QUICstep: Evaluating connection migration based QUIC censorship circumvention

Seungju Lee
seungjulee@princeton.edu
Princeton University

Mona Wang
monaw@princeton.edu
Princeton University

Watson Jia
watsonj@alumni.princeton.edu
Princeton University

Qiang Wu
gfw.report@protonmail.com
GFW Report

Henry Birge-Lee
birgelee@princeton.edu
Princeton University

Liang Wang
lw19@princeton.edu
Princeton University

Prateek Mittal
pmittal@princeton.edu
Princeton University

Abstract

Internet censors often rely on information in the first few packets of a connection to censor unwanted traffic. With the rise of the QUIC transport protocol, prior work has suggested the method of using QUIC *connection migration* to conceal the first few handshake packets using a different network path (e.g., an encrypted proxy channel). However, the use of connection migration for censorship circumvention has not been explored or validated in terms of feasibility or performance. We bridge this gap by providing a rigorous quantitative evaluation of this approach that we name QUICstep. We develop a lightweight, application-agnostic prototype of QUICstep and demonstrate that QUICstep is able to circumvent a real-world QUIC SNI censor. We find that not only does QUICstep outperform a fully encrypted channel in diverse settings, but also that it can significantly reduce traffic load for encrypted channel providers. We also propose using QUICstep as a tool for measuring QUIC connection migration support in the wild and show that support for connection migration is on the rise. While as of now QUIC and connection migration support is limited, we envision that QUICstep can be a useful tool for the future where QUIC is the de facto norm for the Internet.

Keywords

Censorship circumvention, QUIC, Connection migration

1 Introduction

As the Internet has become an indispensible tool in our lives, governments concurrently seek to expand censorship programs in a race to control our access to information. Access Now's 2024 data shows that network disruptions are not only occurring more frequently, but also across more countries [47]. The proliferation of commodity network devices that can perform deep packet inspection (DPI) has made scalable network censorship available to a wider range of

governments and ISPs [55]. Well-established censorship regimes, such as China's Great Firewall (GFW) and Russia's TSPU, continue to deepen their methods for blocking network connections and identifying circumvention technologies [28, 29, 74].

From previous studies of censorship systems in the wild, the vast majority of connections, including fully encrypted connections, are filtered based on information in the handshake packets. For example, SNI-based detection is so critical to censorship [15] that censors began prematurely blocking connections using the encrypted SNI extension as early as 2020 [10]. Similarly, the GFW only prioritizes the first few packets of a connection when deciding whether to exempt it from blocking [70]. Censorship measurement platforms such as OONI and Censored Planet leverage this to measure censorship at a global scale by primarily sending handshake probes [1, 54]. Ultimately, the handshake is critical in providing censors with sufficient connection metadata to make a judgment on whether to block the rest of the connection.

Hiding the handshake from the censor or bootstrapping the connection via another channel is key to circumventing handshake-dependent censorship in practice. In this line, Wang et al. have suggested that connection migration can be used to circumvent stateless censorship of QUIC by splitting handshake and non-handshake packets across different network paths [64]. Connection migration is a notable feature of QUIC that allows connections to be maintained while the IP address or port of endpoints change. By transmitting only handshake packets through a censorship-resilient channel (e.g. VPN), users can minimize the performance overhead and load incurred by the censorship-resilient channel while enjoying the benefits of circumventing censorship.

However, prior work only briefly suggests this censorship circumvention scheme and leaves unanswered critical questions about its design, implementation, practicality, and quantitative performance benefits.

Contributions. We name this approach QUICstep and seek to understand its feasibility and performance: **First, are web servers compatible with QUICstep? Second, can QUICstep effectively circumvent deployed censors? Third, what is the quantitative performance impact of using QUICstep?** To address these questions, we first define a clear threat model (§3.1) that focuses on the

behavior of deployed censors that use simple and lightweight mechanisms (e.g., stateless behavior) over comprehensive but complex mechanisms. We then develop a lightweight, open-source prototype of QUICstep (§3.2) that enables efficient path migration in a real-world web browsing setting. Our implementation is available at https://github.com/inspire-group/QUICstep. Using this implementation, we perform comprehensive evaluations to demonstrate the effectiveness and performance of QUICstep.

We successfully use QUICstep to circumvent active QUIC-SNI censorship in the wild, showcasing its effectiveness against real-world deployed censors who leverage stateless detection techniques (§4.3). While more sophisticated censors could employ stateful traffic analysis to detect QUICstep, this may increase censor's cost; we discuss this possibility and tradeoff in a more thorough security analysis of QUICstep (§5).

We demonstrate the performance benefits of QUICstep across a wide range of experiments. We use the term *handshake channel* to describe the secure tunnel through which a QUICstep client performs the handshake (§3.1.1). Compared to completely relying on the handshake channel for all traffic, QUICstep can reduce page load time by up to 84%, and load on handshake channel providers by a median of **93%**. The performance gain of QUICstep becomes more pronounced when the handshake channel is operating under practical limitations, such as bandwidth limits. QUICstep could be used to reduce load on handshake channel providers, such as VPNs or other censorship circumvention tools.
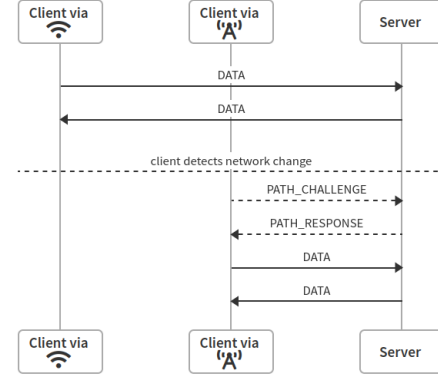
For QUICstep to be practically useful, the website must support both QUIC and connection migration, which currently presents a bottleneck for widespread use of such techniques. Leveraging QUICstep, we conduct a large-scale measurement of connection migration support in the wild over a three-month period, identifying a small but nontrivial number of QUIC websites that are compatible with QUICstep. While support for both QUIC and connection migration is currently limited, the support is also on the rise: the number of websites that partially support connection migration increased by 20% during our measurement period of 3 months. This shows promise for QUICstep becoming a useful tool in the future.

2   Background and related work

QUIC is a transport layer protocol based on UDP and supports multiplexing of application-layer data streams [32]. QUIC was developed to improve the performance of web applications compared to TCP+TLS and is the basis for HTTP/3. The meteoric rise of QUIC is likely to continue as HTTP/3 has been standardized as an RFC [8]. All major browsers and around 22% of top 1 million websites already support QUIC based on our recent measurement (§4.2).

QUIC provides a variety of network performance features. For instance, by rolling together the QUIC and TLS handshake it eliminates a handshake round-trip [57]. A critical performance feature for our work is *connection migration*, which enables QUIC connections to persist across multiple network-layer sessions.

Connection migration. QUIC utilizes a set of connection identifiers rather than the IP address and port tuple in order to uniquely identify connections. The decoupling of QUIC connections from IP addresses and ports allows connections to be maintained even as clients move between different networks. This capability is referred

Figure 1: An illustration of QUIC connection migration. Before the server can receive data from the client on the new network path, it must be validated. The server can cache recent path validations, preventing the need to perform them every time a network migration occurs.
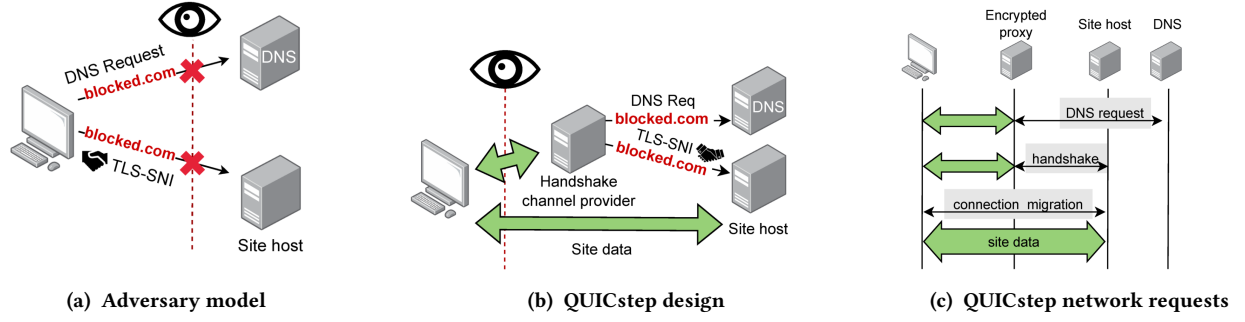
to as *connection migration* [8, §9]. When an endpoint detects a network change, it performs a round-trip *path validation* to ensure that the peer is still reachable before resuming the connection, as demonstrated in Figure 1 [8, §8.2]. Connection migration can only occur after a session has been fully established between a client and server through a QUIC-TLS handshake. QUIC connection migration enables massive performance improvements for mobile users, as connections persist even when devices move across networks such as between a mobile network and a local WiFi network. QUICstep leverages this performance feature to circumvent censorship with minimal latency overhead.

QUIC-TLS handshake. QUIC is designed as a secure-by-default protocol that mandates encryption of data in transit. To achieve this, QUIC is integrated with TLS encryption. Notably, QUIC-TLS encrypts the `Initial` packets with a secret derived from a public salt [57, §5.2]. Although QUIC `Initial` packet encryption does not ensure confidentiality—since the keys can be derived by anyone observing the connection—it still complicates SNI-based censorship by DPI middleboxes as it requires additional computational resources for decryption and more effort to track UDP flows.

2.1   Censorship of QUIC traffic

Network-level adversaries can analyze QUIC-TLS handshake packets and censor connections based on the TLS SNI field, making HTTP/3 connections vulnerable to censorship. Specifically, a censor can first compute the secret used to encrypt the QUIC client's `Initial` packets with the client's destination ID and the public salt [57, §5.2]. It then decrypts the client's `Initial` packet and extracts the SNI field in its TLS `ClientHello` message. If the SNI field matches the censor's blocklist, the censor can then block the ongoing QUIC connection.

