Proxies as Sensors: Measuring Censorship of Refraction Networking in Iran

Abdulrahman Alaraj
Computer Science
University of Colorado Boulder
Boulder, Colorado, USA
Computer Science
Prince Sattam Bin Abdulaziz University
Al-Kharj, Riyadh, Saudi Arabia
abdulrahman.alaraj@colorado.edu

Eric Wustrow
University of Colorado Boulder
Boulder, USA
ewust@colorado.edu

Abstract

Iran expends significant effort to block circumvention tools. This paper offers a unique view into the ongoing cat-and-mouse game between censors and circumventors in practice. We provide evidence of how Iran blocks certain components of our proxy system to discourage use.

While prior work has suggested Iran's censorship is centrally managed, our experience suggests this may be an incomplete view: we observe censorship can vary dramatically across Iran's major ISPs both in the technique and timing of blocking, suggesting there may be ISP-specific deployments or configurations in use.

Our work reveals both the utility and shortcomings of measuring censorship from the circumventor's perspective. Prior work has struggled to measure censorship in locations similar to Iran, due to the difficulty of obtaining vantage points, particularly in mobile or residential networks that serve most users. Our data reflects a type of ground-truth in this regard: the scale of our measurements reveals the type of network interference real users in Iran face in their everyday use. Nonetheless, our perspective is also limited: passive measurements of aggregate user traffic can be noisy, leaving us with a narrow view of how censorship works or evolves. Our goal with this work is to augment prior techniques, providing yet another tool for researchers in the effort to support Internet freedom in Iran and globally.

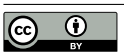
CCS Concepts

• **Networks** → *Network security*; • **Social and professional topics** → *Censorship.*

Keywords

anticensorship, proxies, network security

ACM Reference Format:
Abdulrahman Alaraj and Eric Wustrow. 2025. Proxies as Sensors: Measuring Censorship of Refraction Networking in Iran. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '25), August 25–29, 2025, Hanoi, Vietnam.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3708821.3733879



ASIA CCS '25, Hanoi, Vietnam

ACM ISBN 979-8-4007-1410-8/25/08
https://doi.org/10.1145/3708821.3733879

1 Introduction

Internet censorship is a global problem that impacts billions of people worldwide [23]. Yet despite its reach, it is still challenging to study this pervasive problem. Often it is difficult for researchers to rent servers in censored countries, as there can be limited vendors (particularly for smaller countries), or local laws or policies that require identification for purchasing hosting. Even when servers can be rented, they may be located in datacenters which can experience different censorship than residential Internet users [2].

In this work, we offer a unique perspective to study censorship. We partner with a deployed censorship circumvention tool with millions of users in Iran, and collect aggregate and anonymized information from the server side to learn about the techniques and effectiveness of modern censors in blocking the proxy. Since users of the tool are likely experiencing censorship and come from predominantly residential ISPs, we get to see a unique and realistic view of the types of blocking that takes place for a typical user.

Our method differs from traditional censorship measurements, in that we focus on understanding the censorship techniques of proxies, rather than on what specific websites or content is blocked. As we only observe traffic destined to our proxies, we can learn what techniques a censor uses to block proxies (*e.g.* TCP injection, IP blocking, etc), but not the overall blocklist of domains or resources that the censor attempts to ban. Nonetheless, we believe our complimentary perspective is valuable, and provides insight into a censor's technical capabilities and responsiveness.

We apply our methodology to measure censorship in Iran, a nation characterized by pronounced Internet restrictions [4, 5, 23, 24]. Recently, the Iranian government has intensified its censorship measures and enforced Internet shutdowns following protests ignited by the tragic death of Mahsa Amini while in police custody in September 2022. These restrictions have propelled Iranian citizens towards using proxies to circumvent the censorship.

We partner with Psiphon, a popular circumvention proxy used by millions around the world, with a high concentration of users in Iran. Working with Psiphon, we deploy a proxy based on Conjure [17], a Refraction Networking scheme that places proxies at ISPs in non-censoring countries. Conjure users register with the system and then connect to an agreed-upon unused IP addresses (phantoms) that routes past the deployed ISP proxy, which picks

up the connection. To the censor, this connection looks like a normal one to an arbitrary IP within a large network. In our Conjure deployment, the large network spans hundreds of thousands of IPs across several ISPs in North America.

Our deployment allows us to measure how Iran censors proxies like ours. In contrast to previous studies on censorship in Iran [5], we find that censorship in Iran varies across ISPs, and is not monolithic across the country. Furthermore, we find that while some ISPs change their censorship strategies together, other ISPs are more independent, suggesting a mix of semi-centralized and distributed censorship emerging in Iran.

Our work suggests that more investigation is needed to understand the nature of decentralization of censorship in Iran, and we offer suggestions to circumvention tool researchers and developers that will help make proxies more resilient against Iran's censorship.

We make the following contributions:

- We present a novel way to measure censorship from the proxy operator perspective, complementing prior work that uses in-country vantage points, which are difficult to obtain and not reflective of actual user experience.
- We use our technique to conduct the largest measurement of proxy censorship in Iran to date, studying the censorship dynamics against hundreds of millions of proxy connections over 20 months.
- We reveal that censorship in Iran differs across ISPs, suggesting a new aspect of decentralization in how the country censors.
- We discover and deploy new proxy-side techniques for circumventing proxy censorship in Iran, successfully evading blocking in some ISPs.

2 Conjure Deployment Background

Conjure [17] is a Refraction Networking scheme that leverages unused IP address space that route past an ISP-deployed proxy. We deployed Conjure at three university-sized ISPs: Merit Networks, the University of Michigan, and the University of Colorado Boulder, and have been operating this deployment since January 2021. We partner with Psiphon, a large circumvention tool provider that operates a mobile app with hundreds of millions of downloads according to the Google Play app store as of February 2025.

We integrated the Conjure client into the Psiphon client app for both Android and iOS, and enabled it for several million users in several countries, including Iran. The Psiphon client implements several different proxy techniques in addition to Conjure, including direct proxies, domain fronting [14], and other transports. When a client connects, it chooses a random set of transports and simultaneously connects to each of them. The first successful connection is used by the client, and the remaining connections are discarded. This type of connection race is helpful for censorship resilience: a censor must block all available transports to successfully block this proxy.

Our study focuses on the Conjure transport of Psiphon. In Conjure, a client first **registers** with the system, sharing a secret, and a covert destination they wish to connect to (*e.g.* a blocked website or direct proxy endpoint). Using the shared secret, the Conjure client and station derive a phantom IP address that is within the configurable list of agreed upon networks.

After registering, the client makes a TCP connection to the agreed-upon phantom IP address. This connection passes by the ISP where a Conjure station is deployed. Since the station is aware of the client and phantom (via registration), the station can pretend to be the phantom host, and respond to the connection. In our Conjure deployment, the client must send an authentication tag corresponding to the secret shared during registration, or the station will remain silent after picking up the connection in order to thwart active probing attacks [3, 12, 18, 38]. There are several transport protocols that Conjure supports, but in the simplest mode, the client simply sends a 32-byte tag directly.

Registration Conjure supports several ways for clients to register. The first way is directly via the *API* registrar at an HTTPS RESTful endpoint. However, since this would be easily blocked by a censor, it is not used directly in practice. Instead, Psiphon tunnels these API requests over a more expensive (but reliable) *domain fronting* [14] connection. A second way users can register is via a *decoy* registrar, in which they send their registration encoded in the TLS ciphertext to a decoy site that the Conjure station can observe (similar to TapDance [39]). Finally, users can also register via a *DNS* registrar, where the user encodes their registration in a domain name `<registration>.root-domain`, where `root-domain` is a domain that we control the nameserver of. The client then resolves this using DNS-over-HTTPS to communicate the registration information.

2.1 Client Configuration

Clients are given a configuration file that lists which networks can be used as phantoms. As there are multiple ISPs where Conjure is deployed, each with different network prefixes (subnets) of varying size that route past them, these networks can be **weighted** in the configuration to influence how often clients select each network. Clients will also download the latest Client configuration (based on the version or "generation"), allowing us to update the weightings or networks used remotely.

While in some cases we can update the code running on Conjure clients (with assistance from Psiphon), this process is time consuming, requiring coordination and integration testing to ensure that code updates work with other transports in the Psiphon client. For this reason, we generally refrain from using code updates for one-off experiments, and instead rely on updating the client configuration file. Providing new configuration files allows us to update the set of phantom subnets (and their associated weights), as well as what protocols the client will use to connect. Finally, we can dynamically provide an arbitrary **prefix** to registering clients, which they will prepend to their phantom connection. We explore using these prefixes to evade censorship in Section 5.

Bias bug. Early in our Conjure deployment, we discovered a bug in our subnet weighting logic. Rather than select phantom IPs uniformly from a particular subnet, this bug caused clients to select some IPs much more frequently than others. This does not cause any connectivity issues, but it does cause clients to select some subnets and even individual IPs far more frequently than expected. This serendipitously gives us an accidental yet revealing

experiment: As we detail later, we observe some ISPs in Iran block more aggressively the IPs and subnets that receive a higher number of connections due to this bug.

