP networks, where traffic can exceed 100s of Gbps. Inline blocking devices also carry a higher risk of failing closed, which would disrupt other network traffic, making ISPs leery of deploying the Telex-era protocols.

To address this concern, researchers developed TapDance [27], which only requires a passive tap at the ISP station, obviating the need for inline blocking. Instead,

TapDance clients mute the decoy server by sending an incomplete HTTP request inside the encrypted TLS connection. While the decoy server is waiting for the request to complete, the TapDance station spoofs its IP address and sends a response to the client. In this way, the client and station can communicate bidirectionally, but, to the censor, their dialog appears to be a complete (encrypted) request and response between the client and the decoy server. To date, TapDance is the only Refraction Networking scheme that has been deployed at ISP scale, during a pilot our project conducted in 2017 [8].

TapDance has two important limitations. First, if the connection stays open too long, the decoy will timeout and close the connection. Since its TCP state will be out-of-sync with the station and client, this would create an obvious signal for censors, allowing them to block the client's future connections. While we select decoys that have longer timeouts, this is nonetheless on the order of 20–120 seconds. Second, the client cannot send more than a TCP window of data, since after this upload limit the decoy server will start responding with stale acknowledgements. To overcome these, at the cost of added complexity, we multiplex long-lived sessions over multiple short-lived connections (see Section 3.2).

Beyond TapDance, other Refraction schemes have proposed ways to make it harder to detect or fingerprint proxy connections. Slitheen [2] is a Telex-like scheme that perfectly mimics the timing and packet size characteristics of the decoy, making it harder for censors to block based on web fingerprinting or other classification techniques. Rebound [7] and Waterfall [20] suggest ways to reflect client requests off of decoys, which enables stations that only see downstream (server to client) traffic. MultiFlow [17] provides ideas to adapt Refraction protocols to use TLS 1.3 [23], as most use prior versions and will need to be updated once older protocols are disabled. Finally, Conjure [9] is a recent Refraction protocol that utilizes unused IP addresses at ISP partners to simulate proxy servers at *phantom hosts*. Since phantom hosts are more numerous and cheaper to establish than real proxy servers, censors may find them more difficult to detect and block, particularly if each is only used ephemerally by a single client.

3 Deployment Architecture

In this section, we describe the architecture of our deployment, including the TapDance stations deployed at Merit Network and our client integration with Psiphon.

3.1 Station Placement

Building on a relationship formed during our 2017 TapDance pilot [8], we deployed TapDance stations at a mid-sized regional ISP, Merit Network, which serves several universities and K-12 institutions in the Midwestern United States. Merit has numerous points of presence, routers, and peering points. Although previous work on Refraction Networking generally assumes that any traffic that transits an AS is observed by station, in practice there are constraints that keep this from being true. As pointed out by Gosain, et al. [10], stations capable of processing high line rates are expensive (although our costs are at least an order of magnitude smaller than their estimates, which were aimed at Tier 1 ISP). There are additional costs, from finding rack space in the points of presence, to engineering time to deploy and manage the stations.

Due to these constraints, we used only a small number of stations, placing them opportunistically, and adding more as needed to maintain traffic coverage. Our deployment initially consisted of three stations. A fourth was added when a change in routing caused a large fraction of incoming traffic to arrive along a network path that did not have a station. Three stations use 4x10 Gbps and one uses 2x10 Gbps, for a total capacity of 140 Gbps. Typical peak utilization is about 70 Gbps, as shown in Figure 1. The four stations observe approximately 80% of the ISP's incoming packets, but some paths still do not pass any of them, leading client connections to fail and retry with a different decoy server.

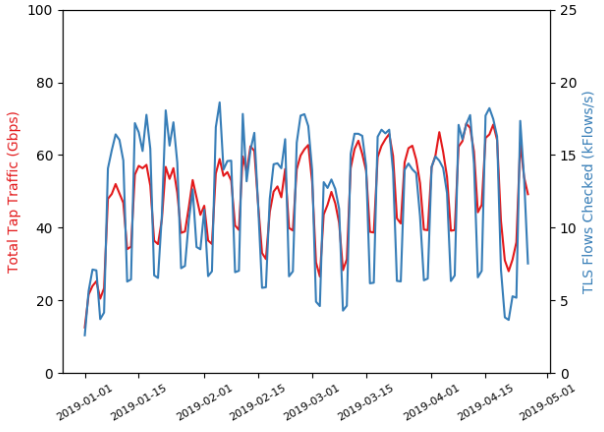

Fig. 1. Traffic at Stations. Our deployment uses four TapDance stations with capacity to ingest a total of 140 Gbps of ISP traffic. Utilization peaked at about 70 Gbps during our measurements. To identify connections from TapDance clients, the stations examined 5,000–20,000 TLS flows/second during a typical week.

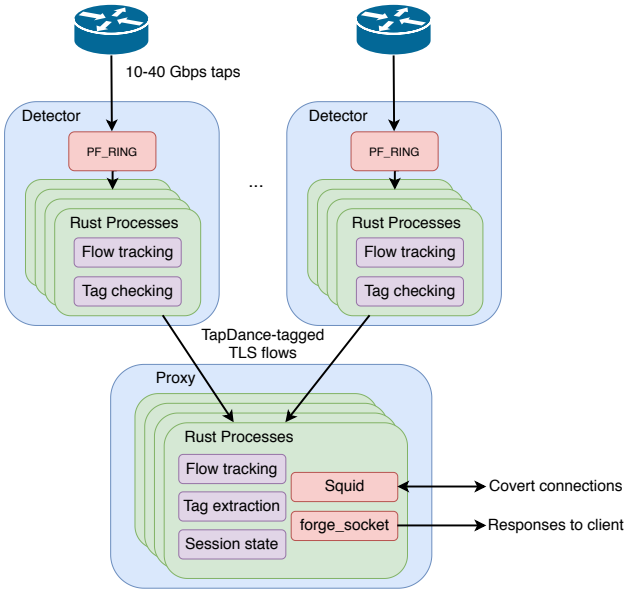

Fig. 2. Multistation Architecture. Light-weight detectors collocated with ISP uplinks identify TapDance flows and forward them to a central proxy. This allows a TapDance session to be multiplexed across connections to any set of decoys within the ISP, regardless of which detectors the paths traverse.

3.2 Station Design and Coordination

The original TapDance design considered only single stations running in isolation [27]. While this makes sense for a prototype, there are additional complexities when scaling the protocol to even a mid-sized ISP.

The primary complicating factor is the need to multiplex traffic over many flows. We term a single connection to a decoy a *TapDance flow*, and a longer-lived connection over our transport to a particular destination a *TapDance session*. Multiplexing works by having clients choose a random 16-byte session identifier, which is sent in the first TapDance flow to the decoy site. On the station, this first connection sets up the session state and connects the client to the covert destination. Before the decoy times out or the client sends data beyond the decoy's upload limit, the client closes the flow and opens a new one with the same session identifier. The station then connects the new flow to the previous session, giving the client the appearance of an uninterrupted long-lived session to the covert destination.