The development and adoption of the QUIC protocol created a temporary gap in QUIC censorship as censors needed time to develop DPI software and devices capable of inspecting and blocking

(a) Adversary model

(b) QUICstep design

(c) QUICstep network requests

Figure 2: This figure demonstrates our adversary model and how QUICstep can be leveraged to circumvent censorship. (a) demonstrates an adversary capable of monitoring, blocking or disrupting client traffic based on plaintext sensitive fields that may be associated with HTTPS requests. (b) illustrates the architecture of QUICstep under this adversary model. Finally, (c) demonstrates, at a high level, the full set of network requests performed by QUICstep.

QUIC traffic. For instance in 2023, following the Turkish government's decision to block a social media website due to criticism over its handling of the Türkiye-Syria earthquake fallout, the website's developers reported that users could circumvent TLS-SNI-based censorship by forcing a QUIC connection to their server [35].

However with the rise of QUIC traffic, censorship regimes continue to devise methods to detect and block QUIC traffic. Researchers reported multiple instances of QUIC blocking around the world, including China [76], Uganda [21], Iran [22], and Russia [21]. Early attempts of QUIC censorship included blocking QUIC traffic in general. For example, in 2022 Xue et al. reported that the Russian TSPU censored QUIC traffic by identifying packets that have certain fingerprints, then dropping all packets in that flow [74]. This strategy aims to censor all QUIC traffic and cannot selectively censor traffic to certain websites. As QUIC traffic is expected to increase, broad QUIC blocking may yield increasing amounts of collateral damage.

As detailed in [76], since April 2024, the GFW started censoring QUIC traffic by first decrypting the QUIC client's `Initial` packets then inspecting if the SNI field in the TLS `ClientHello` message matches the blocklist.

2.2 QUIC connection migration for privacy

2.2.1 CoMPS. CoMPS is a connection-migration traffic splitting framework proposed by Wang et al. aiming to improve robustness against website fingerprinting [64]. CoMPS uses a path scheduler that splits traffic across multiple network paths to limit the amount of traffic an adversary can observe, under the assumption that the adversary can only observe packets on a single path. The client uses connection migration to route traffic across the different paths. This work was also the first to present a high-level sketch of QUICstep, suggesting that by sending handshake packets through a VPN path, CoMPS can be used to circumvent SNI-based censorship. However, censorship circumvention is only mentioned briefly as a potential use case of CoMPS, and is not implemented or evaluated for deployment, feasibility, or performance. Thus, we explore the following open questions: Would connection-migration based circumvention be effective in practice? What would the performance benefits be compared to tunneling all traffic using the handshake channel?

2.2.2 MIMIQ. MIMIQ uses connection migration in QUIC to frequently change client IP address within a trusted network to thwart user tracking and certain types of traffic analysis attacks [27]. MIMIQ requires cooperation from the client network, as it needs the network to set up modified DHCP and an edge switch. Thus it is not suitable for our censorship circumvention scenario where the client network itself may be adversarial.

2.3 Related work

2.3.1 TLS session resumption. TLS session resumption has been proposed as a way to circumvent SNI censorship. Introduced in TLS 1.2, session resumption reduces the need for clients to conduct handshakes for each TLS connection. When a TLS session is first established, the server sends a unique ticket to the client which can be used by the client to resume a TLS session with the server. MultiFlow and REDACT propose using session resumption to enable decoy routing [18, 39]. More recently, BlindTLS proposes establishing a connection to a censored domain through a VPN proxy and resuming the session in plain sight of the censor with a different SNI using TLS session resumption [50]. However, BlindTLS focuses on session resumption in TLS 1.2, and would not work with TLS 1.3 as TLS 1.3 requires that the SNI sent in the resumption handshake matches the SNI associated with the session [44]. In contrast with BlindTLS, QUICstep is compatible with TLS 1.3 and is designed to be independent of TLS version.

Our work is the first to thoroughly investigate leveraging various properties of QUIC as a transport protocol to circumvent handshake censorship, rather than being dependent on application-layer features of TLS. Prior SNI censorship circumvention literature has focused on the traditional TCP setting as most censorship in the wild has been observed in this setting [15, 28].

2.3.2 Encrypted ClientHello. Encrypted ClientHello (ECH) is an extension to TLS 1.3 that encrypts the ClientHello message, including the server name, to protect user privacy [45]. ECH could be used to circumvent censorship based on SNI blocklists since the SNI is hidden from observers. However prior research has found that ECH support in servers is limited and that censors in Russia, China, and Iran currently censor ECH traffic, undermining ECH's practicality in censorship circumvention [42].

2.3.3   Measurement of QUIC connection migration support. Directly relevant to our work is the availability and popularity of connection migration. In 2024, Buchet and Pelsser investigated QUIC connection migration support on the web [13]. This work tests connection migration by sending a packet with a new connection ID to the server, but does not initiate or test if they are able to successfully load resources after migrating connections. We found that such a method was incomplete in measuring the practical success rate of migrating connections after switching to a different port or IP address. For instance, QUIC servers generally seem to support port-based connection migration (e.g. migrating a QUIC connection onto a new UDP port) differently from IP-based connection migration (migrating a QUIC connection onto a new IP address). Our work leverages QUICstep to provide the most complete Internet-wide connection migration support measurement to date. We additionally use data from these results to provide recommendations to standards bodies for ways to improve the usefulness of connection migration for all QUIC clients.

2.3.4   Other censorship circumvention systems. There is a long line of research on censorship circumvention that assume more powerful adversaries compared to QUICstep's threat model. These works use complex traffic shaping, traffic mimicry, and other obfuscation techniques for tunneling encrypted traffic, often against adversaries that can use higher-cost techniques (e.g. powerful machine learning classifiers) to identify blocked content from the metadata of encrypted traffic flows [24, 30, 33, 40, 46]. While we briefly discuss more powerful adversaries in §5, we note that the focus of this work is on realistic adversaries that have more limited resources. QUIC-step's focus on a "lightweight" censor is similar to Geneva, domain fronting, domain shadowing, and Snowflake, many of which are deployed in practice and are used by millions of users to circumvent network censorship [9, 11, 23, 67].

2.3.5   Real-world censors prefer simple and efficient detection methods. As the GFW is often a first-mover in the global censorship ecosystem, we soon expect other censorship regimes to follow suit in enacting QUIC SNI censorship [76]. The goal of this work is to stay ahead of the censors: what might widespread, global QUIC SNI censorship look like in practice? Empirical measurements of real-world censorship machines reveal that censors prefer relatively simple and efficient detection mechanisms over comprehensive but complex or expensive detection mechanisms [3]. For example, Wu et al. discovered that the GFW only inspects the first TCP payload sent by the clients when detecting if a TCP connection is fully encrypted [70]. This is consistent with the idea that the handshake packet, even when encrypted, is the most information-rich portion of the connection that are used by censors to make efficient blocking decisions. Similarly, Zohaib et al. discovered that the GFW assumes the first UDP payload is a complete QUIC client `Initial` packet when conducting QUIC SNI-based censorship [76]. Based on these observations we focus on a censor that performs stateless, lightweight censorship.

We acknowledge that in the future more invasive and powerful censors that perform stateful censorship may emerge. But given that GFW is the primary QUIC-SNI censorship system that is deployed in the real world, we focus on GFW-style censors and consider

stateful censors beyond the scope of this work. We discuss our threat model further in section §3.1.

3   Bringing QUICstep from theory to practice

QUICstep's primary goal is to circumvent QUIC SNI censorship while minimizing overall latency overhead and avoiding modifications to server-side software or the QUIC protocol. In this section, we discuss how we bring QUICstep from theory to practice, by first demonstrating our threat model and implementation goals, and then delving into the challenges and implementation details.

3.1   Threat model

Our threat model follows prior work on research on censorship circumvention techniques such as domain fronting [23] and Geneva [9]. In our threat model, the client is located in a censored network and aims to access a censored domain hosted on a *non-blocked* IP address outside the censored network. The censor uses DPI techniques such as DNS and SNI filtering to identify and prevent such access. We consider a practical censor as discussed in §2.3.5, who employs lightweight methods to achieve real-time detection at scale. Specifically, the censor leverages *stateless* detection techniques that can be performed on a per-packet basis, and does not record all network flows to perform flow-level traffic analysis. Anonymity is not a primary concern for the client, which aligns with the assumptions in previous works (e.g., MassBrowser [41]). In practice, public VPNs or proxies are commonly used by users in censored countries to bypass censorship even though these tools do not guarantee anonymity. User surveys also suggest that users in censored regimes often prioritize content access and internet speed over security or anonymity [14, 72].

3.1.1   Client model and the handshake channel. The client wants to access censored domains with low performance overhead. We assume that the client can already access a secure, blocking resistant but potentially high-latency handshake channel (e.g. a DNS tunnel or public VPN) for every connection. A large amount of censorship resilience literature is focused on the development of such high-security, low-bandwidth channels, such as Tor bridges/pluggable transports, the *rendezvous channel* used in Tor's Snowflake [11], *signaling channels* for Tor bridge distribution [60]. The client can employ any of these synchronous, high-security channels as a handshake channel. Censorship circumvention tools like Hysteria [31], V2Ray [59], Xray [71], Sing-box [52], and Tor pluggable transports such as Meek (domain fronting) [43] are also potential options for handshake channels.

However it is not desirable for the client to rely on this channel for all traffic. The handshake channel is resource-constrained (shared with other users), and thus may charge or impose bandwidth or data constraints on individual users. For example, the free version of Lantern has a monthly data limit of 500MB [36].