3 Datasets

Measuring censorship from the proxy side is challenging. We have limited control over clients, and access to only one side of the connection, making it difficult to apply traditional methods like sending a known probe from a client and measuring if it makes it unaltered to the server endpoint. On the other hand, our proxy is used by real users, in realistic networks that are often difficult to get vantage points in, such as residential and mobile ISPs. We begin by outlining the datasets that we have access to from the proxy operator side (server-side).

Each of these data sources must balance utility against user privacy. Our agreements with Psiphon keep us from storing identifying data about clients, including their IP address, to help mitigate the worry that a censor finds a way to view these logs and uses them to identify users. For this reason, we anonymize and aggregate the data we collect, in an effort to collect useful data while preserving user privacy.

3.1 Conjure Logs

The Conjure station logs generic information about each connection it receives. This includes the client's ASN and geo IP country, the destination phantom IP, the duration of the connection and the count of bytes uploaded and downloaded. We also log transport type, indicating for example if this connection sent a specific protocol prefix at the start of the connection.

However, for a connection to be logged, the client must successfully send the 32-byte connection ID agreed upon during registration. This means that if the censor blocks a connection immediately after the TCP handshake (or blocks the handshake entirely), Conjure will not log information about the connection. As a result, our logs capture only real Conjure connections: we do not log Internet scanners or other non-Conjure users that attempt to connect or probe the phantom IPs. But it also introduces a limitation: a censor that blocks early in a connection will prevent us from logging information about it.

These logs are also limited in that they only reveal application-level behavior. As the server application uses the standard send/recv system calls, it cannot log packet-level interference, such as the presence of TCP RSTs, duplicate TCP data, or other connection interference techniques used by prior work [36]. Nonetheless, our logs provide a valuable means to broadly understand application-level interference.

3.2 Traffic Captures

To better understand the dynamics of censorship at the packet level that is missed by Conjure logs, we wrote a packet capture tool in Rust that allows us to capture a sample of anonymized flows. Our tool observes the interface between the Conjure detector (which ingests raw packets from the ISP network tap) and the Conjure application (which accepts and responds to Conjure phantom connections)[1]. Because the detector only forwards packets that are part

[1]See Appendix E for more details

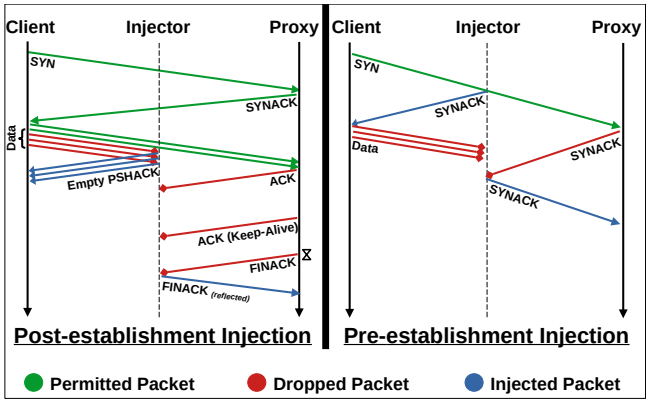


Figure 1: Injection Types — Injection types we observe from our VPS vantage point in Iran are either of these two types depending on the destination. We observe variations of these two type in our dataset from our packet capturing tool for clients in Iran.

of registered Conjure connections, our tool will also only observe Conjure-related traffic. For this traffic, when our capture tool observes a TCP SYN from the client, we begin tracking the 4-tuple flow. Within each flow, we record packets in both directions, replacing the client IP with a deterministically random IP that is within the same BGP-announced prefix. The random IP is derived from the client's IP, BGP prefix, and an HMAC with a runtime-generated secret that is not recorded, allowing each run to have a deterministic mapping (*e.g.* the same client IP will map to the same randomized IP), but this may differ between runs. This determinism allows us to associate packets within the same flow, and reveals the user's geographic location (*i.e.* country) while keeping the true IP of the client private.

The tool writes packets using the PcapNG format with original TCP/IP headers information, yet anonymizes client's IP and port. It also writes the client's ASN, subnet and country information, which we obtain from MaxMind [28], as optional frame comments in the recorded packets.

We run our capture tool once a day during the peak usage of our proxy in Iran, and capture up to the first 100 packets in each flow. Our capture tool also limits the number of total flows per AS each day, set to 10,000 flows per AS per day. Our goal with this tool is to provide us with a daily sample **snapshot** that captures the packet-level dynamics that are occurring early in connections (where censorship may occur), without recording every connection or deanonymizing users.

3.3 Vantage Point in Iran

To help us understand dynamics of censorship, we use one VPS vantage point in the Shiraz region in Iran. This VPS is in an AS that is distinct from our typical clients, meaning the censorship it experiences may not be reflective of that faced by real users. While additional or more representative vantage points would be useful, we find it is difficult to obtain vantage points in Iran, particularly in residential and mobile cell phone networks. Nonetheless, our vantage point is helpful in generating a high-level understanding

Table 1: Top Iranian ASes — We collect a sample of anonymized connection traces over a 9-month period, from July 2023 to April 2024 using our packet capture tool described in §3.2. We discover two main types of ASes: packet-injecting (Injector) and IP-based blocking.

AS Name	ASN	Type	Connections	% of Total From Iran	IPv4 Subnets	Injector	Injection Circumventable	Injection Rate (%)
TCI	58224	Fixed-line	1,486,120	33.48	714	✓	✓	21.7
IranCell	44244	Mobile	1,281,432	28.86	26	✓	×	52.3
MCCI	197207	Mobile	990,913	22.32	58	×	-	0
Mobin Net	50810	Fixed-line	251,478	5.66	38	✓	×	40.8
RighTel	57218	Mobile	150,071	3.38	34	✓	×	20.4
Asiatech	43754	Fixed-line	54,886	1.23	43	✓	×	50.2
Pishgaman	49100	Fixed-line	53,059	1.19	72	✓	×	40.8
Parsonline	16322	Fixed-line	33,994	0.76	18	×	-	0
NGSnet	39501	Hosting	27,592	0.62	17	✓	×	38.4
Shatel	31549	Fixed-line	21,297	0.47	54	×	-	0
Other	(183 ASNs)	-	88,063	1.98	1029	×	-	0

of network and censorship phenomenon that we can then search for in our other datasets.

4 Proxy Censorship in Iran

In this section, we use Conjure, a Refraction-based proxy, as a case study for proxy censorship in Iran and focus on the top 10 Iran ASes by connections, as listed in Table 1. We describe several longitudinal measurements and short-lived experiments, and share the observations we make from our available datasets in order to understand if—and how—Conjure is censored in Iran.

Between July 2023 and February 2025, our capture tool (§3.2) collected tens of millions of Conjure connections from clients in the top 10 Iran ASes—a small sample from over 800 million logged Conjure connections (§3.1) from Iran during the same period. These (anonymized) packet captures and logged connections reveal a surprising trend: different ASes in Iran deploy different censorship techniques.

This observation challenges the view from prior studies [5] that suggest Iranian censorship is centralized. We find that some Iran ASes censor proxies using **packet injection**, **IP blocking**, **mid-connection dropping** and/or **protocol allowlisting**.

4.1 Censorship by Packet Injection

In order to understand how packet injection works and what triggers it, we use both our in-country vantage point (§4.1.1) as well as our traffic capture tool (§3.2).

Despite some implementation differences between injecting ASes, we conclude that this injection technique serves the same purpose: interfering with proxy traffic. This interference affects TCP connections to certain IPv4 phantom addresses and occurs within the first several packets of a TCP connection.

Figure 1 shows the types of injections we observe. We now first describe our analysis from our singular vantage point in Shiraz, Iran, and then second, look for similar types of censorship across our top 10 ASes in Iran using our capture tool dataset.

4.1.1 Detection from a Vantage Point. From our vantage point in the Shiraz region we run several experiments during the first half of 2023 to help understand the nature of packet injections in this AS. We note this AS is distinct from our client ASes, but the lessons we learn here apply to some—but not all—other client ASes in Iran.

Post-establishment Injection To understand how this type of packet injection works, we have our in-country vantage point connect to a heavily-used IPv4 phantom subnet with a Conjure client (after performing the requisite registration). This allows us to observe if registration or phantom connection are blocked, and observe packets on both sides of the connection (client and ISP station we control).

We find evidence of network interference for some of the connections from our vantage point to a heavily-used phantom subnet. Specifically, we observe extra packets on both the client and server. Since we control both endpoints (and capture packets accordingly), we can easily tell these injections come from an on-path network censor. These injections interfere with the connection, preventing progress from being made in the TCP connection.

What Triggers this Injection? We observe that these injections occur for specific phantom IP addresses, while other IPs within the same subnet experience no injections. However, once we see a phantom IP address experience injections, all connections to that phantom IP from our vantage point exhibit censorship, suggesting the censor has marked that IP as a proxy.

Once the censor blocklists a phantom IP address, we observe the censor tracks TCP connections to that IP by inspecting the first 5 packets, while being agnostic about most of the semantics of the TCP protocol. For example, the censor maintains the state of a TCP connection by tracking the 5-tuple information but does not validate sequence or acknowledgment numbers. We test this by sending out-of-sequence TCP segments, and observe that the censor still injects packets.