When each station operates independently, every flow within a session has to be serviced by the same station. During our 2017 pilot, we achieved this by having clients use the same decoy for the duration of a session [8]. However, this approach is unreliable when routing conditions are unstable and subsequent flows can take different paths, which led us to adopt a different station architecture for our long-term deployment. Instead of acting in isolation, stations at multiple uplink locations coordinate, so that sessions can span any set of decoys within the ISP. Figure 2 shows a high-level overview of this architecture.

We split the station design into two components: multiple *detectors* and a single central *proxy*. Detectors located at the ISP's uplink locations process raw traffic and look for tagged TapDance flows. When a tagged flow is identified, its packets are forwarded using ZeroMQ [12] to the central proxy running elsewhere in the ISP. The central proxy maintains session state, demultiplexes flows, and services the TapDance session.

Detectors ingest traffic using the PF_RING high-speed packet capture library [21], which achieves rates from 10–40 Gbps. PF_RING allows us to split packet processing across multiple (4–6) cores on each detector while ensuring that all the packets in a flow are processed by the same core, reducing the need for inter-core communication. To identify TapDance flows, the detectors isolate TLS connections and perform a cryptographic tag check on the first TLS application data packet using Elligator [1]. Depending on its network location, each detector typically processes between 300 and over 16,000 new TLS flows per second.

Once a TapDance-tagged flow is observed, the detector forwards it to the central proxy by sending the flow's TCP SYN packet, the tag-carrying application data packet, and all subsequent packets in the flow. The proxy thus only receives packets for flows that are related to TapDance connections. The proxy runs multiple processes on separate cores, and, as with the detectors, the forwarding scheme ensures that all of a session's flows are handled by the same process.

For each TapDance session, the central proxy maintains a connection to a local HTTP proxy server, which the client uses to connect to covert destinations. (In practice, Psiphon clients simply use it to make a long-lived encrypted connection to an external proxy server operated by Psiphon, so our central proxy does not see actual user traffic.) To communicate with the TapDance client, the central proxy uses a custom Linux kernel module named forge_socket to initialize a socket with the IP/TCP parameters from the client–decoy connection. This lets the central proxy call `send` and `recv` on the socket to produce and consume packets in the TapDance flow, as if it were the decoy server.

One drawback of this multistation architecture is that packets are received from the client at a different network location (the detector) than where they are sent

to the client (the central proxy). This could potentially be used by censors to infer the presence of TapDance, by observing TTL, timing, or network arrival point differences. We have not seen evidence of censors exploiting this (or any other technique) to block TapDance so far. However, if it becomes necessary, the proxy could forward packets back to the detector corresponding to each flow and inject them into the network there.

3.3 Client Integration

To make our deployment available to users who need it, we partnered with a popular censorship circumvention app, Psiphon, which has millions of users globally. We integrated TapDance support into Psiphon's Android and Windows software and distributed it to a cohort of 559,000 users in nine censored regions.

Psiphon clients support a suite of transport protocols and dynamically select the best performing, unblocked transport for the user's network environment. Our integration with Psiphon benefits Psiphon users by giving them access to a greater diversity of circumvention techniques, and it has allowed our team to focus on protocol implementation and operations rather than user community building and front-end development. In the future, our deployment could be integrated with other user-facing circumvention tools in a similar way.

From a user's perspective, Psiphon looks exactly the same with or without TapDance enabled. The app does not expose which transport it is using, and there are no user-configurable options related to TapDance. Users simply install the app, activate it as a system-wide VPN, and enjoy uncensored web browsing.

Psiphon ships with several transport modules. When a circumvention tunnel is needed, Psiphon attempts to establish connections using all available transports. Whichever successfully establishes the connection first is then used, while connections made by other transports are discarded. This selection algorithm provides optimal user experience by prioritizing the unblocked technique with the lowest latency.

Our TapDance deployment is available as one of these modular transports for a subset of Psiphon users. Since our overall capacity is limited by the size of Merit's network and the number of available decoys, Psiphon has prioritized enabling TapDance in aggressively censored countries and networks. However, since Psiphon does not track individual users, the granularity of this distribution is coarse. The Psiphon TapDance user-base was fixed during the measurement period analyzed in this paper, but we have subsequently enabled it for users in several additional countries facing censorship.

Our client library, gotapdance, is written in Go, as is Psiphon's app, which greatly simplified integration. The gotapdance library provides a `Dialer` structure that implements the standard `net.Dialer` interface. It specifies `Dial` and `DialContext` functions to establish connections over TapDance to arbitrary addresses and returns a TapDance connection as a standard `net.Conn` object. Implementation of standard interfaces simplifies integration by providing a familiar API, and it improves modularity, allowing Psiphon to reuse existing code.

While this interface makes establishing TapDance connections with gotapdance straightforward, there are two functions that library consumers like Psiphon may need to call first. The first is `gotapdance.EnableProxyProtocol`, which modifies TapDance requests so that the TapDance station sends the HAProxy PROXY [26] protocol header to the destination address before starting to tunnel the connection. This header includes the IP address of the client, which Psiphon's servers check against an anti-abuse blacklist before discarding. All Psiphon transport modules conform to this behavior. Second, library users need to call `gotapdance.AssetsSetDir` to specify a writeable folder in which the library can persistently store updates to its configuration, including the list of available decoys.

To facilitate testing, Psiphon worked with us to create a version of their application that exclusively uses the TapDance transport. We use this version for automated testing with a continuous integration (CI) system. On any code change to the TapDance library, the CI system runs a suite of tests and builds Android and command-line versions of the app for manual testing.

3.4 Operations and Monitoring

Operating a distributed deployment requires thorough performance monitoring, so that our team can quickly respond to component downtime, detect censorship events or blocking attempts if they occur, and understand the effect of engineering changes on overall throughput and reliability. We rely on a system of logging and analysis technologies that aggregate information from each individual station.

Detectors track and report the number of packets checked, the traffic flow rates, and the current number of live sessions, among other data points. The central proxy produces metrics that allow us to associate flows with sessions and monitor their number, duration,

throughput, and round-trip latency. Data from each station is collected by Prometheus [22] and visualized using Grafana [11] to provide a real-time view of the health of the deployment.

The system has also been instrumented to prevent overloading of decoys and to alert the project team when an outage occurs. Long-term data is stored using an Elastic Stack [6] instance to further aggregation and evaluation. This monitoring architecture allows us to quantify the performance, resilience, and (when any station fails) adaptability of the overall system.