We also assume that the censor is unable to break the security guarantees of QUIC or the handshake channel and will not attack the availability of certain classes of web traffic (e.g. blocking all QUIC traffic) to avoid collateral damage. Further discussion on blocking all or certain types of QUIC traffic continues on §5. Figure 2(a) illustrates a censored network representing our threat model.

3.2 Implementation

Figure 2(b) depicts the high-level architecture of QUICstep. The client first completes a QUIC-TLS handshake and establishes a QUIC session with the server through a secure handshake channel. The client then immediately switches to the native network path, allowing the rest of the packets to be sent directly to the server with minimal latency.

QUICstep requires setting up a handshake channel. The handshake channel can be any secure channel with blocking resistance (e.g., VPN). For our prototype, we used a WireGuard channel. We did not use Tor even though it provides much higher security and anonymity guarantees, as Tor currently does not support UDP tunneling. Furthermore, using WireGuard proxies under our control enables a more controlled experiment where we can vary proxy location or throughput. Many users in censored domains actively use such custom proxies for censorship circumvention [31, 48, 51, 52, 59, 71]. We note that our implementation of QUICstep is agnostic to the particular type of handshake channel so long as the channel provides a virtual network interface.

3.2.1 Implementation goal.
Our primary goal in implementing QUICstep is to develop an easy-to-implement, application-agnostic solution. We want QUICstep to be compatible with the vast technological environments that clients and servers may be running in, e.g., requiring no modifications to client applications, upper-layer protocols, operating systems, and browsers. Our implementation of QUICstep does not necessitate any changes apart from requiring that the client and server support QUIC and connection migration.

3.2.2 Implementation choices and challenges.
Once handshake happens through the handshake channel, the challenge is to identify confirmation of the handshake and route packets accordingly (regardless of the application through which the packets are transmitted). According to the QUIC standard, connection migration is expected to happen after the handshake is *confirmed* at the peer [32]. When the server confirms the handshake, it sends a `HANDSHAKE_DONE` frame to the client in a 1-RTT packet, but this frame is encrypted and hard to identify from outside of the application. If handshake confirmation is not accurately identified it can incur latency overhead as the server sends additional packets through the handshake channel to *complete* the handshake.

To accurately identify handshake confirmation we considered two options: (1) using eBPF [20] to heuristically determine which connections had finished handshake confirmation based on unencrypted parts of payloads and (2) modifying QUIC clients to track the frame. However, the first solution requires managing the state of multiple connections and requires privileged deployment in the client, and the second would be challenging to deploy in practice as it requires changes to the QUIC client. Neither of these satisfied our requirements for flexibility and ease of future deployment.

As such, we chose instead to *approximate* handshake confirmation time by simply routing handshake packets differently from data packets. The key insight is that QUIC handshake packets use the QUIC "long header" format, but data packets use the QUIC "short header" format. The two formats are differentiated by the unencrypted first bit of the QUIC header. Thus it is possible to differentiate handshake packets from data packets using only information exposed on the wire.

3.2.3 Proof-of-concept implementation description.
Our proof-of-concept version of QUICstep uses `iptables` firewall rules to route select packets through the WireGuard interface. Our rules route DNS packets (UDP packets headed to port 53), TCP packets, and QUIC long header packets (UDP packets headed to port 443 whose payloads begin with 1) through the WireGuard interface. This ensures that all packets containing server name are transmitted securely. In addition, QUICstep does not have any requirements for the client besides having access to a WireGuard proxy (or other handshake channel), and incurs no significant latency overhead on the client. If the server does not support QUIC or connection migration, the client would fall back to TCP which is fully transmitted through the handshake channel. The user experiences no significant failure.

The code for our implementation and evaluations are made available at https://github.com/inspire-group/QUICstep.

4 Evaluation

In this section, we evaluate QUICstep's ability to circumvent real-world censorship as well as its performance.

4.1 Research questions and overview

We give an overview of the evaluations designed to answer each of our research questions (as outlined in §1).

4.1.1 QUICstep-compatibility: measuring compatible websites in the wild.
We identify the current state of QUIC and connection migration support among popular websites by performing HTTP/3 GET requests with QUICstep (§4.2). We find that 22% (∼220 K) of top 1 M domains support QUIC and 12.8% (∼28 K) of QUIC supporting domains are compatible with QUICstep. Our findings additionally suggest that the role of service providers is critical in support for connection migration.

4.1.2 Effectiveness: practical censorship circumvention with QUIC-step.
We evaluate QUICstep's ability to circumvent real-world censors that perform SNI-based censorship on TLS and QUIC traffic (§4.3). Since QUICstep transmits handshake packets through a handshake channel, it becomes much more difficult for censors to judge whether to block a particular connection or not. We find that QUICstep indeed successfully circumvents SNI-based censorship in the wild, including the recently implemented QUIC SNI censorship by the GFW.

4.1.3 Performance evaluation.
We provide a quantitative analysis of the latency of QUICstep to scenarios where all traffic is sent through the native channel or the handshake channel under different settings (§4.4). QUIC connection migration ensures that path switching between the direct and tunneled paths only introduces one RTT of latency (i.e., path validation), and does not disrupt the ongoing session between the client and the server. Compared to a native QUIC connection, QUICstep does incur some additional latency due to conducting the handshake over the handshake channel and path validation during path switching. However, we find that this latency overhead is amortized over the entire request and

QUICstep provides up to 84% reduction in page load time compared to using the handshake channel for all traffic.

We also show that QUICstep reduces load on the handshake channel provider by a median of 93% over 100 different websites (§4.4.3). High-latency channels incur hosting costs, such as for rendezvous hosts, Tor bridge volunteers, or for resilient VPN providers. Bandwidth is either explicitly limited due to the mode of transport, otherwise capped by providers, or simply lowered in practice due to the channel being often overloaded. By opportunistically migrating connections to the devices' native network, QUICstep minimizes bandwidth usage and reduces load for the high-latency channel.

4.2 QUICstep support: measuring compatible websites in the wild

To identify the scale of QUICstep's impact, we aim to measure the number of websites (domains) that support QUIC, are fully compatible with QUICstep, or only partially support connection migration. Fully compatible websites are ready to use with QUICstep, while QUIC-supporting websites that do not or partially support connection migration will benefit from QUICstep in the future as QUIC libraries continue to mature.

4.2.1 Experiment setup and methodology. We begin with identifying QUIC-supporting websites in the wild. To identify server-side support for QUIC, we sent HTTP/3 GET requests over QUIC to Tranco top 1 M domains [37], denoting support for QUIC if the client successfully connects to the server. We used the Chromium `quic_client` to perform these tests [26]. We then aim to find QUICstep-compatible websites among QUIC-support websites. To measure QUICstep compatibility we conducted HTTP/3 GET requests with QUICstep enabled and denoted success when the fetch succeeded without error. We examined packet captures during the connection for a sample of successful QUICstep fetches and verified that the connection was functioning after migration. We repeated requests for both the parent and `www` subdomain. We denoted success if either of the two succeeded. We excluded 404 Error pages served over QUIC, but included redirect pages. The client and handshake channel provider for QUICstep were located in North Virginia and Ohio, respectively.

4.2.2 A notable portion of QUIC websites are QUICstep-compatible. As of October 25, 2024, we found 219,729 websites (22%) among top 1 M that support QUIC. Out of these QUIC-supporting websites, 28,104 (12.8%) were compatible with QUICstep. While QUIC support is yet far from universal, we expect that in the future QUIC will be both more widespread (more entities supporting QUIC) and complete (QUIC implementations fully adhering to standards).

We further analyzed the associated network providers of these compatible domains. Table 1 shows top 10 QUIC providers within the top 1 M domains and QUICstep-compatibility in each provider. As observed, 74.6% of QUIC-supporting domains are using Cloudflare; however, only a small fraction (0.2%) of Cloudflare's QUIC-supporting domains were compatible with QUICstep. Upon further investigation, we found that this low compatibility rate stems from a specific limitation within the Cloudflare CDN: it does not yet fully support connection migration. Since connection migration support is expected as per the QUIC RFC [32], this demonstrates that even

Provider	# QUIC domain	# of QUICstep-compatible domains
Cloudflare	163900 (16.4%)	335 (0.2%)
Google	7191 (0.72%)	748 (10.4%)
Amazon	7067 (0.71%)	3858 (54.6%)
Hostinger	4738 (0.48%)	2770 (58.5%)
Fastly	4140 (0.41%)	2114 (51.1%)
Hetzner	2187 (0.22%)	1802 (82.4%)
Automattic	1675 (0.17%)	9 (0.5%)
Wix	1084 (0.11%)	0 (0.0%)
OVH	1068 (0.11%)	710 (66.5%)
Bigcommerce	896 (0.09%)	1 (0.1%)

Table 1: Top 10 QUIC providers and the proportion of QUICstep compatibility in each.

broadly-used implementations of QUIC are still lagging behind the expectations of the full specification. We also tested Cloudflare's open-source QUIC implementation, quiche, and confirmed that it currently lacks proper support for connection migration.
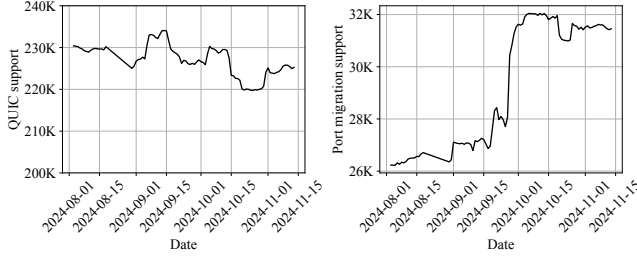
Hetzner has a particularly high support of connection migration among their QUIC-supporting domains. Most of the connection migration supporting domains in Hetzner (likely using Hetzner's hosting service) use the Litespeed server, which provides connection migration capability. This shows major cloud/hosting providers play a key role in the rollout of connection migration; a simple service update may significantly increase the number of websites able to benefit from connection migration.