Reminiscent of prior work that revealed a protocol allowlist in Iran [8], if a TCP connection to a blocklisted IP starts with a TCP packet with the `SYN` flag set and one of the remaining 4 TCP packets contains a censor-defined signature (discussed later) of an allowlisted application-layer protocol, then the connection is allowed, and no injection occurs. Otherwise, the connection enters a *residual censorship* phase for 60 seconds as follows: the censor drops all packets after the fifth one in the connection and bidirectionally injects response packets to both the client and server, by *reflecting* any packets (besides bare ACKs) back to the sender, and removing

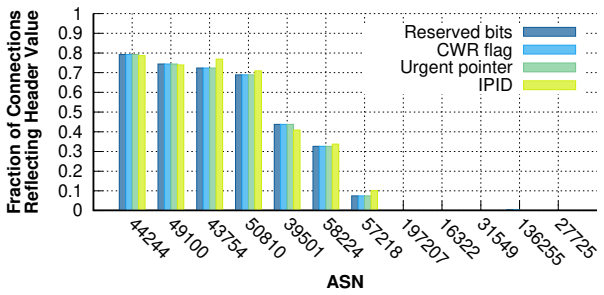


Figure 2: Reflection Experiment — To confirm the censor is reflecting our packets, we configured our Conjure station to set several IPv4 and TCP header fields to constant values in packets it sends. We observe clients in some Iranian ASes (packet injecting ones) reflect these values, indicating the presence of a middlebox that reflects our packets back to us (with modifications) in these ASes. In other ASes, we rarely (if ever) see these values, suggesting censorship is done by other means.

any payload. For instance, if the client sends a PSH-ACK packet containing a payload, the censor will inject an empty PSH-ACK response back to the client. An empty PSH-ACK is unusual, because the PSH flag indicates the provided data should be immediately delivered to the application, but in this case there is no data.

How Are Injected Packets Crafted? The injected packets are crafted in the following manner: First, the censor copies values from the TCP and IPv4 headers of the first dropped packet (the fifth packet) and sets the ACK flag; it also sets the TCP sequence number of the injected packet to the acknowledgement number of the dropped packet, and sets the acknowledgement number of the injected packet to the sequence number of the dropped packet plus the payload length. If the dropped packet had the FIN flag set, the injected packet's acknowledgement number is incremented by 1, adhering to the normal semantics of the TCP CLOSE mechanism [1].

The censor also copies the TTL and IPID values from the IPv4 header, allowing for a convenient injection detection. We also observe that during the *residual censorship* period, any additional packets sent in the same flow by either endpoint resets the 60-second residual censorship timer. Thus, a client that continues to retransmit would elongate the residual censorship window for that connection.

What Exempts Injection? In order for a TCP connection involving a blocklisted address to be permitted, one of the first 4 packets following the SYN packet must contain a payload of an allowlisted protocol, *e.g.* a TLS Client Hello. We tested several common application-layer protocol prefixes in order to identify which ones are allowlisted. Namely, we test HTTP (GET and POST), TLS and SSH. Only TLS and SSH protocols were permitted from our vantage point regardless of the destination port. Appendix B contains more details on how this censor detects allowlisted protocols, and its network hop location from our VPS vantage point.

Pre-establishment Injection In some cases, an IP will receive injections even during the TCP handshake. To understand how this type of injection works, we test using over 500,000 random IPv4 addresses that are geo-located outside of Iran and run a TCP application on port 443. We obtained these addresses from an Internet-wide scanning machine hosted at a university campus in North America, using ZMap [13]. The purpose of this experiment is to identify which addresses are subject to blocking by pre-establishment injection.

From our vantage point in Iran, we send a PSH+ACK[2] packet to port 443 on each address. Each packet contains a fixed, arbitrary payload, fixed sequence and acknowledgement numbers and fixed IPID to allow for a convenient injection detection. Simultaneously, we capture traffic to and from these address at our vantage point. Because we have not established a TCP handshake with these hosts, a TCP compliant host should either ignore or respond with a TCP RST packet.

However, we detect injections matching the fingerprint described previously (*e.g.* reflected IPID) on 43,479 addresses scanned (8.7% of those scanned). Over 70% of these affected addresses are in two Akamai ASes, AS16625 and AS20940, potentially indicating attempts to block domain fronting proxies hosted there.

Does this Injection Type Have a Different Fingerprint? Contrary to the criteria described in the post-establishment injection, these injections were in response to the first, and only packet sent to the scanned addresses, suggesting that these injections serve as a TCP establishment interception (had we sent SYN packet) and ensures no data packet reaches these destinations, while allowing the censor to collect the payloads that were intended to be sent to these addresses. We scanned a small sample[3] of these destinations using SYN packets and indeed find that we receive injected SYN+ACK packets in response. Figure 1 (Pre-establishment Injection) shows how this type of injection works. Although some ASes in Iran have overlapping characteristics with the injector we encounter from our vantage point, we later show that not all injectors are the same, even those that are in the same AS (§C). Additionally, we observe that injectors change technique over time. For instance, we observe the set of protocols that exempt a connection from blocking injection has changed over time at our vantage point: originally, we observed HTTP GET exempts blocking from our vantage point, but after May 2023, it no longer does. This suggests that the injectors continue to evolve, and perhaps are a developed version of the protocol filter first studied by Bock *et al.* [8].

4.1.2 Server-side Detection. After observing packet injections from our vantage point in Iran, we look for evidence of injections from other ASes in our Conjure datasets to see if real clients also experience the same kind of censorship.

We run two experiments and capture traffic for each one. In the first experiment that lasted an hour at the end of March 2024, we set the IPID value for all the packets we send from one of our Conjure stations to a non-zero constant value, x, using the nftables [30] utility. Typically, we expect a uniform distribution for the clients' IPID values (with the exception of zero being a notable outlier [35, 36]). However, when we set a fixed IPID in the station's outgoing packets, we observe that the three most common IPID values from clients for this experiment are x (the constant

[2]We use + to indicate that both flags are set in a single packet
[3]To avoid drawing unwanted attention to our vantage point

value we set) in the vast majority of packets from Iran, followed by 0 and finally $x + 1$ (which appears to be a signal of injection from another country). In the second experiment in mid-April 2024 that lasted an hour, also using the `nftables` utility, we fix three TCP header values in the packets we send as follows: we set the `CWR` flag, set the fourth bit in the reserved bits, and set the urgent pointer to a constant value.

Figure 2 shows the percentage of connections that reflect the fixed values we set during both experiments. We include client packets from two ASes outside Iran in our analysis, one in Myanmar and the other in Cuba, both of which are reported not to enforce censorship for Psiphon [25, 31] during both experiments. We see that not all ASes in Iran reflect IPv4 and TCP header values at the same rate, and some never do. Finally, when we examine the clients' packets that reflect the values we set, we observe `TTL` anomalies that suggest these reflected packets were sent by sources closer to us (in hop distance) compared to the actual clients, based on the `TTL` difference between the clients' original packets (*e.g.* `SYN`) and the reflected ones. We conclude that these reflected packets are indeed indications of censorship middlebox injections.

We also observe `SYN+ACK` packets from some clients in Iran, an unusual packet for a connecting client. Examining these packets reveals that their `IPID` value reflects the `IPID` value in the `SYN+ACK` our proxy sent, indicating the Pre-establishment Injection we saw from our vantage point in Section 4.1.1.

Moreover, if we send a data-carrying packet to a client in a censored connection with the `PSH+ACK` flags set, the censor will inject a packet with the `PSH+ACK` flags set (and the same `IPID`), but with a zero-length payload. In all cases, we observe that the censor clears the TCP options in all injection cases, despite being set in our original packets.

Heuristics for Detecting Injection from the Server. Our observations inform the heuristics we use to detect injections in our dataset of captured packets: if we detect a client sending `SYN+ACK` packet, a `PSH+ACK` packet with zero-length payload, or a `FIN+ACK` packet having the same (reflected) `IPID` as the `FIN+ACK` packet sent by the station due to connection timeout in a given connection, we label these as censored connections by packet injection.

We apply these heuristics to the anonymized packet traces we collect between July 2023 and April 2024. The results in Table 1 show differences in injection rates between injecting ASes. Notably, some ASes do not ever exhibit this censorship behavior, indicating differences in proxy censorship across Iran.

4.2 Censorship by IP Blocking

IP-based blocking is a widely-used censorship technique typically enforced by null-routing network traffic to the blocked IPs [27]. We also observe IP-based blocking in Iran (in addition to the packet injection mentioned above), and see that can be applied country-wide or on an AS level. In some cases, IP blocking appears to be applied in phases depending on how often those IPs are connected to. Our findings support the observation that proxy censorship in Iran is a mix of centralized and decentralized deployments.

4.2.1 Country-wide Deployment of IP-based Blocking. On April 22, 2024, we observed a considerable country-wide drop in the number of connections from Iran to our most heavily-used /24 IPv4 phantom subnet, shown in Figure 3. We also confirmed this blocking from our vantage point in Iran, observing that the block appears to drop `SYN` packets, indicating a country-wide IP block.