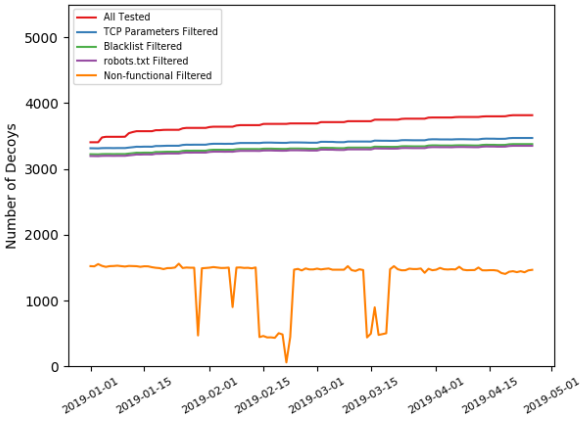

Fig. 3. Decoy Discovery. Over the course of our study we attempted to discover new decoys each day. TLS servers in Merit's network are discovered through port scanning and tested in four stages before being published to clients. The step with the greatest loss is when we attempt to connect to them with a real client. Troughs in this line correspond to station outages.

3.5 Decoy Selection

Two important and related questions that a Refraction Networking deployment needs to address are where to place stations and how clients will choose which decoy websites to contact. The former question has largely been studied in the context of AS-level deployment to transit ISPs [4, 10, 13, 19, 25]. The latter has not been explored in detail, though Houmansadr et al. [13] do consider how effective randomized client-side probing can be for a given level of deployment.

Our deployment has characteristics that are significantly different from those explored in previous work. Our partner ISP is a regional access ISP, acting as a primary point of Internet access for its customers. At the same time, it has many peering and client connections at numerous points of presence, so capturing *all* traffic that transits it is challenging. Additionally, many of its customers do have other network interconnections on which traffic arrives, complicating the choice of decoys.

The nature of our deployment means that we can enumerate the customers that are served by Merit and their corresponding AS numbers. This allows us to survey all of the publicly reachable websites operated by Merit's clients by scanning port 443 across this address range and noting the domain names on certificates returned. The process yields roughly 3,000 potential decoy sites. Not all of them are usable by TapDance, however. There are four potential problems:

1. Does the site have a large enough initial TCP window to allow sending a sufficient amount of data?
2. Does the site support a large enough timeout to keep a TCP connection open after a partial request has been sent?
3. Does the site support and select a TLS cipher suite compatible with our TapDance implementation? Currently, we support only AES GCM ciphers, and our clients send a Chrome 62 TLS fingerprint. If the server selects a different cipher, we will be unable to use the connection.
4. Does traffic from to the server pass a station?

To address the first three problems, we implemented a testing script that checks the size of the TCP window, the duration of timeouts, and the selected TLS cipher suite. (Our approach to the last problem is to simply rely on station placement to attempt to capture most traffic entering our partner ISP.) Figure 3 shows the results over the course of our test period.

Our test script filters the available websites by first discarding servers with a TCP window less than 15 KB or a timeout less than 30 seconds. Next, we apply a manual blocklist of subnets with servers that behave poorly when used as decoys, due to server reliability or throughput issues. We then remove domains that include a specific user agent in their `/robots.txt` file, as described in opt-out instructions provided to decoys via a URL in the user-agent header sent by clients. (As of February 2020, only two domains had opted out.) Finally, we make a test connection with our client to ensure that a usable cipher suite is chosen and the connection is functional.

This last step removes the most decoys, due to cipher suite selection and station placement. We occasionally see large drops in the success rate, due to station downtime or fault in our decoy discovery infrastructure. To prevent temporary drops from affecting the deployment's overall availability, we do not distribute decoy list updates to clients if more than 30% of the previous day's decoys are no longer present.

4 Evaluation and Measurements

In this section we evaluate our deployment of TapDance. Following a 2017 pilot [8] and about 18 months of further development and testing, we launched the current deployment with a small cohort of Psiphon users in October 2018. We slowly increased the number of users until entering full-scale production in early December 2018 and have been operating continuously since then.

The analysis we present here is based on data from the first four months of 2019. We chose this period because our partner ISP predominately serves academic institutions, and this is a representative semester, a high-load period for the provider. It was also a period when we did not alter our user base, and no major engineering changes were pursued. This affords a steady-state view of aspects of the deployment within our control.

We observe changes in TapDance use over this period in spite of our consistency. These changes are due primarily to changes in censor behavior. A significant fraction of our usage occurred in a single censoring country, and such actions taken to disrupt Internet freedom affect how circumvention protocols are used. The most striking example is the major censorship event in mid-April that caused TapDance usage to more than double.

We note that our evaluation period ends after traffic returned to normal in late-April. At that point, we restarted our central proxy for maintenance and inadvertently disabled some of the logging that we use for our analysis, resulting in a gap in our data.

4.1 Psiphon Impact

During our observation period, TapDance was one of several circumvention protocols offered by Psiphon, and we served approximately 10% of traffic for Psiphon users who had TapDance available. Daily usage varied significantly in a weekly pattern from 5,000 to 15,000 users.

When other Psiphon transports were more heavily blocked, TapDance usage increased, peaking at above 40% of client traffic and 25,000 daily users, as shown in Figures 4 and 5. The largest peak resulted from censorship of more direct, single-hop circumvention techniques that were previously very reliable. With these protocols blocked, clients automatically switch to other available transports, including TapDance, causing an apparent influx of clients. We also observe the opposite effect: when other, lower-latency circumvention protocols were temporarily unblocked, users tended to select those, causing a decrease in TapDance usage, as seen in early March.

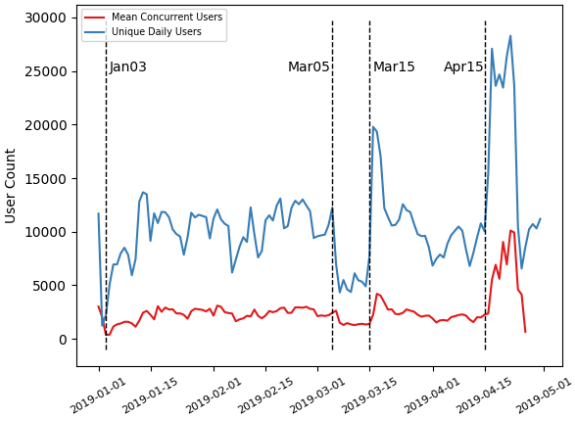

Fig. 4. User Counts. Our user base was composed of tens of thousands of people in censoring countries. During the evaluation period, the deployment averaged about three thousand concurrent users and ten thousand daily unique users. However, this varied over time. We indicate dates of significant changes in observed censor behavior in the country that had the largest share of our users. **Jan03**: Censor reduced restrictions on Domain Fronting. **Mar05**: Censor reduced restrictions on direct protocols. **Mar15**: Direct single-hop protocols and some domain fronting providers were blocked. **Apr15**: New censorship capabilities demonstrated, restricting several long-reliable techniques.