We argue that some domain owners may be unaware they are missing out on QUIC's performance benefits due to the incompatibility of their websites' dependencies (CDN, load balancer, etc.) with standard QUIC. Some dependencies do not fully support all critical QUIC features defined in the RFC standards such as connection migration despite announcing QUIC support.

Another case is AWS CloudFront. CloudFront claims to support connection migration, but in practice support was often inconsistent. We will discuss more of this in §4.2.3.

4.2.3 The number of websites with partial support for connection migration is growing. In our measurements, we found websites that do not support IP address migration but support port migration. This is likely because QUIC libraries that web servers or their dependencies use fail to properly implement the connection migration feature. We anticipate these partial supporters could become compatible with QUICstep in the future as QUIC libraries continue to mature.

We conducted a long-term measurement to track these *potential* QUICstep-compatible websites. To measure port migration support, we leverage the ephemeral port migration option provided in the `quic_client`. This option performs an HTTP/3 GET request and then migrates to an ephemeral port to perform a second HTTP/3 GET request on the same connection. We note that some of those port migration supporters may already support IP address migration, which makes them fully QUICstep-compatible; nevertheless, we count them as potential. We performed daily scans on top 1 M domains from August 3, 2024 to November 13, 2024. Figure 3 shows

Figure 3: Number of websites that support (a) QUIC and (b) port migration from daily Tranco top 1M websites from August 3, 2024 to November 13, 2024. Support for port migration increased sharply around late September, 2024.

that while the number of QUIC-supporting websites has been fluctuating, the number of potential QUICstep-compatible websites has increased significantly. When we began measurements on August 3, 2024, 26,234 domains supported port migration. On September 26, 28,060 websites supported port migration; by September 29 it increased to 31,262 and the number has stayed above 31,000 since.

The increase in potential QUICstep-compatible websites could be attributed to QUIC library upgrades. For example, Varnish CDN did not support connection migration when we began the measurement, but our recent check confirms that it now fully supports port migration and some of its servers support IP address migration.

There are some special cases that support only IP address migration and not port migration. We exclude this case from measurement because nearly all of these domains are associated with the CloudFront CDN. Additionally, we found that different servers (distinguished by IP addresses) that host `amazon` domains have varying responses to IP address migration. Even for the same domain, some servers support IP address migration but some do not. We suspect there were deployment issues that caused the servers to use different versions of QUIC implementations or configurations (e.g., we have received "connection migration disabled by server" error messages from some servers but not others). We expect that robust and consistent migration support will extend to all servers in the near future.

4.2.4 The road to connection migration. Our measurements indicate that while connection migration adoption is on the rise, two major practical challenges could impede further deployment and usage: (1) QUIC implementations are not compliant with the RFC standard, and do not or only partially support connection migration. (2) The complex dependencies of modern websites demand robust support for connection migration at critical infrastructure points. Tackling these challenges requires cooperation between different stakeholders such as QUIC developers, application providers, and service providers. Particularly, due to the centralization of the Internet, service providers play a critical role in the process; their willingness to adopt updated implementations and configure the infrastructure to support connection migration will directly impact the feature's success. For example, if Cloudflare properly supports connection migration, **87.2%** of QUIC-supporting domains would be able to leverage connection migration.

We believe that a crucial step in advancing connection migration rollout is to develop a reliable tool for testing connection migration in both QUIC library development and QUIC-supporting service deployment. We see QUICstep as a stepping stone. As discussed above, connection migration involves complex cases that have often been overlooked and can only be discovered during actual migration events e.g., supporting port but not IP address migration. QUICstep can be viewed as a system that introduces artificial mobility events into the network to activate the migration features of native QUIC. With QUICstep, there is no need to *infer* a server's support for connection migration using indirect information, as done in previous work [13]. Instead, QUICstep obtains a *definitive* confirmation of migration support based on whether the migration has actually succeeded.

4.3 Effectiveness: practical censorship circumvention with QUICstep

We test the capability of QUICstep to circumvent real-world SNI-based censors, confirming that QUICstep is capable of accessing blocked websites.

4.3.1 Real-world TCP+TLS SNI censorship. Our first evaluation was against our local [anonymized institution] Palo Alto Networks firewall that performs TLS SNI-based censorship. We accessed a domain under our control that was blocked by the firewall via SNI censorship. We did not control the firewall, and the domain was added to the firewall by Palo Alto's automatic system due to the domain name being relevant to cryptocurrency.

Our tests verified that the domain was blocked via SNI-based censorship. DNS requests to this domain returned the correct IP address, TCP SYN -> SYN+ACK handshake with the IP address completed successfully, and directly connecting to the IP address without domain name also succeeded. However, we observed a middlebox inserting an RST packet into the connection immediately after the client sent a TLS `ClientHello` with this domain name in the SNI field.

We successfully established a QUIC session with this domain via QUICstep. We note that connecting to the web server with a native QUIC connection also succeeded, implying that SNI censorship performed by this firewall did not perform SNI-based blocking in QUIC traffic. Nevertheless, this does not diminish QUICstep's success as our findings confirm that QUICstep's applicability includes conventional TCP SNI censorship.

4.3.2 Real-world QUIC SNI censorship. We take particular interest in the case of GFW, one of the leading censorship systems, has recently begun selective censorship of QUIC traffic using QUIC SNI since April 2024 [76]. We tested QUICstep against real-world QUIC SNI censorship in China and found that QUICstep can effectively bypass QUIC SNI censorship. Websites that could not be reached with the native connection were reliably accessible with QUICstep.

We used an Alibaba VM in mainland China as the client and ran the `quic_client` to fetch websites (with SNI specified). We consulted the authors of [76] to understand the nature and extent of QUIC SNI censorship in China and obtained a list of QUIC SNI censored domains. During the time span of our experiments, we

varied the locations of handshake channel provider across different AWS regions to avoid leaving consistent traces.

Our first experiment used a QUIC server under our control, running on an AWS EC2 instance located in North Virginia. We first set up this server with both a non-censored hostname and a censored hostname (`youtube.com`). Both the native client and the QUICstep client were able to access the server with the QUIC SNI set as the non-censored hostname, over multiple repeated trials. With `youtube.com` as the QUIC SNI hostname, the native QUIC client was blocked after several repeated fetches. The blocking does not happen immediately because the censor inspects traffic in a probabilistic manner and blocks the connection to the destination after it detects unwanted traffic. However, the QUICstep client successfully accessed the server throughout 50 consecutive fetches.
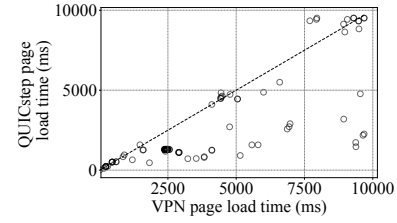
We also identified real-world domains that can be unblocked with QUICstep as discussed in §4.3.3 and tested them. We used a small sample of real-world websites for testing the feasibility of QUICstep. We tested 7 subdomains of `tiktokcdn.com` that were QUIC SNI blocked, but accessible with QUICstep. We repeatedly found that while the native connection failed after several fetches, QUICstep consistently succeeded in accessing the domains.

4.3.3   One-third of websites currently censored by QUIC SNI in China could potentially benefit from using QUICstep. We measured QUIC migration support of the websites censored by QUIC SNI in China. We find that connection migration support is high across websites that are censored under particular regimes. A concurrent work [76] tested the full Tranco list (~7 M websites) obtained on October 2, 2024[1] and found 28,458 domain names were on the GFW's QUIC SNI blocklist: If a QUIC `Initial` packet contains any of these domains as the SNI, the connection will be dropped. However, this serves more as a preventive measure since many of these domains do not support QUIC. We found that among these websites 2,404 **(8.45%)** support QUIC and among them 828 **(34.4%)** are compatible with QUICstep. QUICstep can unblock these websites if the client can identify some unblocked IP address associated with the domains.

We additionally tested QUICstep compatibility with websites that were censored by TCP SNI, as these websites could be gradually added to the GFW's QUIC SNI blocklist in the future. Using the methodology from [15] with a test list of 65,153,600 domains[2], we identified 5,700,928 websites censored by TCP SNI in China. Among these websites 3,524,808 **(61.8%)** support QUIC and among them 3,516,979 **(99.8%)** were compatible with QUICstep. This is due to a large proportion (99.2% of QUIC supporting domains) of `blogspot.com` and `wixsite.com` subdomains within the blocklist, both of which support QUICstep.

4.4   Performance evaluation

In this section we aim to understand the factors that affect performance of QUICstep with comparison to completely relying on the proxy under different settings. We investigate the effect of different configurations including proxy bandwidth, client location, and proxy location on page load time and time to first byte. We demonstrate that QUICstep significantly reduces the load on a VPN

Figure 4: Page load time of QUICstep and VPN connections for 100 different domains. Client in London, handshake channel provider in Ohio with a maximum throughput of 5 Mbps. QUICstep generally provides shorter page load time compared to VPN.

proxy compared to full VPN connections, and provides greater performance gain in bandwidth-limited environments. By optimizing proxy location and DNS resolution, the performance of QUICstep can be further boosted.

4.4.1   Methodology. In our default setting, our client is in London (AWS region), and the handshake channel provider is in Ohio, rate-limited to a maximum throughput of 5 Mbps. We used Chrome controlled by Selenium to visit a website that supports connection migration through the native connection, full VPN connection (i.e., tunneling all traffic via the handshake channel provider), and QUICstep 100 times. In each round, we alternated between the 3 connection types to mitigate the effects of network fluctuation. We recorded two performance metrics through Selenium: *Time to First Byte* (`responseStart-navigationStart`) and *Page Load Time* (`domComplete-responseStart`). Time to first byte (TTFB) includes the latency from the handshake, which occurs through the encrypted VPN channel. Therefore, we expect TTFB for QUICstep to be similar to that of the VPN. However, after the handshake, QUICstep fetches the website content through the native connection, so we anticipate a reduced page load time for QUICstep compared to the VPN.