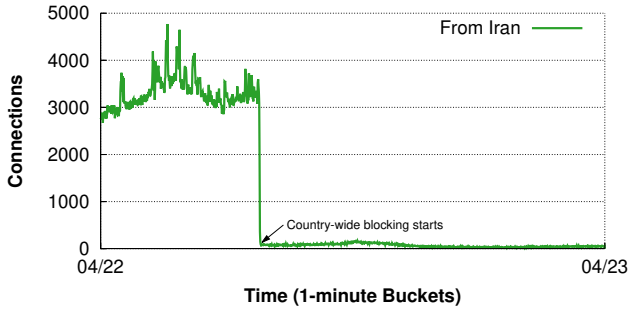


Figure 3: Country-wide Blocking Affecting our Heavily-used Phantom Subnet — The number of connections falls from over 1 M in the first 8 hours of the day (04/22/2024) to near-zero due to IP-based blocking.

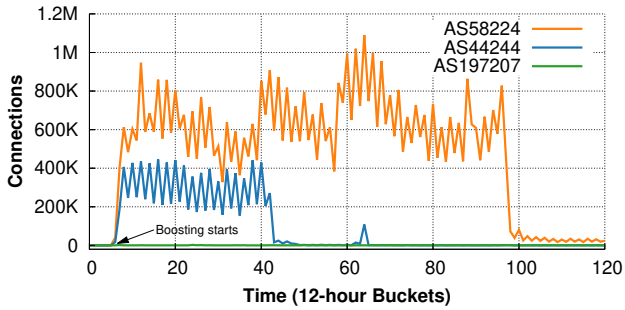


Figure 4: Boosted Subnet — We boost the weight of a lightly-used IPv4 phantom subnet and observe different incidences of AS-specific IP-based blocking, indicating a mixed proxy censorship deployment in Iran.

Curiously, this block did not extend across the full /24: two IPs were exempt (out of 256): one corresponds to a registration server and the other to an IP with a high bias. We do not know why this highly-used IP was left out from blocking, and note that other high bias IPs were blocked.

4.2.2 AS-specific Deployment of IP-based Blocking. Because our heavily-used /24 IPv4 phantom subnet was blocked in Iran, we decided to decrease its weight in Psiphon clients configuration and boost the weight of a different /24 IPv4 phantom subnet. Although this new /24 was within a /16 in the client configuration and had been receiving connections, those connections were small in quantity due to the lower weight assigned to the larger (/16) subnet, and consequently, the number of connections the /24 received was further smaller.

On May 25, 2024, we boosted the weight of this new /24 subnet, as shown in the spike of the number of connections from AS58224 and AS44244 in Figure 4. Surprisingly, AS197207 had already blocked this subnet before we pushed the updated weights to clients. It appears this AS blocked most of our phantom subnets, even those with minimal weights. We believe operators at AS197207 may have

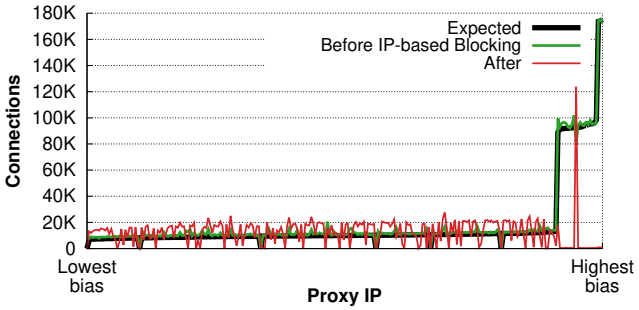


Figure 5: Effect of Address Selection Bias — An address selection bug (§2.1) biased the IPs clients connected to. AS197207 disproportionately blocked addresses to IPs its clients connect to more than others. The IPs shown are from our heavily used /24 phantom subnet.

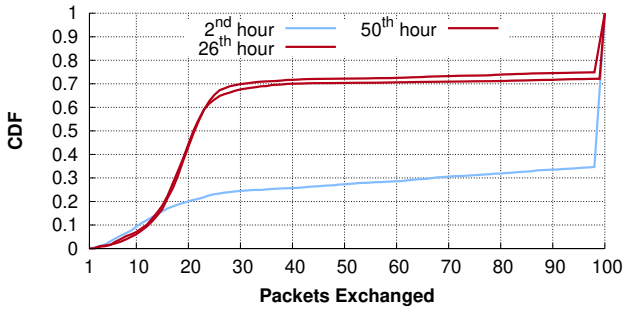


Figure 6: CDF of exchanged packets with clients in AS197207 on the 2nd (before mid-connection dropping begins), 26th and 50th hours after the experiment onset. The effect of mid-connection dropping on the number of packets exchanged is evident. We capture the first 100 packets in a subset of flows using our packet capturing tool that runs daily for a fixed hour.

obtained a list of our phantom subnets, which is publicly available, and blocked the addresses manually. We note this means they are blocking all hosts at several large universities in North America, even those unrelated from Conjure.

For the remaining ASes, we observed IP-based blocking of the new boosted subnet after less than three weeks of experiment onset in ASes 44244 and 57218. ASes 58224, 50810, 43754, 49100 and 39501 did not start IP-based blocking until about 50 days after the experiment onset. During the entire two months of the experiment, we did not observe any IP-based blocking by AS16322. Finally, AS31549 did not produce the number of connections needed for meaningful analysis.

4.2.3 Volume-Proportional IP-based Blocking. One of the largest ISPs, Mobile Communications Company of Iran (MCCI Hamrah-e Avval, AS197207), appears to have applied IP-based blocking in a phased way—blocking heavily-used IP addresses first, and then later blocking nearly all addresses. We note that clients tend to pick some proxy addresses over others due to an IP selection bias in the client logic that we later fixed (detailed in §2.1). Figure 5 shows the total number of connections per address in our most heavily-used /24 IPv4 phantom subnet 15 days prior to the first detection of blocking and 15 days after (from July 20 through August 3 and from August 4 through August 18 of 2023) using data from our packet capturing tool. We see that blocking appears to be proportional (or happen earlier for) higher-biased phantom IPs (those that are connected to more frequently), with the exception of one address. We are unsure why this single address (out of 256) was excluded from this pattern. Nonetheless, we find the order in which the IPs were blocked to be highly correlated to the rate they were connected to (see details in §D).

4.3 Censorship by Mid-connection Dropping

To confirm whether blocking was proportional to connections, we performed a follow-up experiment to artificially steer increasing amounts of traffic to different subnets, and observe which are blocked. Our hypothesis is that higher-volume subnets would be blocked more quickly than lower-volume ones.

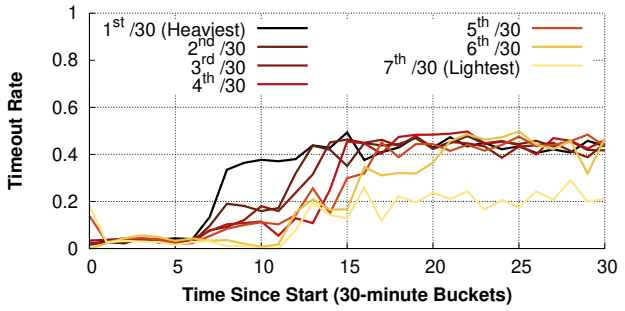


Figure 7: AS197207 blocks connections to proxies based on their perceived rate of connectivity—The more connections a proxy IP receives, the quicker it gets blocked.

On September 2, 2024, we used a new /27 IPv4 subnet that had not been previously used in our proxy before, to avoid contamination from past connection behaviors. We divide this /27 into 7 /30 subnets (leaving the 8th /30 unused) and assign increasing weights across the seven in an arithmetic sequence. The highest weight is equivalent to a weight that we observed sufficient to be blocked within a few hours in a prior experiment (§5.2). We provided IPs in these (weighted) subnets to clients via our bidirectional registration API, meaning the subnets were never placed in a public client configuration; only individual IPs were provided to individual clients when they registered, proportional to their subnet weighting. This avoids the address selection bias bug described in §2.1, and gives us direct control over the proportion of connections made to each subnet. We configure clients to connect on port 443 using the Min transport, a fully-encrypted protocol based on ObfuscatedSSH [26].

Figure 7 shows the timeout rate for connections from clients in AS197207 to the weighted /30 subnets. The time to the sharp increase in the percentage of connections that timeout for each subnet is directly proportional to the subnet's weight—the higher the connectedness to a proxy IP the censor observes, the faster it blocks it.

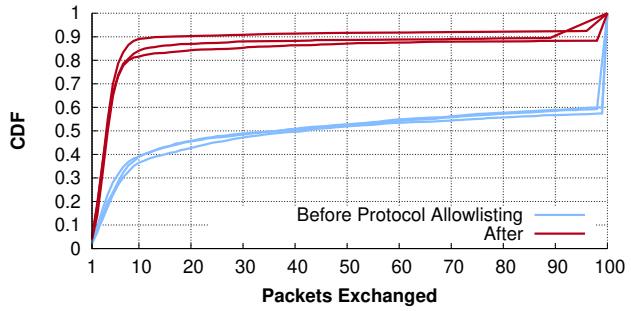


Figure 8: CDFs of exchanged packets with clients in AS197207 using Min transport, three days before and three days after the activation of the protocol allowlist, show a sharp increase in the share of clients exchanging only a few packets.