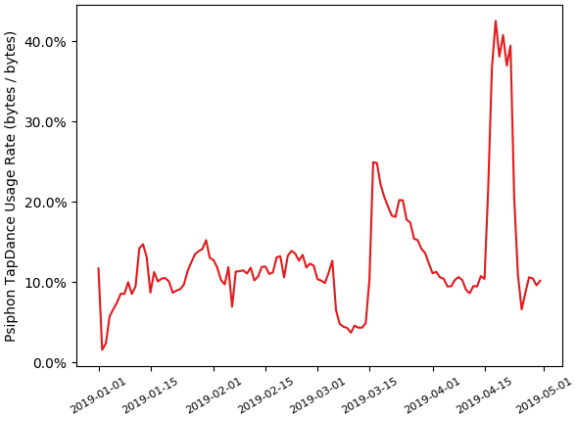

Fig. 5. TapDance Usage Rate. We show what fraction of bytes transmitted and received by Psiphon users with TapDance available were carried via TapDance. This graph has the same features as those in Figure 4, indicating that the size of our user base is driven by how frequently Psiphon clients select TapDance.

4.2 Client Performance

To evaluate the quality of service that users experienced and the overall utility of our deployment, we analyze the network performance that clients achieved and the overall throughput of the system, ensure there was no degradation of service under load, and confirm that no clients monopolized system resources.

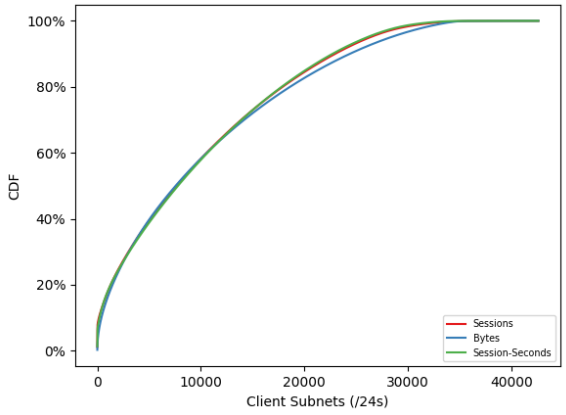

Fig. 6. Resource Consumption by Client /24. We show the CDF of consumption of sessions, data transmission, and decoy utilization by client /24 subnet. We see no major difference in any of these lines, indicating a uniform allocation of resources per session over our user base. Half of all resources are used by 8,000 client subnets and 1% are used by 10,000.

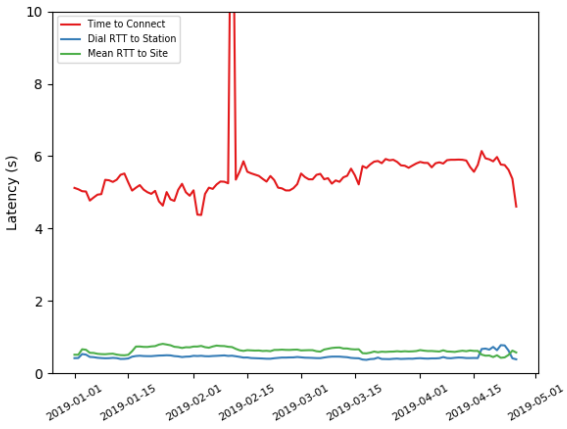

Fig. 8. System Latency. Latency was steady throughout our measurement period, but the time to connect spiked in early February when a brief outage caused persistent clients to wait a long time for the system to return before they could reconnect. We also note that There was an increase in dial RTT towards the end of our evaluation period, which is detailed in Figure 20.

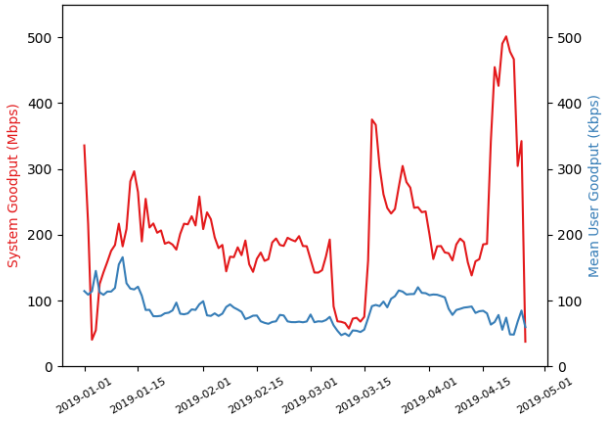

Fig. 7. Goodput. Over our study period, useful data transmission peaked at around 500 Mbps, during the final week. User traffic per session corresponded to approximately 100 Kbps per user throughout the study period, and did not decrease under increased user load.

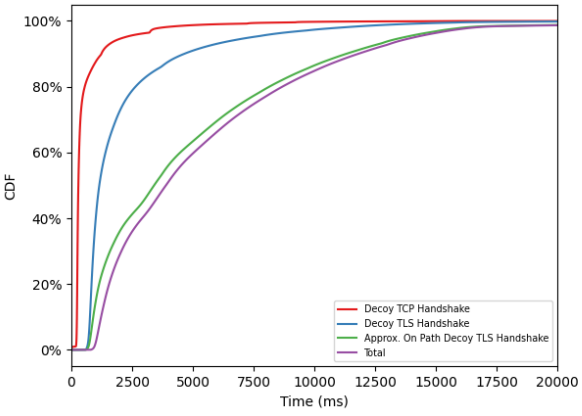

Fig. 9. Connection Establishment Time. We plot the CDFs of connection establishment times to identify which step is the largest contributor. Stations failing to pick up for the first decoy a client tries and causing a new decoy to be used is the source of the greatest delay in both the worst and typical cases.

To protect user privacy, we do not track individual clients. Instead, we log the subnet (/24) from which the client appeared to connect. Even at this coarser granularity, we do not observe monopolization of any system resources by small sets of clients, as shown in Figure 6. We analyzed the number of bytes up, bytes down, and session duration, and find that these all correlate strongly, showing a similar usage of bytes and duration per session across all client subnets (bytes up and down, $r = 0.88$; bytes up and duration, $r = 0.82$; bytes down and duration, $r = 0.74$). This suggests that most users receive similar performance.

Over this study period, the average user data throughput did not drive or depend upon system utilization; rather, total system throughput depended upon the number of clients using the system, as shown in Figure 7. This indicates there was likely available capacity during much of our evaluation period that went unused due to Psiphon's protocol selection strategy.

Clients saw an average round-trip-time (RTT) under one second for the entirety of our evaluation period. We note this may be due to survivorship bias in our measurement strategy, as clients that took longer may use another transport, closing connections before we receive

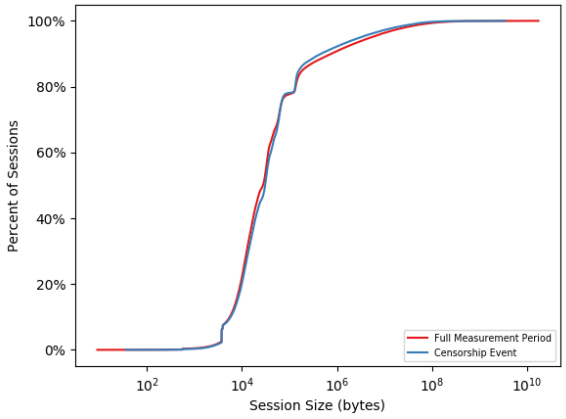

Fig. 10. Session Size CDFs. Session size distributions were similar over our full measurement period and during the censorship event in the final week. Over 90% of sessions transmitted and received more than 10 kB. We did not see a large effect on session size during the increased utilization of the censorship event.