In our evaluation, we varied client location, handshake channel provider location, and handshake channel bandwidth to understand their influence on QUICstep performance. We conducted our measurement with three client locations (London, New Jersey, and Osaka) and seven proxy locations (Frankfurt, Ireland, Montreal, Ohio, Oregon, Seoul, and Tokyo), resulting in a total of 21 location combinations. Besides, we set different rate limits on the handshake channel provider to emulate the bandwidth-limited secure channels real-world users would have. We tested a maximum throughput of 1 Mbps, 5 Mbps, 10 Mbps, and with no rate limit. These numbers were chosen to match the scale of the throughput of popular proxy services: Tor has a median throughput of around 10 Mbps [56], Psiphon's free version limits throughput to 2 Mbps [66, 68].
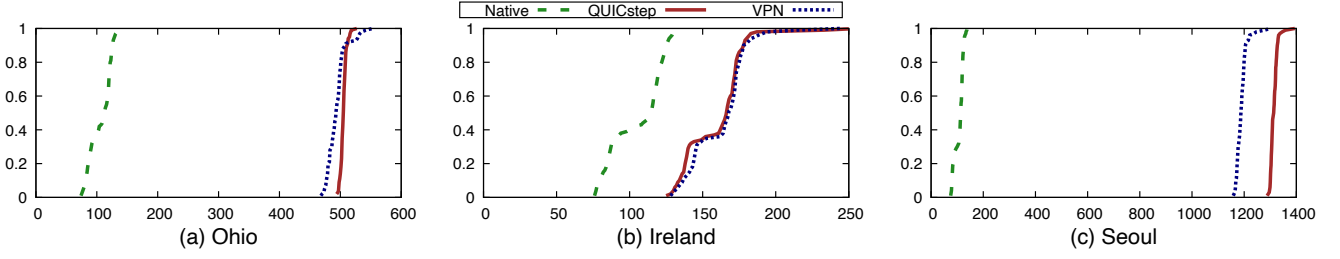
Unless explicitly mentioned, our observations are generally consistent across various settings, and for clarity and simplicity, we only report on the result from the default setting (client: London, handshake channel provider: Ohio, rate: 5 Mbps). The performance number is the median of 100 rounds of measurements.

4.4.2   QUICstep generally offers improved performance compared to full VPN. Our measurement quantifies the performance gain

---

[1]Domain list available at https://tranco-list.eu/list/664NX
[2]A combination of any domain that ever appeared in one of Alexa Top 1M, Tranco 1M, Cisco Umbrella 1M for at least one day between Jun 23, 2021 and Jun 23, 2022

Figure 5: CDF of time to first byte (in milliseconds) from 100 fetches of www.youtube.com with the client located in London and proxies located in Ohio, Ireland, Seoul with proxies' maximum throughput limited to 5 Mbps. QUICstep performance is comparable to the VPN connection.

QUICstep provides over full VPN connections. Figure 4 shows the distribution of VPN and QUICstep median page load time (across 20 rounds) for 100 different domains that are QUICstep-compatible in the Tranco top 1 K list from October 25, 2024[3]. The majority of tested domains experience shorter page load time when using QUICstep compared to VPN, with the time reduction being as great as 84%. As noted in §3.2.2, our implementation approximates handshake completion, so with a more accurate migration we may be able to achieve even greater reduction.

Our measurement also confirms our hypothesis about the TTFB and page load time discussed in §4.4.1. We show the TTFB and page load time of `www.youtube.com` under different client/proxy location combinations in Figure 5 and Figure 6, respectively. Here we chose `www.youtube.com` as our target website because it is the largest website that reliably supports connection migration. We can clearly observe that though the TTFB of QUICstep is comparable to that of full VPN, QUICstep provides a significant gain in page load time compared to full VPN.

4.4.3 QUICstep significantly reduces load on proxy (handshake channel provider). [4] The intuition behind the performance improvement is simple: QUICstep can significantly reduce the traffic that needs to go through the proxy (i.e., only handshake packets in QUICstep vs. full connection in conventional VPN). To quantitatively understand the load reduction benefit offered by QUICstep, we captured the packets at the proxy during each website visit when using QUICstep and VPN, and computed the traffic ratio. 'Traffic ratio' refers to the ratio of the size of traffic through proxy in a QUICstep connection to the size of traffic through proxy in a full VPN connection, and 'load reduction' refers to $1 - (\text{traffic ratio})$. As shown in Figure 7, **QUICstep reduced the load on the VPN proxy by a median of 93% compared to VPN**. In the case of `www.youtube.com`, there was 3.634 MB traffic through the proxy with the full VPN connection but only 96 KB with the QUICstep configuration, reducing load by 97.4%. This load reduction helps users alleviate costs for pay-as-you-go proxy servers or enables them to maximally utilize proxy services with data limits.

4.4.4 QUICstep can achieve performance comparable to native QUIC via optimizing proxy location. As expected, proxy (handshake channel provider) location affects the performance of QUICstep. When
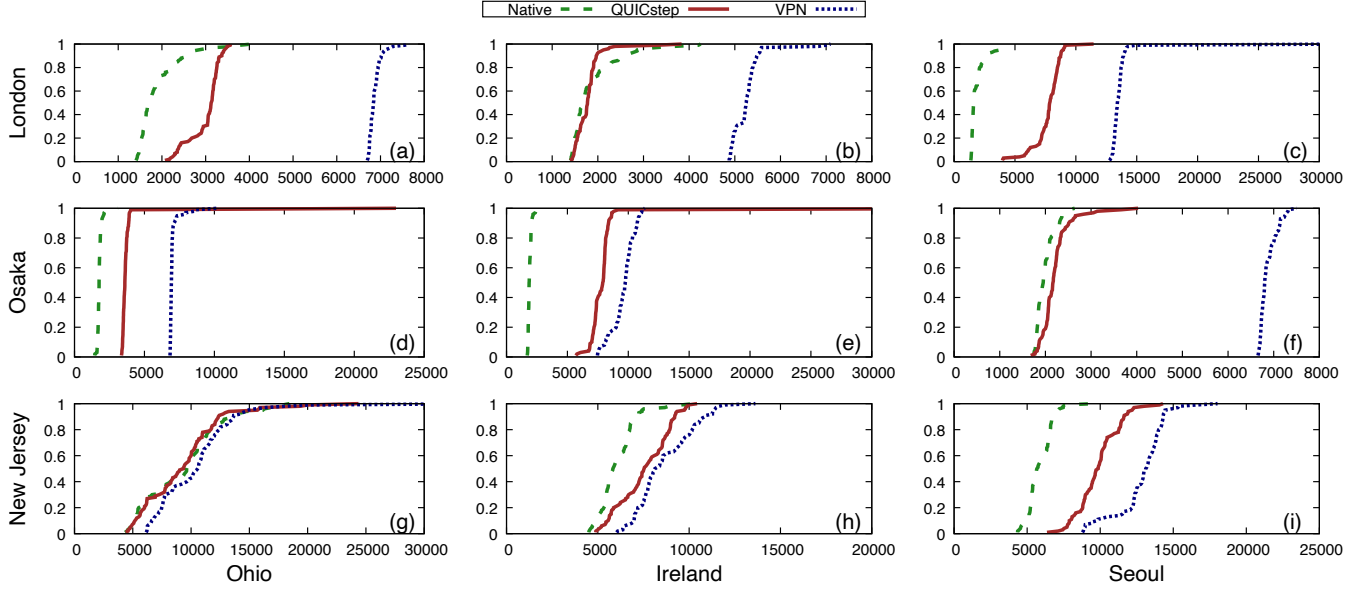
the proxy moves farther away (geographically) from the client, QUICstep adds more additional page load time to the native connection (Figure 6). One obvious reason for this performance degradation is that the round trip time between the client and proxy increases as the proxy moves farther away. However, another notable aspect is that changing the proxy location also changes the server's location since many websites are served through CDNs. In QUICstep (as well as VPN), the DNS request is performed through the proxy, so the client connects to a web server near the proxy; in the native connection, the client connects to a server near itself.

We believe server location could have a greater impact on performance than proxy round-trip time, given that the majority of traffic in QUICstep is sent directly to the server through the native connection. In fact, we do observe that in certain cases, the performance of QUICstep could be comparable to native QUIC (e.g., Figure 6 (b) (f) (g)). We hypothesize that in these cases our requests happened to be served by servers in the same geographical region. This also suggests that **the client could strategically choose a proxy location that is as geographically close as possible to the client's own location that is outside the censor's regime to achieve optimal performance**, especially when connecting to websites like `www.youtube.com` that are served on CDNs consisting of geographically diverse servers.
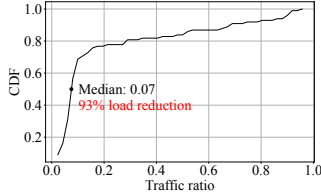
4.4.5 QUICstep provides greater performance gain in the bandwidth-limited environment. Recall that real-world proxies may only provide limited bandwidth. handshake channel providers may also further downgrade services when demand is high due to resource constraints [72]. To understand QUICstep's performance benefit in the bandwidth-limited environment, we examined the ratio of QUICstep page load time to VPN page load time while fetching `www.youtube.com` with varying proxy maximum throughput and client/proxy locations. A small ratio indicates a larger performance gain over VPN. Table 2 (full table in Appendix Table 4) shows the results, and we observe that **QUICstep's performance gain becomes more evident as proxy bandwidth decreases**. Particularly, when the proxy throughput is 1 Mbps or 5 Mbps, QUICstep always performs better than the VPN. When the bandwidth is 10 Mbps, there are some cases where the VPN outperforms QUICstep. This is likely due to AWS-based proxies being well-connected, making the proxy route potentially more efficient than the direct route between the client and server. Even when there is no rate limit, we were able to identify at least one proxy location where

---

[3]Domain list available at https://tranco-list.eu/list/N34YW
[4]For brevity, 'proxy' refers to handshake channel provider.