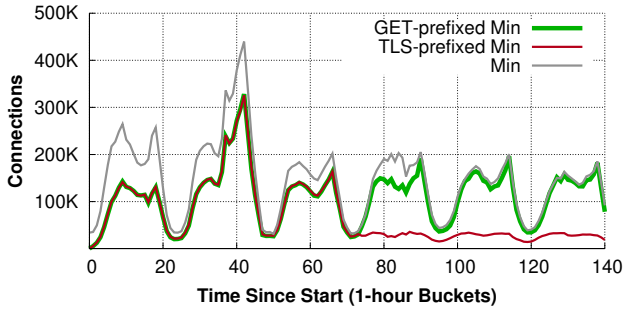


Figure 9: Different proxy censorship techniques are observed for different traffic types. Country-wide IP-based blocking affects phantom IPs that are connected to using the TLS-prefixed Min transport, whereas mid-connection dropping (see Figure 13) affects connections that use the non-prefixed Min transport. Surprisingly, prefixing connections with some application protocols (such as HTTP GET) avoids censorship.

To understand how connections to these subnets time out, we use our packet capturing tool[4] (see §3.2) and examine the number of packets in each connection before and after the increase in the connections timeout rate.

Figure 6 shows a CDF of exchanged packets between clients in AS197207 and phantom IPs in the 1st (Heaviest) subnet before blocking (2nd hour) and after (26th and 50th hours). We see a significant increase in the number of connections that exchange 15—25 packets in the 26th and 50th hours compared to the 2nd hour, indicating packets are dropped mid-connection. Recall that our packet capturing tool limits the number of packets it captures to 100 per connection and captures traffic once per day.

4.4 Censorship by Protocol Allowlisting

On November 12, 2024, we designed an experiment intended to trigger the mid-connection dropping effect we observed in AS197207 and AS16322 (§4.3). We configured clients to use three *uncontaminated* /24 IPv4 phantom subnets as follows. In the first /24, clients connect to phantom IPs using the Min transport. In the second /24, clients use an HTTP GET-prefixed Min transport, sending a GET request prior to the rest of the protocol. The third /24 has clients use a mixture of Min, and prefixed (including HTTP GET, HTTP POST, and SSH) transports, to test if the presence of multiple protocols influences censor behavior.

After two months of running this experiment, we observed a sharp increase in the timeout rate for connections to the Min-only subnet and to the Min-transported subset of connections to the mixed subnet, indicating censorship of the Min protocol. Meanwhile, the timeout for HTTP GET and other prefix transports in the mixed subnet remained stable. We observed this differentiated blocking by protocol in ASes 197207, 58224, 49100, 16322 and 31549, while we saw no evidence of blocking in the other five ASes listed in Table 1.

We disabled the original Min-only subnet and configured clients to connect to a new *uncontaminated* /24 IPv4 phantom subnet using the Min transport, essentially replacing the original, blocked subnet. However, the connection timeout rate did not improve, indicating that this censorship technique does not block proxies based on their IPs but rather drops traffic that does not match a protocol allowlist. Recall that a phantom IP in the mixed subnet receives connections transported by both the Min and the prefix transports, yet the prefix-transported connections do not experience timeouts at the same rate as in the Min-transported ones.

Upon inspecting traffic captures from our packet capturing tool, we observe this censorship technique drops packets after around 6 exchanged packets, as seen in Figure 8; which is different from the mid-connection dropping technique that drops packets after exchanging around 20 packets, as seen in Figure 13. Also, the mid-connection dropping is applied on the IP-level (see §5.2), while the protocol allowlist performs application-level filtering irrespective of the destination IP. We believe this technique is a variant of the protocol allowlist first discovered by Bock *et al.* [8], although we observe differences compared to their results, including that its deployment is AS-specific and not country-wide.

5 Server-side Censorship Evasion

In this section, we detail two active experiments to evade censorship in Iran. In the first experiment, we inject packets from the server side to trick the censor into confusing which side originated the connection. In the second experiment, we instruct the client to send various protocol prefixes that can convince the censor the connection is an allowed protocol, exempting it from blocking.

5.1 Client Hello Injections

Inspired by the effectiveness of the evasion technique we discovered from our vantage point in Iran during the first half of 2023 (§4.1.1), we designed an experiment to inject packets with certain payload types from the **server side** upon receiving client's intent to connect (clients SYN packets). The goal of this experiment is to learn if our injections can confuse a censor into allowing a connection that would otherwise be blocked. In August 2023, we ran three experiments where we inject one of three application-layer protocol payloads each time using the same experiment setup: TLS (Client Hello), SSH (Version Exchange), and HTTP (GET). These are typically payloads sent by the client, and our goal was to confuse the censor into mixing up client and server.

[4]which ran on the 2nd, 26th and 50th hours relative to the experiment onset

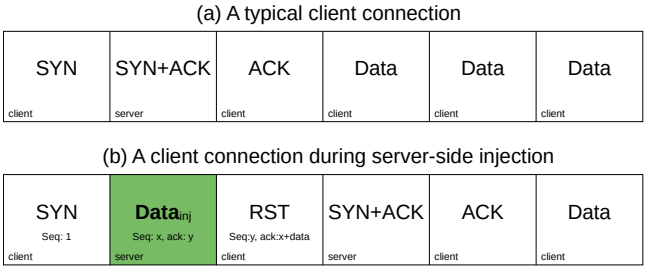


Figure 10: The first 6 Packets in Different Settings — A depiction of packet ordering in a given connection from the perspective of a middlebox closer to the client. The order of packets 3 through 6 in (b) can vary.

For brevity, we only detail the TLS experiment and report on the effectiveness of the other two protocols, which differ only in payload sent.

Our experiments in Section 4.1.1 suggest that at least some censors in Iran only inspect the first 5 packets in a connection, and presumably use them to determine if it contains a payload that should exempt it from blocking. From this intuition, we design an experiment that injects a likely-exempt payload from the server to the client. However, for it to be received within the first 5 packets at the censor's perspective, waiting for the 3-packet TCP handshake to complete may be too late, as the client (who is closer to the censor) may have sent two data packets by the time ours reaches the censor.

Thus, we inject our application-layer packet (with `PSH+ACK` TCP flags and no TCP options) before we send the `SYN+ACK` to accept the TCP connection, and we further delay our `SYN+ACK` packet by 10ms using the `tc` Linux utility to ensure our injected packet reaches the censor first.

We set the sequence and acknowledgment numbers for our injected packet to constant values. While a TCP-compliant implementation would reject this data packet sent before the handshake has completed, prior work has shown that censor middleboxes can be confused with similar techniques [7]. Meanwhile, the client will ignore this data and send a `RST` in response, and wait for the `SYN+ACK`. The `RST` may further confuse a censor, who may believe the connection is being torn down. Figure 10 shows a depiction of the censor's packet buffer for normal connections compared to our server-side injected ones.

Using a tool we wrote in Go, we injected a fixed set of bytes (including a fixed Client Random) representing a TLS Client Hello with an innocuous SNI (users.rust-lang.org) to every `SYN` packet we received, and simultaneously captured traffic. The TLS fingerprint of the TLS Client Hello message we use in our injections does not match any common TLS fingerprint [19]. To minimize any potential disruption, we ran this experiment for 20 minutes. After the experiment concluded, we captured traffic for another 20 minutes to serve as our control group. We present the results in Table 2.

Surprisingly, we find that our server-side injection was effective in evading censorship in one AS (AS58224), where we observe a significant drop in the censor's injection rate and a significant increase in the bytes transferred. Note that we only capture the first 100 packets in a given connection. We also observe no significant difference in the connection establishment rate across all clients, suggesting that this form of injection does not break clients' connections. However, we observe a significant drop in the transfer rate for clients in AS44244. We are unsure what causes it, but conjecture that a threshold number of allowed packets is reached faster (due to two additional packets *i.e.* our injection packet and the clients induced `RST` packet in response to our injection) or that the sequence of packets triggers censorship faster for these clients.

Injecting with an SSH version exchange payload achieved similar results to the TLS Client Hello payload. However, injecting with HTTP GET payload was ineffective across all clients.

These results suggest that it is not the client-induced `RST` that is responsible for censorship evasion, but rather the application message itself (Client Hello or SSH Version Exchange), because other non-effective payloads, *e.g.* a non-SNI Client Hello or HTTP GET, induce the same `RST` and yet fail to evade censorship.

In the beginning of 2025, and inspired by the discovery of the protocol allowlist in some ASes (§4.4) in which the censor looks for protocol signatures in the clients first few packets, we replicated this server-side injection experiment against clients in ASes 197207 and 58224, injecting HTTP GET payloads upon every connection attempt. However, our injections did not help trick the protocol allowlist, indicating the allowlist operates unidirectionally, while the packet injector we observe in §4.1 is bidirectional.

5.2 Detection-resistant Protocols

On July 23, 2024, we deployed an experiment to examine which application protocol prefixes are resistant to censorship in Iran. We use six previously-unused /26 phantom subnets, and configure clients to connect to the first two subnets using the Min transport only, and to the next two subnet using the TLS-prefixed Min transport on port 443, using a TLS fingerprint collected from a RIPE Atlas [34] probe. We used an SNI of `duckduckgo.com`, which we confirmed is not blocked in Iran based on RIPE Atlas probes results and OONI [15] data. Lastly, we configure clients to connect to the last two subnet using the HTTP GET-prefixed Min transport on port 80, in which we use a 16-byte HTTP GET payload with no `Host` header. We equally divide connections to the four prefixed Min subnets between TLS and GET.