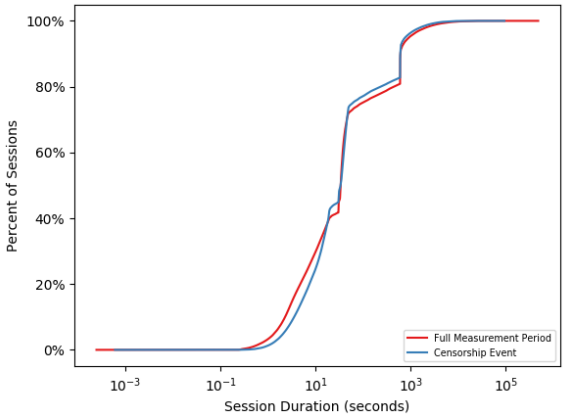

Fig. 11. Session Duration CDFs. Like session size, session duration followed similar distributions during both the full measurement period and the censorship event. The median connection lasted 50 seconds. We also did not see a large effect on session duration during the increased utilization of the censorship event.

their metrics. The average time to transfer the first byte of a connection was considerably longer, often over five seconds, as shown in Figure 8. This is due to the large number of round trips required before useful data is delivered. This effect is multiplied in cases when the first attempt to communicate with a decoy fails. We observe that, for many decoys, our stations are not reliably on the connection path for all clients. This forces the client to try multiple decoys before a connection succeeds, as shown later in Figure 14. This effect contributes the majority of our time to first byte, as supported by Figure 9.

In spite of the long time to first byte, our clients observe reasonable session lengths in bytes and time, shown in Figure 10 and Figure 11 respectively. Client connections lasted approximately 50 seconds and transfer 20kB in a median session. Moreover, during the censorship event, under increased load, we did not observe degradation in these performance distributions.

Since Psiphon operates as a transport for the censored user, very often for connections to websites over HTTPS, it has limited visibility into the range of time taken to download pages or even individual files. However, we have reason to conclude that TapDance at least does not degrade user experience for Psiphon clients. First, our deployment is only used as a transport when it has the lowest latency among available techniques. This means that our long connection establishment time and high latency are only experienced by clients when they are the fastest functional option. Second, Psiphon has a feedback mechanism that is monitored for complaints of poor performance, and they did not see significant increases where our technique was deployed.

4.3 Decoy Impact

One concern that we and our ISP partner share is avoiding burdening sites used as decoys. It is important not to overload them with excessive traffic or large numbers of concurrent connections. In Figure 12, we show that our impact on decoys was generally small. In Figure 13, we see that half of all decoys typically had two or fewer concurrent connections during our observations.

In spite of the top 10% of decoys seeing the most use, the peak usage is restricted to below an average of 50 concurrent connections over the busiest day. As shown in Table 1, over the entire measurement period, the busiest decoy saw an average of 13.24 concurrent connections, corresponding to 12.32 MB of uncensored

Table 1. Top Decoys. We show the ten most frequently used decoys during our 115-day measurement period. These top decoys are well distributed through the ISP's address space, except for ranks 5, 9, and 10 (*), which are in the same /24.

Rank	Mean Concurrent Connections	Connections	Transfer Rate (MB/Day)
1	13.24	163,991	12.32
2	12.76	167,277	10.74
3	12.00	167,144	10.70
4	10.75	167,507	9.14
5*	10.70	128,691	13.29
6	10.68	151,699	8.04
7	10.48	127,980	12.89
8	10.42	161,146	9.15
9*	10.41	127,971	13.40
10*	10.34	127,948	12.67

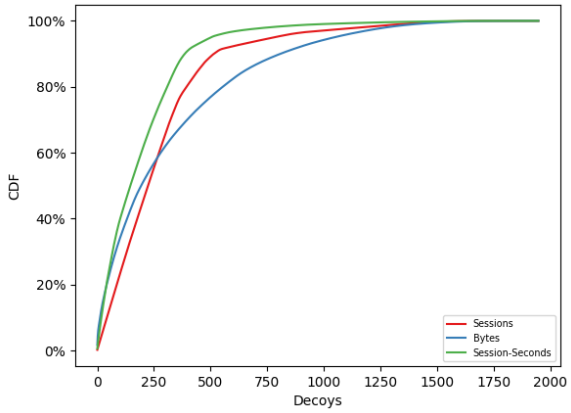

Fig. 12. Decoy Distribution. The CDFs of sessions, bytes, and session-time across decoys are not as similar as those in Figure 6, indicating a difference in utilization across these resources. Session-time and bytes were particularly focused on the most popular decoys, as indicated by steepness near the left of the plot.

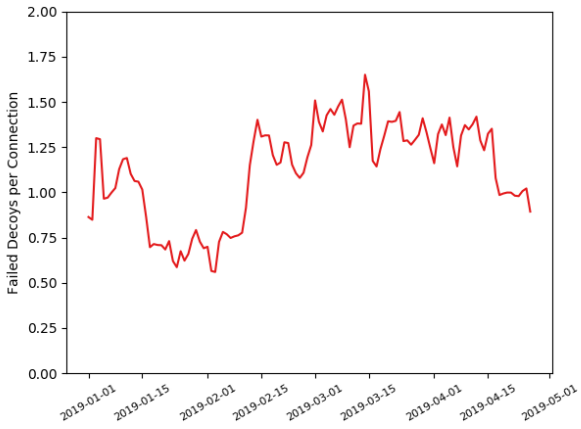

Fig. 14. Client Decoy Failures. Clients select a random decoy from a provided list; after multiple attempts, they retry with a different decoy. There was about one failure for each successful connection. Which decoys failed was inconsistent across clients. Predicting which clients can use which decoys is challenging.

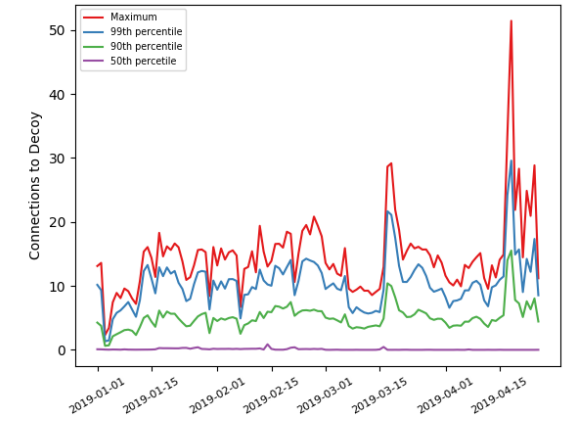

Fig. 13. Decoy Quantiles. Traffic was unevenly distributed across decoys. While the median decoy typically had two or fewer clients connected simultaneously, the 90th percentile saw around ten.