Figure 6: CDF of page load time (in milliseconds) from 100 fetches of www.youtube.com, with clients in London, Osaka, New Jersey (Y-axis) and handshake channel provider in Ohio, Ireland, Seoul (X-axis). The handshake channel provider's maximum throughput is limited to 5 Mbps. QUICstep outperforms the VPN connection in all locations and closely follows the performance of the native connection when the client and proxy are geographically proximate such as (b), (f), (g).
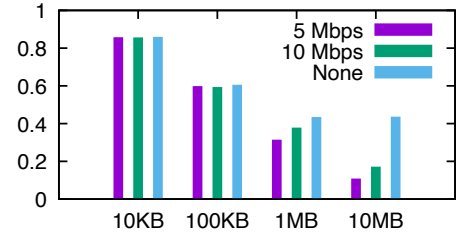

Figure 7: CDF of traffic ratio (defined in §4.4.3). QUICstep provides a 93% median load reduction.

Figure 8: The ratio of file download time of QUICstep over VPN under different rate limits and file sizes. A smaller ratio indicates better performance. The full result is in Table 5.

QUICstep provides comparable or better page load time than the VPN connection and incurs less than 5% latency overhead compared to the native connection. We observed a similar pattern for TTFB as in §4.4.2 and omit the results here to save space.

Overall, QUICstep can provide greater performance gain when the proxy has limited throughput. This is particularly appealing for real-world users, given that public VPN services or volunteer proxies are often throttled. By reducing the amount of traffic that must pass through the rate-limited channel, QUICstep effectively mitigates potential performance bottlenecks.

4.4.6 QUICstep provides greater performance gain for large websites. Another common factor that could affect browsing performance is website size. However, evaluating the impact of website size on QUICstep in a real-world setting is challenging, as various "noise" factors (e.g., dependencies on third-party servers) can affect performance estimation. Therefore, we performed a controlled experiment to eliminate noise. We set up our own QUIC server (using Google's QUICHE) and hosted files of varying sizes, and used the Chromium

`quic_client` to fetch the files under various proxy throughput limits. We show the ratio of file download time of QUICstep over VPN in Figure 8, and more detailed numbers are in Appendix Table 5. As expected, we see QUICstep provides greater performance gain over VPN when the file is larger. This pattern becomes more evident when observing QUICstep's overhead over the native latency (Appendix Table 5). QUICstep connection does not induce additional latency after handshake, so the latency overhead stays consistent regardless of file size.

We expect that for large websites, QUICstep provides greater performance gain over VPN because there is a greater proportion of traffic through the native channel.

4.4.7 Optimizing DNS resolution can further improve QUICstep performance. As we discussed in §4.4.4, the location where DNS queries are performed has a non-negligible impact on QUICstep

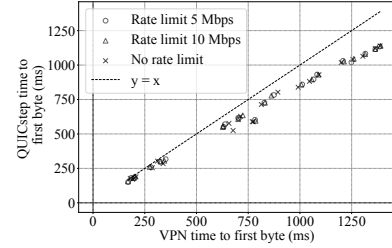| Client	Proxy	Max throughput 1 Mbps	Max throughput 5 Mbps	Max throughput 10 Mbps	No rate limit
	Ireland	0.069	0.338	0.510	0.996
	Frankfurt	0.070	0.340	0.616	0.898
	Montreal	0.072	0.463	0.738	1.052
London	Ohio	0.088	0.458	0.588	0.996
	Oregon	0.088	0.554	0.860	1.071
	Seoul	0.128	0.586	0.633	0.765
	Tokyo	0.124	0.682	0.976	1.553

Table 2: Ratio of QUICstep page load time to VPN page load time with different proxy locations and proxy rate limits. Proxies are listed in order of geographical distance from the client. The full result is in Table 4.

performance, because it may consequently influence the location of the server the client will connect to. Ideally, we would like the client to perform DNS queries by itself to ensure the selection of the most optimized server. However, this is not feasible in practice, as DNS censorship is the most basic and prevalent form of censorship. Censors often read plaintext DNS queries and interfere by blocking the query or injecting responses containing their own IP addresses [29, 34].

There are several approaches for the client and proxy to select an optimized server IP address:

(1) The client could perform an encrypted DNS request using methods like DNS over TLS (DoT) or DNS over HTTPS (DoH). The greatest challenge in using encrypted DNS is whether the client can access encrypted DNS resolvers, particularly since some countries are known to censor those resolvers [34]. There are several other considerations and limitations to using encrypted DNS for censorship circumvention. If the resolver is within the censor's regime, censors can manipulate the unencrypted traffic between the resolver and the nameserver [34]. Also, DoH downgrade to plaintext DNS is not uncommon in practice [38]. (2) The proxy can leverage EDNS Client Subnet (ECS) to reveal the client's network prefix to the authoritative DNS server. We note that anonymity is not a main concern for the client, as discussed in §3.1. If the target domain's authoritative DNS servers support ECS, the client will receive IP addresses that are closer in proximity to the client's network specified in the DNS query. We note that ECS support is not ubiquitously available across recursive resolvers (e.g., Cloudflare [17]) and authoritative servers. The proxy can use a modified DNS client or run a local recursive resolver for DNS resolution, as standard DNS clients do not natively support ECS. (3) The client may directly connect to the optimized server (or frontend) IP address obtained via an out-of-band channel (e.g., a system similar to Lox [58] and rBridge [65]).

We evaluated QUICstep's performance under the case where the client has some means of safely obtaining an optimized server IP address that is geographically close to itself. All native, VPN, and QUICstep connections used an IP address btained by the client, which eliminates the additional latency caused by the client connecting to a server far away from itself. Figure 9 shows that QUICstep marginally reduces even the TTFB compared to VPN, unlike Figure 5 where QUICstep and VPN had comparable TTFB latencies. Table 3 shows QUICstep's page load time compared to VPN and native connections with different proxy locations. QUICstep to VPN page load time ratio is significantly reduced for proxies far from the

Figure 9: Time to first byte for VPN and QUICstep connections when DNS is performed at the client. QUICstep shows better time to first byte than the VPN connection unlike the default setting where the two values are comparable.

client. For example, with a proxy in Tokyo, using QUICstep reduced latency by 32% over VPN when the client in London accessed a server near Tokyo. But latency reduction was as great as 70%, a gain 2.65 times more significant, when the client accessed a server near itself.

Again, it is important to note that while the DNS resolution process itself has manageable overhead, the primary factor is the server location determined by the DNS query location. By optimizing DNS resolution, QUICstep performance can be boosted significantly.

Summary of performance evaluations. QUICstep enables clients to make more efficient and effective use of secure handshake channels with limited performance by significantly reducing the amount of traffic that goes through the handshake channel. QUICstep's performance benefits are more pronounced when the handshake channel incurs greater increase in page load time compared to the native channel: when the handshake channel has low bandwidth and when the website is large. QUICstep's performance is strongly correlated with the location of the proxy and the server. Using a proxy close to the client and connecting to a server close to the client by leveraging ECS or out-of-band channels can enhance QUICstep performance.

5 Potential attacks against QUICstep

In this section, we explore potential attacks against QUICstep and their feasibility, starting from simple protocol blocking and progressing to more sophisticated traffic analysis. We primarily focus on practical attacks that have been/could be employed by real-world

Proxy	QUICstep/VPN		QUICstep/Native	
	Default	DNS optimized	Default	DNS optimized
Ireland	0.338	0.454	1.066	1.000
Frankfurt	0.340	0.438	1.328	0.997
Montreal	0.463	0.423	1.931	1.207
Ohio	0.458	0.449	1.847	1.175
Oregon	**0.554**	**0.238**	2.278	1.395
Seoul	**0.586**	**0.282**	4.920	2.949
Tokyo	**0.682**	**0.301**	3.842	2.444

Table 3: Ratio of QUICstep page load time to VPN and native connections when DNS resolution is optimized, compared to the default setup. Proxies are listed in order of distance from the client. The client is in London and the proxies' maximum throughput is 5 Mbps. DNS optimization significantly enhances QUICstep's performance gain over VPN when the proxy is far from the client (boldfaced).

censors, which are typically *stateless* as studied in [3, 70, 76]. We define a stateless attack as one that can be performed on a per-packet basis without requiring information from two or more packets. For example, QUIC SNI censorship can be stateless if the censor follows the reference implementation provided by Google [25].

DNS blocking and SNI censorship. DNS blocking [4, 29] and SNI censorship are two major techniques censors employ to censor websites. QUICstep evades DNS blocking and QUIC-SNI blocking as both DNS requests and QUIC `Handshake` packets are transmitted through a secure and encrypted tunnel that is not censored.

IP address blocking. Some censors may adopt IP address blocking [15]. As discussed in §3.1, following prior work, QUICstep is applicable to destinations whose IP addresses are not blocked. Censored domains can leverage non-blocked CDNs or cloud services to circumvent IP address blocking. Note that QUICstep sometimes helps circumvent IP address-based blocking. In QUICstep, DNS requests are performed through a proxy located outside of the censored regime and may resolve to a different IP address that may not be on the censor's IP address blocklist [15].

The censor may also attempt to block the IP address of the handshake channel provider. Handshake channel providers may remedy this by not making the IP addresses public like Tor bridges or frequently rotating IP addresses like Snowflake [11]. The design of QUICstep also helps reduce the risk of handshake channel provider IP addresses being identified compared to relying on VPNs for all communication. One way to fingerprint VPN usage is identifying when the client communicates predominantly over a single IP address; this does not happen with QUICstep as the client also communicates directly with the destination server.