At 72 hours after the experiment onset, we observe a country-wide IP-based blocking of both TLS subnets. Figure 9 shows the drop in the number of connections we receive from Iran for the two TLS subnets. Less than 24 hours later, we observe a sharp increase in the rate of connections timeout for the two Min subnets for traffic from AS197207 as shown in Figure 13, consistent with our previous findings (§4.3. We then split one of the Min subnets into two /27 subnets: one still receives Min-only connections and the other receives GET-prefixed Min connections. However, the GET prefix did not help lift the blocking for that /27, indicating that IPs are blocked for longer-term durations than individual connections.

Finally, four weeks after the experiment onset, we do not observe any drop in the number of connections or increase in the connections timeout rate for the two GET subnets for all traffic from Iran. To ensure that it is the protocol prefix (GET) and not the destination port (80) that contributes to this "non-detection" by

Table 2: Server-side Evasion Results — We inject TLS Client Hello bytes from the station side to every `SYN` packet we receive. Results show injection evasion effectiveness in AS58224. We include an AS outside of Iran to showcase that our injections do not break clients' connections.

ASN (CC)	Connections		Rate of Change (%)		
	Experiment	Control	Average Bytes (↑ / ↓)	Connection Establishment Rate	Censor Injection Rate
58224 (IR)	623	618	+37/+69	-3	-87
197207 (IR)	2,738	2,952	-7/-3	-3	0
44244 (IR)	227	210	-27/-73	0	+12
136255 (MM)	1,618	1,546	+1/+7	0	0

censors in Iran, we reconfigure clients to connect to the two GET subnets using destination port 443 instead of 80. One week after this switch, we still do not observe any signs of censorship.

This more recent finding appears to contradict the findings from our server-side evasion techniques (§5.1) in which we find that the injecting HTTP GET bytes to clients does not help evade the censors injections. This may be due to directionality, or that our server-side injection experiment targeted already in-use subnets that may have different reputation to the censor.

More Prefixing with Application Protocols. On August 29, 2024, we expand on our experimentation, and using three previously-unused /26 phantom subnets, we tested three additional application-level protocols: In the first, a 17-byte HTTP POST prefix (on port 80), second, a 21-byte SSH version prefix (on port 22), and third, a DNS query for `www.wechat.com` (on TCP port 53), motivated by the findings of Bock *et al.* [8] that DNS-over-TCP payloads are blocking exempt).

The first subnet (HTTP POST) experienced IP-based blocks by ASes 44244 and 57218 three days after the experiment onset. However, the remaining eight ASes did not block this subnet throughout the six weeks of the experiment.

For the first 72 hours, the second subnet (SSH) remained unblocked across all ASes. However, 72 hours in, we switched the configured destination port for this subnet from 22 to 53, and immediately saw a 50% decline in the number of connections from ASes 44244 and 57218. The other eight ASes showed no signs of blocking even after switching ports.

The third subnet (DNS) similarly experienced approximately a 50% drop in the connections from the same two ASes immediately after we switched the configured destination port for this subnet from 53 to 22. Whereas timeout rate for connections from ASes 197207, 58224, 16322 and 31549 increased significantly (even before we configure clients to switch the destination port), indicating a sign of blocking. For the remaining ASes, we do not observe any signs of blocking throughout the six weeks of the experiment.

These experiments suggest prefixing proxy connections with common application protocol payloads allows for censorship avoidance, complementing prior findings [8] and presenting an opportunity for building complete protocol mimicry to avoid proxy censorship in Iran.

6 Related Work

Our work is the first to measure and document censorship from a production Refraction Networking deployment. Prior work has detailed other Refraction deployments [37], including TapDance [39], but there was no evidence of blocking by censors in this deployment. More recently, researchers documented a deployment of Snowflake [11], a WebRTC-based peer-to-peer proxy used in Tor. While they did not observe ISP-specific blocking, they presented evidence of blocking in Iran based on TLS fingerprint [19].

Bhaskar *et al.* [6] found that load balancing led to some connections in China not being censored, while our work explicitly studies proxy censorship, and found active ways to circumvent censorship in Iran (Section 5).

Prior work has also studied ways to detect and infer censorship in countries including Iran using proxy traffic [29]. However, this work focuses on what content and apps are blocked, while our work attempts to measure the techniques used to block proxies themselves in Iran.

6.1 Iran censorship measurements

Our study is one of the first to use a production censorship circumvention proxy to measure censorship. However, prior to using proxies, researchers have used vantage points in Iran to more directly measure what is blocked. In 2013, Aryan *et al.* used a single vantage point in an unspecified major ISP in Iran to measure what websites in the Alexa top 500 sites were blocked by DNS, HTTP, and DPI-based blocking [5]. Using traceroutes and `TTL`-limited probes, this study concluded that blocking occurred at least partially upstream of the vantage point's ISP, in Telecommunication Company of Iran (TCI). This study also found SSH and obfuscated traffic subject to throttling, while HTTP(S) traffic is spared. Anderson [4] also observes similar throttling. We, however, find inserting SSH fingerprints helps avoid censorship in some ASes, and inserting HTTPS fingerprints does not.

More recently, in 2020 Bock *et al.* [8] observe a country-wide protocol filter in Iran from 6 vantage points around the country, finding that only a small set of protocols including HTTP, TLS, and DNS are allowed on their native TCP ports. We observe a similar censorship (§4.4), but with subtle differences. For instance, contrary to Bock *et al.* , we observe the deactivation of the protocol allowlist as described by the authors between July 2023 and the end of 2024, and observe its re-deployment in January 2025 in some, but not all, ASes.

Others have suggested that Iran's censorship is centralized, based on observations from a collection of vantage points and the similarity of what sites are blocked in each [20]. However, this study is limited to censorship of specific sites, and misses network interference targeting proxies.

Collectively, this body of work underscores the challenges in obtaining vantage points in Iran, particularly in residential or mobile networks, emphasizing the need for alternative approaches like ours.

6.2 Measurements on censorship decentralization

Prior work has also studied decentralized censorship in other countries. For instance, prior studies have focused on the varying ISP censorship in Russia [33, 41], and others have observed inconsistency in how ISPs block domain names in India [42]. More recently, researchers have also observed that China's Great Firewall (GFW) is subtly different depending on the direction probes are sent across the firewall [21], despite the long-standing belief that the GFW is symmetric. These studies highlight the need to study and understand censorship at a deeper level than the entire country-level, which our work also highlights specifically in Iran.

7 Future Work

Our work identifies that censorship in Iran is not as monolithic as previously assumed. While we show some evidence of a country-wide censorship apparatus, there also appears to be individual differences between, and within, ASes, motivating further study in several directions.

First, we suggest researchers further investigate ISP- and subnet-level censorship differences in Iran and other countries previously thought to use centralized blocking systems. Using traditional and novel measurement techniques to focus on proxy censorship can help reveal emerging censorship.

Second, we suggest further investigation into applying evasive strategies in this evolving domain. While we outline first steps in Section 5, there is room for additional techniques to be employed. For example, our experiments with protocol prefixes suggest this and other protocol mimicry remains a promising technique. However, we caution these approaches are fraught with challenges, as identified in prior work [19, 22, 40].

Finally, deployed proxies with server-side control offer unique directions to explore in terms of automated evasion techniques, such as with automated circumvention tools like Geneva [9, 10]. Integrating these techniques into Conjure or other production proxies could yield a rapid and automatic way to respond to the ever evolving censorship landscape.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback and comments. We also thank the team at Psiphon for working with us to deploy our proxy and conduct our measurements, including Joe Arshat, Rod Hynes, Adam Kruger, Keith Mcmanamen, and Irv Simpson. We also thank the Refraction Networking team, including Mingye Chen, J. Alex Halderman, Michalis Kallitsis, Jackson Sippe, and Jack Wampler for their help in operating and instrumenting the proxy.

This work was partially supported by the U.S. National Science Foundation (CNS #2145783), until the award was abruptly terminated as part of the agency's shift in priorities [16], significantly impacting ongoing research efforts.

References

[1] 2022. *Transmission Control Protocol (TCP)*. RFC 9293. RFC Editor. https://www.rfc-editor.org/rfc/rfc9293

[2] Giuseppe Aceto, Antonio Montieri, and Antonio Pescapè. 2016. Internet Censorship in Italy: a first look at 3G/4G networks. In *Cryptology and Network Security: 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings 15*. Springer, 737–742.

[3] Alice, Bob, Carol, Jan Beznazwy, and Amir Houmansadr. 2020. How china detects and blocks shadowsocks. In *Proceedings of the ACM Internet Measurement Conference*. 111–124.

[4] Collin Anderson. 2013. Dimming the Internet: Detecting throttling as a mechanism of censorship in Iran. *arXiv preprint arXiv:1306.4361* (2013).

[5] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. 2013. Internet Censorship in Iran: A First Look. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/foci13/workshop-program/presentation/aryan

[6] Abhishek Bhaskar and Paul Pearce. 2022. Many Roads Lead To Rome: How Packet Headers Influence DNS Censorship Measurement. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 449–464. https://www.usenix.org/conference/usenixsecurity22/presentation/bhaskar

[7] Kevin Bock, Abdulrahman Alaraj, Yair Fax, Kyle Hurley, Eric Wustrow, and Dave Levin. 2021. Weaponizing middleboxes for TCP reflected amplification. In *30th USENIX Security Symposium (USENIX Security 21)*. 3345–3361.