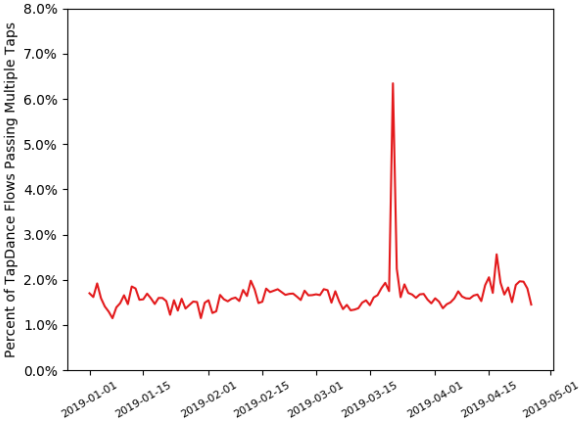

Fig. 15. Duplicate Detection. We observed consistently low duplication of client requests to decoys, with the exception of one peak in late March corresponding to a routing change.

traffic per day. We note that decoys do not see the bulk of proxied traffic, since most bytes are sent from the stations to clients, though decoys do receive (and ignore) data and acknowledgements from clients.

We find evidence that our mechanism for communicating and performing decoy opt-out worked. Two domains included us in their `robots.txt` files and were automatically excluded from future decoy lists.

4.4 Station Performance

During the evaluation period, we did not observe any of the stations saturating its computation, disk, or memory resources. Clients reported the number of failed decoy connections they attempted before each successful connection, as shown in Figure 14; this metric does not increase during period with heavy usage, supporting a conclusion that stations were able to keep up with client load. We only observe a dip when a routing configuration was temporarily used that more reliably routed decoys past our stations. This change and reversion was outside of our direction.

Another routing change briefly caused an elevated number of requests to pass two stations, resulting in a spike in client requests observed twice (Figure 15). This exercised our multistation architecture's ability to allow connections to span multiple detectors. The baseline rate was about 1.5%, indicating that this capability is also useful during normal operations.

Nonetheless, the average number of failed decoys for each successful connection was above one, indicating that, for every successful connection, clients typically tried and failed at least once. While clients automatically retry until unsuccessful, this metric shows that improvements to decoy selection or station placement would likely positively impact time to first byte.

In order to quantify the impact of the volume of traffic we serve on the quality of service, we looked for correlations between the traffic volume and quality metrics. In Figure 16, we show a scatter plot where each dot represents a day of operation. The horizontal axis indicates the number of connections we handled that day, normalized by the mean value over our measurement period. The vertical axis is a similarly normalized scale of reliability, as quantified by client connection success rate and the amount of data transmitted per connection. The amount of data transmitted per connection and the total connections served have a Pearson's R of -0.11. Upon linear regression, we find that the correlation effect size is insignificant ($p = 0.05$). The connection success rate and total connections served correlate with a Pearson's R of 0.41. Interestingly, this demonstrated a small positive correlation (linear regression slope $= 0.068 \pm 0.030$).

Both of these correlations support the claim that we operated within the bounds of what our system can reliably handle; neither shows statistically significant negative correlation. Qualitatively, we also observe that our plots do not display a "tipping point" after which we would see significant performance degradation.

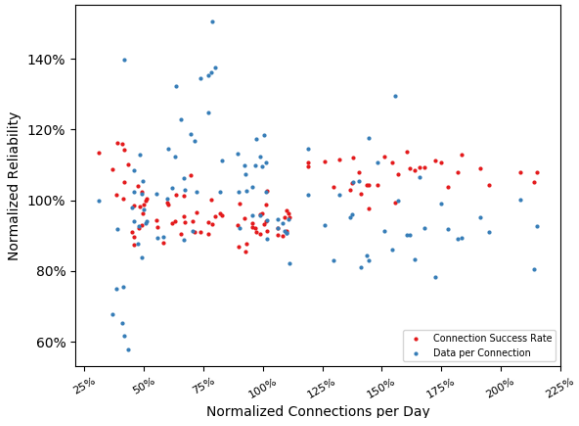

Fig. 16. Reliability Under Varied Load. We did not find evidence of negative correlation between the system's traffic volume and quality metrics. In this plot, each point represents a day of operation, with connection volume indicated along the horizontal axis and the normalized client connection success rate indicated along the vertical axis. The lack of correlation supports that the deployment operated within its capacity limits.

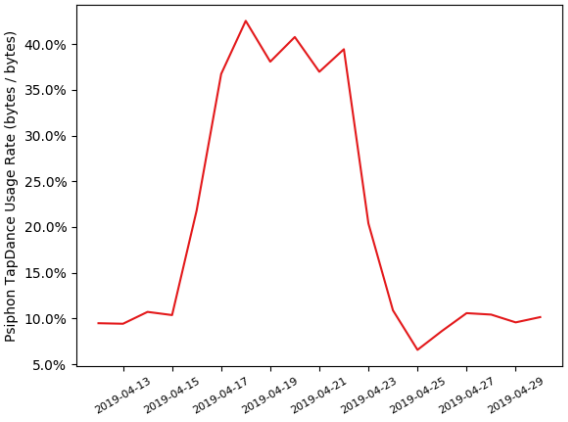

Fig. 17. TapDance Usage Rate Under Censorship. This plot shows the same metric as Figure 5 during and after the censorship event at the end of the study period.

4.5 Censorship Event

During the final two weeks of our measurement period, we observed a significant increase in usage. This was the result of the deployment of new censorship techniques in the country where most of our users were located. While many Psiphon transports were disrupted by the censor, our TapDance deployment remained accessible, so we received increased load as Psiphon clients automatically switched to using TapDance. This accounted for a 4× increase in the fraction of TapDance-enabled Psiphon clients' traffic that used our system, as shown in Figure 17.

The spike in traffic was not limited to only a few client subnets. Figure 18 plots the CDF of client usage across subnets during the censorship event and has a similar shape to the distribution across the entire measurement period. Although some client subnets maintained longer connections, we do not see a change in the byte per session distribution in any subnets. This indicates that the change in traffic was due to an increased frequency of use, not because a small set of subnets were using significantly more bandwidth. In Figure 19, we see that during the censorship event, most subnets exhibited increased use of our system. However, some saw a drastic decline, particularly those unaffected by the change in censorship. Even the small increase in connection establishment time while our deployment was under censorship load, shown in Figure 20, might have caused otherwise unaffected Psiphon clients to shift to other transports. This further suggests that efforts to decrease establishment time (such as better decoy selection or more aggressive timeouts for failed decoys) may increase the share of clients that select TapDance.