Blocking all QUIC traffic. One attack strategy to counter QUICstep is to block all QUIC traffic. We argue that such an aggressive strategy would lead to high collateral damage, which could be undesirable for censors, given the rapid increase in QUIC/HTTP3 deployment. For example, major cloud providers in China (Alibaba, Tencent, Huawei, etc.) all provide QUIC-based services, and

therefore blocking QUIC could negatively impact the revenue of these Chinese companies. We noticed that Russia was suspected of blocking QUIC traffic in 2022 during the early stage of QUIC deployment [74]. This is likely because DPIs could not decrypt SNIs in encrypted QUIC payloads at that time. However, as QUIC SNI detection techniques have matured and been integrated into commercial DPI systems (e.g., Cisco [16]), it could incentivize nation-state censors to adopt QUIC SNI detection to minimize collateral damage. In fact, recent evidence indicates that Russia has transitioned to using QUIC SNI detection [69]. As discussed, QUICstep is an effective approach to bypass QUIC SNI censorship.

Blocking all QUIC connection migrated traffic. Theoretically, a censor could try to identify QUIC connection migration events and block all migrated QUIC connections. A potential indicator of a migrated connection is the absence of `Handshake` or `Initial` packets. However, such attacks face a significant challenge: QUICstep migrations are indistinguishable from those triggered by typical client mobility events in terms of traffic characteristics. One can view QUICstep as a system that uses artificial mobility events to trigger migration of native QUIC. There is no reliable way to differentiate between "normal" migration and QUICstep migration on a per-connection basis. Connection migration is likely to become common in future mobile and vehicular networks, which is a key consideration that inspired the design of QUIC's connection migration feature. Therefore, simply blocking all migrated connections may cause substantial collateral damage.

To evaluate the feasibility of censors dropping QUIC connections that do not contain `Handshake` or `Initial` packets, we analyzed QUIC traffic from a campus network collected for 24 hours on November 6th, 2022.[5] **Out of 3,786,050 unique QUIC connections, 1,100,439 (29.1%) did not contain a QUIC `Initial` nor a QUIC `Handshake` packet.** They are likely flows created from regular connection migration activity.

Hypothetical case: Stateful traffic analysis. As a lightweight approach, QUICstep does not consider stateful traffic analysis and makes no security claims against such attacks (e.g., detecting QUICstep based on the abnormal frequency of connection migration events). Several deployed censorship circumvention tools also share a similar limitation, e.g., some Tor pluggable transports and VPNs can be fingerprinted via traffic analysis [2, 62, 63, 73, 75]. We acknowledge that extensive research has been conducted on censorship circumvention techniques that are robust against stateful traffic analysis [5, 6, 12, 19, 24, 33, 46, 53]. However, we note that such attacks are resource-intensive, as they require the censor to store significantly more state information. Practical real-time DPIs still favor lightweight detection mechanisms such as keyword matching (Section 2.3.5). It remains an open question whether advanced traffic analysis attacks against QUICstep can achieve the efficiency required to meet real-time blocking goals of deployed censors.

---

[5] The network traces are anonymized and we have obtained IRB approval from our institution. Refer to §7 for details.

6 Discussion and conclusion

In summary, QUICstep presents a promising direction for censorship circumvention in a QUIC-first world. QUICstep successfully circumvents real-world censors and provides significant performance gain compared to relying on a secure but resource constrained channel for all communication. QUICstep is particularly useful in that it provides greater performance gain when the user fetches more data per-connection (e.g. when accessing large websites; §4.4.6) and when the user has limited resources (e.g. when the handshake channel has low bandwidth; §4.4.5).

The application-agnostic nature of QUICstep enables it to be integrated as part of existing censorship circumvention tools. Integration of QUICstep to existing proxy services can benefit providers as QUICstep substantially reduces resource consumption of each client. Our work also shows the current state of QUIC and connection migration support in the wild, and QUICstep can also be used as a tool for evaluating connection migration support.

In this section, we discuss the path to large-scale QUICstep deployment, for which the main bottleneck is the limited QUIC and connection migration support in the current Internet.

Increasing QUIC support and adoption. Rüth et al. reported 1.2% QUIC support among the Alexa top 1 M list in October 2017 [49]; our results find over 22% QUIC support among the Tranco top 1 M list in November 2024.[6] With the standardization of HTTP/3 and the ongoing increase in QUIC deployment we envision that QUIC will become the de facto norm for internet traffic [7, 61].

Increasing connection migration support and adoption. Our results in §4.2 demonstrate that connection migration support is increasing, but still does not extend to most QUIC-supporting websites. In our tests, we found QUICstep is not only useful for censorship circumvention, but also for understanding whether and how websites support QUIC connection migration. In this work, we provide the most comprehensive measurement of QUIC connection migration deployment to-date, differentiating between different types of support. We hope our work will be a driver for increased connection migration support, and that service providers (e.g., Cloudflare) will recognize the power of connection migration.

Standardizing a mechanism for advertising connection migration support. Due to the gap between QUIC and QUIC connection migration support as highlighted in our measurements, clients need to discover whether a host supports connection migration before they can leverage its performance benefits. Clients currently have no standard method for discovering connection migration support. As demonstrated by our own measurements in addition to prior work, it is not easy to reliably measure whether a website supports connection migration. In addition, many websites support only port or IP migration and not the other. The standardization of a method for discovering connection migration support, perhaps within an established QUIC channel, would allow all QUIC clients to benefit from the performance benefits of connection migration. At the moment, since it is nontrivial for clients to identify support for connection migration, clients need to maintain a list of endpoints that support connection migration.

Integrating QUICstep into usable deployments. In this work, we demonstrate that implementing the core functionality QUICstep is lightweight, and can be simplified to a number of packet-based routing rules. We suggest that with a mechanism for opportunistically discovering QUIC and connection migration support, QUICstep can be deployed as a standalone tool (e.g. packaged as an Android VPN), or better yet, alongside existing censorship circumvention tooling. The performance benefits of QUICstep also generalize to reducing bandwidth usage of a highly-secure handshake channel, which could be leveraged by existing circumvention software to reduce load on volunteer (or otherwise limited) providers.

7 Ethical considerations

Censorship circumvention experiments. Our censorship circumvention experiments in §4.3 did not involve any human subjects. In our measurements with the GFW, we used a client machine (under our control) hosted at a large-scale commercial VPS provider with a dedicated IP address. We accessed only a limited number of real-world domains to avoid the client machine itself being blocked. We ran a QUIC server under our control with different domain names for further testing. We note that this was run on localhost such that only our client machine (that knows the IP address of the QUIC server) could access it.

Network trace analysis. To evaluate the collateral damage of blocking all QUIC connection migrated traffic in §5, we analyzed anonymized traces gathered from our campus network. Port 443 UDP traffic was captured by our campus network operator who managed the tap, sanitized the traces, anonymized IP addresses and source ports, and removed all protected payloads before releasing the traces to us. The traces' network storage was also managed by our campus network operator and protected with restricted access. These traces were used by other campus projects as well and we did not have control over the lifetime of the data. We obtained IRB approval from our institution to study this data and we did not perform any analysis beyond the aggregate statistic presented in the paper.

References

[1] OONI: Open Observatory of Network Interference. In *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*, Bellevue, WA, August 2012. USENIX Association.

[2] Sultan Almutairi, Yogev Neumann, and Khaled Harfoush. Fingerprinting vpns with custom router firmware: A new censorship threat model. In *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, pages 976–981. IEEE, 2024.

[3] Anonymous and Amonymous. Sharing a modified Shadowsocks as well as our thoughts on the cat-and-mouse game, October 2022.

[4] Anonymous, Arian Akhavan Niaki, Nguyen Phong Hoang, Phillipa Gill, and Amir Houmansadr. Triplet censors: Demystifying Great Firewall's DNS censorship behavior. In *Free and Open Communications on the Internet.* USENIX, 2020.

---

[6]The 2017 result did not exclude 404 error pages. Including 404 error pages, our numbers jump to around 30%.

[5] Diogo Barradas, Nuno Santos, and Luís Rodrigues. DeltaShaper: Enabling unobservable censorship-resistant TCP tunneling over videoconferencing streams. *Proceedings on Privacy Enhancing Technologies*, 2017(4):5–22, 2017.

[6] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. Poking a hole in the wall: Efficient censorship-resistant Internet communications by parasitizing on WebRTC. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 35–48, 2020.

[7] David Belson and Lucas Pardue. Examining HTTP/3 usage one year on, June 2023.

[8] M Bishop. RFC 9114: HTTP/3, 2022.

[9] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. Geneva: Evolving Censorship Evasion Strategies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2199–2214, New York, NY, USA, 2019. Association for Computing Machinery.

[10] Kevin Bock, iyouport, Anonymous, Louis-Henri Merino, David Fifield, Amir Houmansadr, and Dave Levin. Exposing and circumventing china's censorship of esni. Technical report, GFW Report, August 2020.

[11] Cecylia Bocovich, Arlo Breault, David Fifield, Serene, and Xiaokang Wang. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2635–2652, Philadelphia, PA, August 2024. USENIX Association.

[12] Chad Brubaker, Amir Houmansadr, and Vitaly Shmatikov. Cloudtransport: Using cloud storage for censorship-resistant networking. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–20. Springer, 2014.

[13] Aurélien Buchet and Cristel Pelsser. An analysis of QUIC connection migration in the wild, 2024.

[14] Cormac Callanan, Hein Dries-Ziekenheiner, Alberto Escudero-Pascual, and Robert Guerra. Leaping over the firewall: A review of censorship circumvention tools. *Report by Freedom House*, 2011.

[15] Zimo Chai, Amirhossein Ghafari, and Amir Houmansadr. On the importance of encrypted-SNI (ESNI) to censorship circumvention. In *Free and Open Communications on the Internet*. USENIX, 2019.

[16] Cisco. Support for SNI Detection, unknown.

[17] Cloudflare. 1.1.1.1 (DNS Resolver) FAQ, 2024.