[8] Kevin Bock, Yair Fax, Kyle Reese, Jasraj Singh, and Dave Levin. 2020. Detecting and Evading Censorship-in-Depth: A Case Study of Iran's Protocol Whitelister. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association. https://www.usenix.org/conference/foci20/presentation/bock

[9] Kevin Bock, George Hughey, Louis-Henri Merino, Tania Arya, Daniel Liscinsky, Regina Pogosian, and Dave Levin. 2020. Come as you are: Helping unmodified clients bypass censorship with server-side evasion. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 586–598.

[10] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. 2019. Geneva: Evolving censorship evasion strategies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2199–2214.

[11] Cecylia Bocovich, Arlo Breault, David Fifield, Xiaokang Wang, et al. 2024. Snowflake, a censorship circumvention system using temporary {WebRTC} proxies. In *33rd USENIX Security Symposium (USENIX Security 24)*. 2635–2652.

[12] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *Free and Open Communications on the Internet*. USENIX. https://www.usenix.org/system/files/conference/foci18/foci18-paper-dunna.pdf

[13] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications.

[14] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies* (2015).

[15] Arturo Filasto and Jacob Appelbaum. 2012. OONI: Open Observatory of Network Interference.

[16] National Science Foundation. 2025. Updates on NSF Priorities. https://www.nsf.gov/updates-on-priorities.

[17] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J Alex Halderman, Nikita Borisov, and Eric Wustrow. 2019. Conjure: Summoning proxies from unused address space. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2215–2229.

[18] Sergey Frolov, Jack Wampler, and Eric Wustrow. 2020. Detecting Probe-resistant Proxies.. In *NDSS*.

[19] Sergey Frolov and Eric Wustrow. 2019. The use of TLS in Censorship Circumvention.. In *NDSS*.

[20] 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 Transactions on the Web (TWEB)* 9, 1 (2015), 1–29.

[21] Nguyen Phong Hoang, Jakub Dalek, Masashi Crete-Nishihata, Nicolas Christin, Vinod Yegneswaran, Michalis Polychronakis, and Nick Feamster. 2024. GFWeb: Measuring the Great Firewall's Web Censorship at Scale. In *USENIX Security Symposium*. USENIX. https://www.usenix.org/system/files/sec24fall-prepub-310-hoang.pdf

[22] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. 2013. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 65–79.

[23] Freedom House. 2022. Freedom on the Net 2022: Countering an Authoritarian Overhaul of the Internet. https://freedomhouse.org/sites/default/files/2022-10/FOTN2022Digital.pdf.

[24] Freedom House. 2023. Freedom on the Net 2023: The Repressive Power of Artificial Intelligence. https://freedomhouse.org/sites/default/files/2023-10/Freedom-on-the-net-2023-DigitalBooklet.pdf.
[25] The Internet Monitoring Action Project (iMAP). 2023. iMAP Myanmar 2023: Internet Censorship Report. https://imap.sinarproject.org/reports/2023/imap-myanmar-2023-internet-censorship-report/imap-myanmar-2023-internet-censorship-report.pdf.
[26] Bruce Leidl. 2009. Obfuscated-openssh. https://github.com/brl/obfuscated-openssh.
[27] Alexander Master and Christina Garman. 2023. A Worldwide View of Nation-state Internet Censorship. In *Free and Open Communications on the Internet*.
[28] MaxMind. 2024. GeoLite2. https://dev.maxmind.com/geoip/geoip2/geolite2.
[29] Keith McManamen, Simin Kargar, and Jacob Klein. [n. d.]. Early Detection of Censorship Events with Psiphon Network Data. ([n. d.]).
[30] Netfilter. 2024. nftables. https://netfilter.org/projects/nftables.
[31] OONI. [n. d.]. OONI Measurement Aggregation Toolkit. https://explorer.ooni.org/chart/mat?probe_cc=MM&since=2023-07-12&until=2024-04-17&time_grain=day&axis_x=measurement_start_day&test_name=psiphon. Accessed: 2025-05-02.
[32] Ram Sundara Raman, Mona Wang, Jakub Dalek, Jonathan Mayer, and Roya Ensafi. 2022. Network Measurement Methods for Locating and Examining Censorship Devices. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies* (Roma, Italy) *(CoNEXT '22)*. Association for Computing Machinery, New York, NY, USA, 18–34. https://doi.org/10.1145/3555050.3569133
[33] Reethika Ramesh, Ram Sundara Raman, Matthew Bernhard, Victor Ongkowijaya, Leonid Evdokimov, Anne Edmundson, Steven Sprecher, Muhammad Ikram, and Roya Ensafi. 2020. Decentralized control: A case study of russia. In *Network and Distributed Systems Security (NDSS) Symposium 2020*.
[34] RIPE NCC Staff. 2015. RIPE Atlas: A global internet measurement network. *Internet Protocol Journal* 18, 3 (2015).
[35] Flavia Salutari, Danilo Cicalese, and Dario J Rossi. 2018. A closer look at ip-id behavior in the wild. In *Passive and Active Measurement: 19th International Conference, PAM 2018, Berlin, Germany, March 26–27, 2018, Proceedings 19*. Springer, 243–254.
[36] Ram Sundara Raman, Louis-Henri Merino, Kevin Bock, Marwan Fayed, Dave Levin, Nick Sullivan, and Luke Valenta. 2023. Global, Passive Detection of Connection Tampering. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 622–636.
[37] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J Alex Halderman, Nikita Borisov, and Eric Wustrow. 2020. Running refraction networking for real. *Proceedings on Privacy Enhancing Technologies* (2020).
[38] Tim Wilde. Jan. 7, 2012. Knock knock knockin' on bridges' doors. Tor Blog. https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors.
[39] Eric Wustrow, Colleen M Swanson, and J Alex Halderman. 2014. TapDance:End-to-Middle Anticensorship without Flow Blocking. In *23rd USENIX Security Symposium (USENIX Security 14)*. 159–174.
[40] Diwen Xue, Michalis Kallitsis, Amir Houmansadr, and Roya Ensafi. 2024. Fingerprinting Obfuscated Proxy Traffic with Encapsulated TLS Handshakes. In *USENIX Security Symposium*. USENIX. https://www.usenix.org/system/files/sec24summer-prepub-465-xue.pdf
[41] Diwen Xue, Benjamin Mixon-Baca, ValdikSS, Anna Ablove, Beau Kujath, Jedidiah R Crandall, and Roya Ensafi. 2022. TSPU: Russia's decentralized censorship system. In *Proceedings of the 22nd ACM Internet Measurement Conference*. 179–194.
[42] Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. 2018. Where the light gets in: Analyzing web censorship mechanisms in india. In *Proceedings of the Internet Measurement Conference 2018*. 252–264.

A Ethics

User data is particularly sensitive in censorship circumvention contexts, and we prioritize safety and privacy of the clients involved in this study. In particular, our logs and datasets do not collect or contain any personally identifiable information (PII) or other sensitive information. The architecture of our system prevents us from seeing the ultimate destination website a user intends to go to, as we only connect the user to a next-hop proxy operated by Psiphon, where the client's traffic is end-to-end encrypted to. In addition, we do not record client IP addresses, only the geolocation country and AS number. This complies with the privacy policy of Psiphon, agreed to by users of the system.

Table 3: Subnets Overview — Packet injection rates for the top 4 subnets in two of the largest ASes in Iran during the 9-month measurement show large differences between some subnets, suggesting censorship in these ASes affects their respective clients differently.

ASN	Subnet	% of AS Connections	Avg Injection Rate (%)
58224	5.232.0.0/14	7.90	25.54
	151.233.0.0/16	5.04	41.48
	2.183.0.0/16	4.36	21.77
	5.200.128.0/17	3.33	20.64
44244	5.120.0.0/13	58.35	56.01
	5.119.0.0/16	10.99	66.69
	2.146.0.0/15	9.60	62.77
	5.112.0.0/14	5.40	0.04

B Understanding the Censor Affecting our VPS Vantage Point

Hop-based localization of this censor. Because the censor looks for a valid TLS handshake record in the first 5 packets in a given TCP connection that involves a blocklisted IP, we leverage this fact for estimating the censoring middlebox hop distance in the network. We run 10 experiments to send a sequence of 10 TCP packets to a blocklisted IP as follows: the first packet has the SYN flag set, the third packet has the PSH flag set and contains a TLS handshake record in its payload. We also set the TTL of the third packet to a value from the range [1,10], one value for each experiment. The remaining packets have the PSH flag set and contain random data in their payloads. For a given experiment, we expect injected responses from the censor if it did not see the TTL-limited packet (the third packet *i.e.* the TLS handshake record). This packet is supposed to permit the connection from being censored. For the experiments where the third packet's TTL value is within the range [1,4], we observe the censor's injected responses; however, for TTL values within the range [5,10] we observe no injected responses. This suggests that the censor is located between hops 4 and 5, inclusive. Although this method is known to be effective in locating censoring middleboxes [32], a censor could conceal its actual hop location by further decrementing the TTL of the packets it inspects before processing them; however, we believe this to be unlikely. We find that this censoring middlebox is located at the same hop range as the one that censors sensitive DNS and HTTP(S) requests involving non-blocklisted IP addresses. However, we are not certain if these middleboxes are the same functioning unit or different ones.