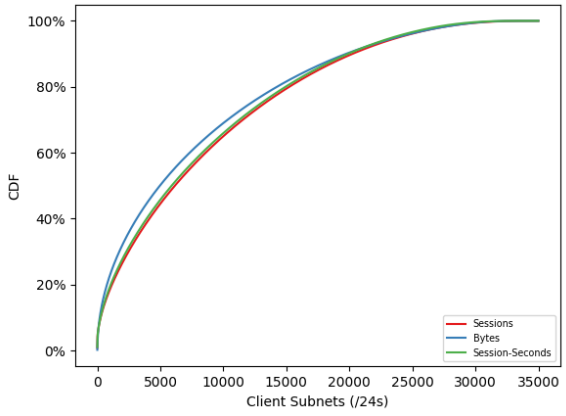

Fig. 18. Client Subnet Distribution Under Censorship. We replot Figure 6 during the censorship event, highlighting an increase in concentration of some subnets, particularly in byte usage, though half of all bytes were spread over 14% of subnets.

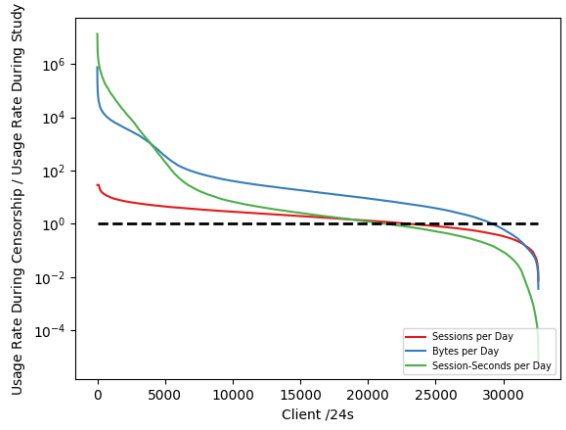

Fig. 19. Client Distribution Change Under Censorship. During the censorship event, most clients showed an increase in sessions, bytes, and connection time. However, some clients showed decreased usage, especially those in regions unaffected by the event.

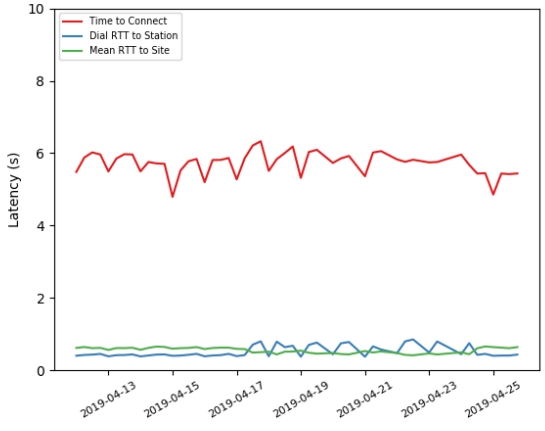

Fig. 20. System Latency Under Censorship. We highlight latency metrics during the censorship event. Dial RTT increases slightly, but it does not noticeably impact total time to connect.

5 Discussion

In this section we discuss some of the lessons we learned about the unique challenges of deploying a Refraction Networking scheme in practice, and what this means for the future prospects of the technology.

5.1 Where Refraction Provides Value

Our TapDance deployment serves only a relatively small share of Psiphon's traffic. It was enabled for a small fraction of Psiphon users and served about ten percent of those users' traffic over our measurement period. Given the costs of developing and operating such a complex distributed system and navigating the institutional relationships necessary to deploy it, this may not seem like a worthwhile investment on its face. However, Refraction Networking played a critical role during the censorship event, when it kept thousands of users connected that would otherwise have been successfully censored. Maintaining connectivity during periods of heightened censorship is vital, as it allows updates to circumvention software and news relevant to the censorship event to reach populations who need them. Our experience indicates that advanced circumvention techniques such as TapDance can effectively provide such a lifeline.

5.2 Deployment Costs

Other than research and development, the major cost of our ISP deployment was hardware, which cost about 6,000 USD per site (i.e., four detector stations, plus the central proxy), for a total of 30,000 USD. This was used for purchasing hardware such as commodity 1U servers, network cards, optics, etc. Merit donated the bandwidth and hosting costs; it estimates that the co-location cost (i.e. rack space and power) of the current deployment would be about 13,000 USD per year, and the bandwidth cost for a 2 Gbps upstream connection would be 24,000 USD per year. In addition, Merit assigned an engineer with an effort of about 40% FTE to help with system maintenance, management, and operation. Many of these costs are specific to Refraction schemes, which have the unusual requirements of co-location at network operators and elevated engineering time for system maintenance. While engineering costs will go down with stability and scale, the cost of operating core system infrastructure at ISPs incurs costs beyond those of other circumvention approaches.

5.3 Addressing Partner Concerns

Merit and other network operators we have engaged with have expressed several very reasonable concerns concerning deploying Refraction Networking. We review some of the most prominent ones here and discuss how we mitigated the issues.

Will the deployment impact normal production traffic?
TapDance is specifically designed to avoid interference to an ISP's normal operation. Since TapDance stations only observe a mirror of traffic, their outages will not affect regular commodity traffic flowing through the ISP. Although an ISP's network might become saturated if too many TapDance clients started using it, we provisioned our deployment to avoid this: Merit observed 70 Gbps of commodity traffic (out of a 140 Gbps capacity), while our user traffic added only about 500 Mbps, much less than a problematic level. We also have the ability to modulate usage at a coarse granularity if necessary to address capacity concerns, though we have not had to use this capability in practice. Proxied connections originate from address space managed by Psiphon, which also manages responding to abuse.

Will the deployment affect user privacy?
Stations observe network traffic to identify connections from Refraction users. To protect privacy and reduce risks, stations only need to receive traffic that is already end-to-end encrypted via TLS. This does not remove all privacy risks in the case of a compromised station—IP addresses and domains from SNI headers and certificates would be visible—but exposure is greatly reduced compared to a full packet tap. To reduce privacy risks for TapDance users, clients connect to Psiphon proxies through an encrypted tunnel over the deployment, so stations cannot see the content users request or receive.

How will decoy websites be affected?
Our clients used 1500–2000 decoy websites at times during the measurement period. These sites do indeed see a small increase in load, but, since client traffic is spread across all available decoys, the burden on individual sites should be negligible. We monitored the number of connections to each decoy to ensure it was under a conservative threshold (see Figure 13). We also offered a simple way for sites to opt-out of being used as decoys, but only two sites did so during our evaluation period.

Will censors attack the ISP in retaliation?
Our ISP partner was also concerned that censors might try to attack it or its customers in retaliation for hosting TapDance. Such a response would be extraordinary, though not completely unprecedented [18]. We have not observed any evidence of retaliatory attacks taking place, but Merit mitigated the risk by proactively contracting with a DDoS protection service provider.

5.4 Lessons and Future Directions

Some of the lessons we learned in the course of deploying TapDance may be useful for those looking to improve upon Refraction Networking techniques or manage further deployments. In particular:

1. TapDance's complexity, and the need to work around TCP window and server timeout limitations, created ongoing engineering and operational challenges, as well as bottlenecks to some aspects of the deployment's measured performance. Simplifying the design would enhance the deployability of future Refraction approaches.
2. Decoy attrition did not pose a significant challenge for our scale of deployment, at least over the 18 months of operation to date. Few servers opted out, and none reported operational problems.
3. Router-level route prediction for client-to-decoy connections is important to the performance of Refraction techniques, but this problem is complex when deploying in networks like Merits, in which it is prohibitive to place stations on every incoming path to many decoys. Our current approach is overly simplistic—about half of client connection attempts fail to pass a station due to routing behavior—and it would be even less effective for deployments farther from the edge of the network. Further work is needed.
4. ISP partnerships remain a bottleneck to the growth of Refraction Networking. Despite TapDance's ease of deployment relative to earlier Refraction schemes, partners remain concerned about effects on decoy sites and other operational risks. Partnership with Tier 1 or Tier 2 ISPs may also raise more concern about impacts on decoy websites, as the lack of a direct customer relationship between the ISP and site operators may make ISPs less comfortable with an opt-out model. Significant investments in partner-building will continue to be necessary in order to grow our deployment.
5. Although larger Refraction Networking deployments would have more capacity and be more prohibitive to block, even relatively small deploy-

ments such as ours can be surprisingly valuable as a fallback technique to keep users connected during periods of heightened censorship. This suggests that a deployment composed of a constellation of mid-sized network operators like Merit could be a powerful anticensorship tool, and it would require far less investment than the Tier-1 scale installations envisioned in early Refraction research.

Some of these lessons are incorporated into the design of Conjure [9], a new Refraction protocol in which the importance of decoys backed by real websites is reduced. Rather, beyond initial registration, decoys are produced from address space with no web server present. This approach holds promise as a practical way to reduce complexity and performance bottlenecks and obviate concerns about impacts on decoy sites. However, we note that Conjure does not resolve all of the remaining challenges to Refraction Network's deployability, and there is a continued need for research.

6 Conclusion

This paper presents results from the first deployment of a Refraction Networking scheme to enter continuous production with real-world users. Our experience running TapDance in production for 18 months demonstrates that Refraction Networking can play a vital role in providing connectivity, even when censors increase their efforts to block other circumvention techniques. We hope our work will inform the design and operation of further advanced anticensorship systems, which are more important than ever for people living under censorship in countries worldwide.

Acknowledgements

We are grateful to the many people who helped bring Refraction Networking out of the laboratory and into production, including Sol Bermann, Rosalind Deibert, Fred Douglas, Dan Ellard, Alexis Gantous, Ian Goldberg, Aaron Helsinger, Michael Hull, Rod Hynes, Ciprian Ianculovici, Adam Kruger, Victoria Manfredi, Allison McDonald, David Robinson, Joseph Sawasky, Steve Schultze, Will Scott, Colleen Swanson, and Scott Wolchok, and to our outstanding partner organizations, Merit Network and Psiphon. This material is based in part upon work supported by the National Science Foundation under grants CNS-1518888 and OAC-1925476.

References

[1] D. J. Bernstein, M. Hamburg, A. Krasnova, and T. Lange. Elligator: Elliptic-curve points indistinguishable from uniform random strings. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[2] C. Bocovich and I. Goldberg. Slitheen: Perfectly imitated decoy routing through traffic replacement. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[3] C. Bocovich and I. Goldberg. Secure asymmetry and deployability for decoy routing systems. *Proceedings on Privacy Enhancing Technologies*, 2018(3), 2018.

[4] J. Cesareo, J. Karlin, M. Schapira, and J. Rexford. Optimizing the placement of implicit proxies, June 2012. Technical Report, Available: http://www.cs.princeton.edu/~jrex/papers/decoy-routing.pdf.

[5] L. Dixon, T. Ristenpart, and T. Shrimpton. Network traffic obfuscation and automated Internet censorship. *IEEE Security & Privacy*, 14(6):43–53, 2016.

[6] Elastic, Co. Elastic stack and product documentation. Available: https://www.elastic.co/guide/index.html.

[7] D. Ellard, A. Jackson, C. Jones, V. Manfredi, W. T. Strayer, B. Thapa, and M. V. Welie. Rebound: Decoy routing on asymmetric routes via error messages. In *IEEE Conference on Local Computer Networks (LCN)*, 2015.

[8] S. Frolov, F. Douglas, W. Scott, A. McDonald, B. VanderSloot, R. Hynes, A. Kruger, M. Kallitsis, D. Robinson, N. Borisov, J. A. Halderman, and E. Wustrow. An ISP-scale deployment of TapDance. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2017.

[9] S. Frolov, J. Wampler, S. C. Tan, J. A. Halderman, N. Borisov, and E. Wustrow. Conjure: Summoning proxies from unused address space. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[10] D. Gosain, A. Agarwal, S. Chakravarty, and H. B. Acharya. The devil's in the details: Placing decoy routers in the Internet. In *Annual Computer Security Applications Conference (ACSAC)*, 2017.

[11] Grafana Labs. Grafana documentation. Available: https://grafana.com/docs/.

[12] P. Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly, 2013.

[13] A. Houmansadr, G. T. K. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[14] A. Houmansadr, E. L. Wong, and V. Shmatikov. No direction home: The true cost of routing around decoys. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2014.

[15] M. Kan. Russia to block 9 VPNs for rejecting censorship demand. PCMag, June 7, 2019. Available: https://www.pcmag.com/news/russia-to-block-9-vpns-for-rejecting-censorship-demand.

[16] J. Karlin, D. Ellard, A. W. Jackson, C. E. Jones, G. Lauer, D. P. Mankins, and W. T. Strayer. Decoy routing: Toward unblockable Internet communication. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[17] V. Manfredi and P. Songkuntham. Multiflow: Cross-connection decoy routing using TLS 1.3 session resumption. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2018.

[18] B. Marczak, N. Weaver, J. Dalek, R. Ensafi, D. Fifield, S. McKune, A. Rey, J. Railton, R. Deibert, and V. Paxson. An analysis of China's Great Cannon. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2015.

[19] M. Nasr and A. Houmansadr. Game of decoys: Optimal decoy routing through game theory. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[20] M. Nasr, H. Zolfaghari, and A. Houmansadr. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[21] Ntop. PF_RING. Available: http://www.ntop.org/products/pf_ring.

[22] Prometheus: Monitoring system and time series database. Available: https://prometheus.io.

[23] E. Rescorla. The transport layer security (TLS) protocol version 1.3. RFC 8446, 2018.

[24] D. Robinson, H. Yu, and A. An. Collateral freedom: A snapshot of Chinese Internet users circumventing censorship, 2013. Available at https://www.upturn.org/static/files/CollateralFreedom.pdf.

[25] M. Schuchard, J. Geddes, C. Thompson, and N. Hopper. Routing around decoys. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[26] W. Tarreau. The PROXY protocol versions 1 & 2, 2017. Available: https://www.haproxy.org/download/1.8/doc/proxy-protocol.txt.

[27] E. Wustrow, C. M. Swanson, and J. A. Halderman. TapDance: End-to-middle anticensorship without flow blocking. In *USENIX Security Symposium*, 2014.

[28] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *USENIX Security Symposium*, 2011.