How allowlisted protocols are detected. To further understand how the censor detects these protocols if a blocklisted IP is involved in the connection, we manipulate fields of the protocol prefix (the first message of the protocol, typically TLS Client Hello or SSH protocol version exchange) and examine whether these manipulations still permits connections to blocklisted addresses or not. These manipulations involve addition, deletion, or substitution of the protocol fields. We find that the censor detects TLS connections involving blocklisted addresses and permits them if any of the four packets

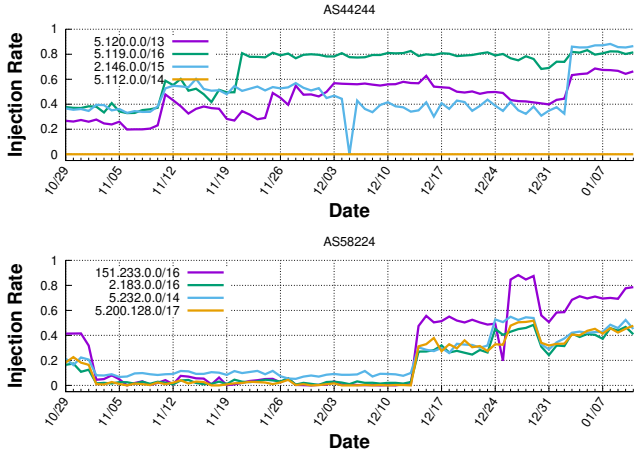


Figure 11: Snapshot of packet injection rate over time for the top 4 subnets in two ASes in Iran.

proceeding a SYN contain a *valid* TLS handshake record. What constitutes a *valid* TLS handshake record from the censor's perspective is only the first three TLS fields. First, the TLS message type field must be set to a handshake record (`0x16`). Second, the TLS protocol version must be set to a valid TLS protocol version (`0x0301`, `0x0302`, `0x0303`, `0x0304`) or to `0x0300`, which is an invalid value for a TLS protocol version. Third, the value of length field must be valid: it must match the actual count of bytes that follows that field. Finally, the number of bytes following the length field must be greater than 35. On the other hand, we find that the censor detects SSH connections by looking for a valid protocol version exchange regardless of what follows beyond that, and what constitutes a valid one from the censor's perspective is the bytes corresponding to the ASCII string "SSH-" in the beginning of the SSH version exchange.

C Intra-AS Variation in Injection Rates

Are injection rates consistent? We observe that not all Conjure connections receive injections. To understand how injections vary, we focus on two major ASes, AS44244 and AS58224 where we observe injections, and compare the injection rate for the top 4 client subnets in these ASes.

We observe varying injection rates inter- and intra-AS. We pick a snapshot in time where we show the injection rate per subnet for the two ASes. Figure 11 shows the injection rates for the labeled subnets during the period of October 29, 2023 to January 11, 2024.

In AS58224, a drop in injection rate can be observed on November 2, on which we activated an evasion strategy (a Client Hello injection tool we describe in Section 5.1) for clients in this AS, and a spike in the injection rate on December 14 when we turned this technique off.

On the other hand in AS44244, we see subnets experiencing varying rates of injections, however, one subnet is not experiencing any injections.

Table 3 shows the number of connections along with injection rates for the traffic capture measurement period of 9 months. We note that the number of connections per subnet remained respectively high during the snapshot we show in Figure 11.

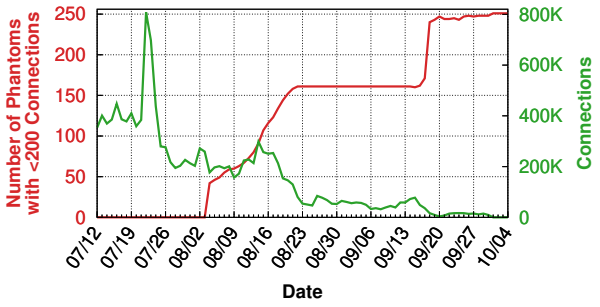


Figure 12: Phased Blocking — AS197207 employed a phased IP-based blocking against addresses in our most-connected-to /24 phantom subnet in 2023.

D Did AS197207 Really Block Phantom IPs in Phases Based on Phantom Connectedness?

In this section, we further investigate the phased IP-based blocking carried out by AS197207 against the high+medium bias phantom IPs in our heavily-used /24 IPv4 phantom subnet, as shown in Figure 5. We perform a statistical test, namely the Wilcoxon Signed-Rank Test, to understand whether there is a statistically significant difference between the blocking that affects the high+medium bias phantom IPs and the low bias ones. We pick a specific point in time where we observe the beginning of blocking by this AS and compare the number of connections 15 days prior to and 15 days after this blocking event. Figure 12 shows an increase in the number of phantom IPs that receive less than 200 connections[5] a day from this AS on August 4. Thus, we label the 15-day period prior to this date as *before censorship* and the 15-day after as *after censorship*. We calculate the number of connections each phantom in the high+medium and the low bias ranges receives (23 phantoms for each of the two categories).

Our goal is to compare the number of connections that phantoms in each category receive before and after the event of censorship, and find out if the difference between the two periods is statistically significant under the significance level $\alpha = 0.01$. For our hypothesis to hold (blocking begins with the high+medium bias phantoms first), it is necessary that the difference in the number of connections in the high+medium category before and after the censorship event is statistically significant and the difference in the low bias category is statistically insignificant.

To formulate this problem, let:

- X_i^{before} be the number of connections to IP i before the suspected censorship event.
- X_i^{after} be the number of connections to IP i after the suspected censorship event.
- N be the number of tracked IPs in the high+medium and low bias ranges *i.e.* 23 IPs for each category.

We want to test the hypothesis for each category:

$$H_0 : P(X_i^{\text{after}}) = P(X_i^{\text{before}})$$

$$H_A : P(X_i^{\text{after}}) < P(X_i^{\text{before}})$$

[5] a threshold we set heuristically

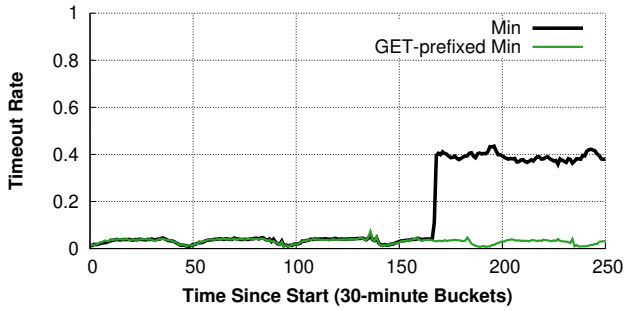


Figure 13: We configure clients in AS197207 to connect to two phantom subnet, one using the Min transport and the other using GET-prefixed Min. Connections to the Min-only subnet experience censorship by mid-connection dropping several days after the experiment onset as indicated by the spike in connection timeout rate.

Meaning, the probability distribution of connections per IP exhibits one-sided change *i.e.* a statistically significant drop in the observed number of connections after the censorship event.

We apply the Wilcoxon Signed-Rank Test, a non-parametric test that compares paired samples $(X_i^{\text{before}}, X_i^{\text{after}})$ for $i \in \{1, \ldots, N\}$.

The test statistic is computed as:

$$W = \sum_{i=1}^{N} R_i \cdot \text{sign}(X_i^{\text{before}} - X_i^{\text{after}})$$

where:

- R_i is the rank of $|X_i^{\text{before}} - X_i^{\text{after}}|$.
- $\text{sign}(x)$ is the sign function:

$$\text{sign}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}$$

We compute the W statistic for each category and calculate the z score based on it. We find that for the significance level we set ($\alpha = 0.01$) there is a statistically significant difference between the number of connections in the high+medium bias category before and after the censorship event, rejecting the null hypothesis for this category (p-value ≈ 0). Whereas there is no statistically significant difference between the number of connections in the low bias category before and after the censorship event, failing to reject the null hypothesis for this category (p-value of 0.02).

E Conjure Station

Conjure station is the core component in the Conjure system that allows clients to proxy their connections to their desired (covert) destinations. Although stations consist of a few modules, the two main ones are the **detector** and the **application**.

The detector performs flow tracking for clients who have active registrations in the Conjure system. It fetches packets from a ring buffer offered by PF_RING and forwards them to the application, if and only if they were sent by registered clients, of which the detector maintains a map.

The application, on the other hand, handles clients connections and proxies them if and only if they demonstrate knowledge of a

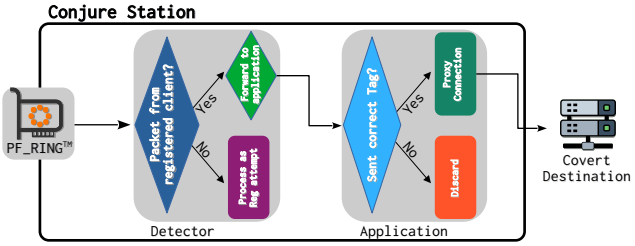


Figure 14: A flowchart showing proxy connection processing in a Conjure station.

shared secret agreed upon during the registration process. By design, the application will not receive packets sent by non-registered clients, since the detector will not forward them to the application in the first place. Figure 14 shows a flowchart highlighting client connections handled by a conjure station.