[18] Arjun Devraj, Liang Wang, and Jennifer Rexford. Redact: refraction networking from the data center. *ACM SIGCOMM Computer Communication Review*, 51(4):15–22, 2021.

[19] Kevin P Dyer, Scott E Coull, and Thomas Shrimpton. Marionette: A programmable network traffic obfuscation system. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 367–382, 2015.

[20] The eBPF Foundation. eBPF Documentation, 2024.

[21] Kathrin Elmenhorst. A Quick Look at QUIC Censorship, Apr 2022.

[22] Kathrin Elmenhorst, Bertram Schütz, Nils Aschenbruck, and Simone Basso. Web censorship measurements of HTTP/3 over QUIC. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 276–282, 2021.

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

[24] Gabriel Figueira, Diogo Barradas, and Nuno Santos. Stegozoa: Enhancing WebRTC Covert Channels with Video Steganography for Internet Censorship Circumvention. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 1154–1167, 2022.

[25] Google. Parsing QUIC Client Hellos, 2021.

[26] Google. QUICHE. https://quiche.googlesource.com/quiche/, 2022.

[27] Yashodhar Govil, Liang Wang, and Jennifer Rexford. MIMIQ: Masking IPs with migration in QUIC. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2020.

[28] Nguyen Phong Hoang, Jakub Dalek, Masashi Crete-Nishihata, Nicolas Christin, Vinod Yegneswaran, Michalis Polychronakis, and Nick Feamster. GFWeb: Measuring the Great Firewall's Web censorship at scale. In *USENIX Security Symposium*. USENIX, 2024.

[29] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How great is the Great Firewall? Measuring China's DNS censorship. In *USENIX Security Symposium*. USENIX, 2021.

[30] Amir Houmansadr, Thomas J Riedl, Nikita Borisov, and Andrew C Singer. I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention. In *NDSS*, 2013.

[31] Hysteria developers. Hysteria.

[32] Jana Iyengar and Martin Thomson. RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport. *Omtermet Emgomeeromg Task Force*, 2021.

[33] Watson Jia, Joseph Eichenhofer, Liang Wang, and Prateek Mittal. Voiceover: Censorship-circumventing protocol tunnels with generative modeling. *Free and Open Communications on the Internet*, 2023.

[34] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. Understanding the impact of encrypted DNS on internet censorship. In *Proceedings of the Web Conference 2021*, pages 484–495, 2021.

[35] Sedat Kappanoğlu. turkish ISPs use two methods for blocking access... https://twitter.com/esesci/status/1630024112071491586.

[36] Lantern. Lantern - Frequently Asked Questions.

[37] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, February 2019.

[38] Jinseo Lee, David Mohaisen, and Min Suk Kang. Measuring dns-over-https downgrades: Prevalence, techniques, and bypass strategies. *Proceedings of the ACM on Networking*, 2(CoNEXT4):1–22, 2024.

[39] Victoria Manfredi and Pi Songkuntham. MultiFlow: Cross-Connection decoy routing using TLS 1.3 session resumption. In *8th USENIX Workshop on Free and Open Communications on the Internet (FOCI 18)*, Baltimore, MD, August 2018. USENIX Association.

[40] Hooman Mohajeri Moghaddam, Baiyu Li, Mohammad Derakhshani, and Ian Goldberg. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 97–108, 2012.

[41] Milad Nasr, Hadi Zolfaghari, Amir Houmansadr, and Amirhossein Ghafari. Massbrowser: Unblocking the censored web for the masses, by the masses. In *NDSS*, 2020.

[42] Niklas Niere, Felix Lange, Nico Heitmann, and Juraj Somorovsky. Encrypted client hello (ech) in censorship circumvention. *Free and Open Communications on the Internet*, 2025.

[43] Tor Project. meek, 2020.

[44] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[45] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-25, Internet Engineering Task Force, June 2025. Work in Progress.

[46] Marc B Rosen, James Parker, and Alex J Malozemoff. Balboa: Bobbing and weaving around network censorship. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3399–3413, 2021.

[47] Zach Rosson, Felicia, Carolyn Tackett, and Meabh Maguire. Lives on hold: internet shutdowns in 2024, February 2025.

[48] Shadowsocks rust developers. Shadowsocks-rust.

[49] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. A first look at quic in the wild. In *Passive and Active Measurement: 19th International Conference, PAM 2018, Berlin, Germany, March 26–27, 2018, Proceedings 19*, pages 255–268. Springer, 2018.

[50] Sambhav Satija and Rahul Chatterjee. BlindTLS: Circumventing TLS-based HTTPS censorship. In *Free and Open Communications on the Internet*. ACM, 2021.

[51] Shadowsocks developers. Shadowsocks.

[52] Sing-box developers. Sing-box.

[53] Zhen Sun and Vitaly Shmatikov. Telepath: A minecraft-based covert communication system. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2223–2237. IEEE, 2023.

[54] Ram Sundara Raman, Prerana Shenoy, Katharina Kohls, and Roya Ensafi. Censored Planet: An Internet-Wide, Longitudinal Censorship Observatory. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 49–66, New York, NY, USA, 2020. Association for Computing Machinery.

[55] Ram Sundara Raman, Adrian Stoll, Jakub Dalek, Reethika Ramesh, Will Scott, and Roya Ensafi. Measuring the Deployment of Network Censorship Filters at Global Scale. In *NDSS*, 2020.

[56] The Tor Project. Tor metrics - performance, 2024.

[57] Martin Thomson and Sean Turner. Using TLS to Secure QUIC. RFC 9001, May 2021.

[58] Lindsey Tulloch. Lox: Protecting the social graph in bridge distribution. Master's thesis, University of Waterloo, 2022.

[59] V2Ray developers. V2Ray.

[60] Paul Vines, Samuel McKay, Jesse Jenter, and Suresh Krishnaswamy. Communication breakdown: Modularizing application tunneling for signaling around censorship. *Privacy Enhancing Technologies*, 2024(1), 2024.

[61] W3Techs. Usage statistics of http/3 for websites, May 2025.

[62] Ryan Wails, George Arnold Sullivan, Micah Sherr, and Rob Jansen. On precisely detecting censorship circumvention in real-world networks. In *Network and Distributed System Security*, 2024.

[63] Liang Wang, Kevin P Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. Seeing through network-protocol obfuscation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 57–69, 2015.

[64] Mona Wang, Anunay Kulshrestha, Liang Wang, and Prateek Mittal. Leveraging strategic connection migration-powered traffic splitting for privacy. In *Proceedings on Privacy Enhancing Technologies*, page 498–515, 2022.

[65] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas Hopper. rbridge: User reputation based tor bridge distribution with privacy preservation. In *NDSS*, 2013.

[66] Jon Watson. How to Use Psiphon The Censorship-Circumvention Tool, 2021.

[67] Mingkui Wei. Domain shadowing: Leveraging content delivery networks for robust Blocking-Resistant communications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3327–3343. USENIX Association, August 2021.

[68] Mike Williams. Pshiphon review, 2020.

[69] wkrp. Throttling→blocking of YouTube in Russia, 2024-07-12, 2024.

[70] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. How the Great Firewall of China detects and blocks fully encrypted traffic. In *USENIX Security Symposium*. USENIX, 2023.

[71] XRay developers. XRay.

[72] Diwen Xue, Anna Ablove, Reethika Ramesh, Grace Kwak Danciu, and Roya Ensafi. Bridging barriers: A survey of challenges and priorities in the censorship circumvention landscape. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2671–2688, Philadelphia, PA, August 2024. USENIX Association.

[73] Diwen Xue, Michalis Kallitsis, Amir Houmansadr, and Roya Ensafi. Fingerprinting obfuscated proxy traffic with encapsulated {TLS} handshakes. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2689–2706, 2024.

[74] Diwen Xue, Benjamin Mixon-Baca, ValdikSS, Anna Ablove, Beau Kujath, Jedidiah R Crandall, and Roya Ensafi. Tspu: Russia's decentralized censorship system. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 179–194, 2022.

[75] Diwen Xue, Reethika Ramesh, Arham Jain, Michaelis Kallitsis, J Alex Halderman, Jedidiah R Crandall, and Roya Ensafi. OpenVPN is open to VPN fingerprinting. *Communications of the ACM*, 2022.

[76] Ali Zohaib, Qiang Zao, Jackson Sippe, Abdulrahman Alaraj, Amir Houmansadr, Zakir Durumeric, and Eric Wustrow. Exposing and circumventing SNI-based QUIC censorship of the Great Firewall of China. In *USENIX Security Symposium*. USENIX, 2025.

A   Additional performance evaluations

Client	Proxy	Max throughput 1 Mbps	Max throughput 5 Mbps	Max throughput 10 Mbps	No rate limit
London	Ireland	0.069	0.338	0.510	0.996
	Frankfurt	0.070	0.340	0.616	0.898
	Montreal	0.072	0.463	0.738	1.052
	Ohio	0.088	0.458	0.588	0.996
	Oregon	0.088	0.554	0.860	1.071
	Seoul	0.128	0.586	0.633	0.765
	Tokyo	0.124	0.682	0.976	1.553
Osaka	Tokyo	0.079	0.269	0.449	0.971
	Seoul	0.079	0.319	0.511	0.868
	Oregon	0.095	0.476	0.740	1.027
	Frankfurt	0.255	0.719	1.015	1.273
	Ireland	0.223	0.811	1.293	1.183
	Montreal	0.136	0.560	0.752	1.002
	Ohio	0.134	0.519	0.751	0.965
New Jersey	Montreal	0.140	0.876	0.956	0.963
	Ohio	0.147	0.882	0.977	0.973
	Oregon	0.156	0.895	0.932	1.010
	Ireland	0.210	0.921	1.194	1.065
	Frankfurt	0.213	0.952	1.216	1.174
	Tokyo	0.236	0.951	1.032	1.117
	Seoul	0.265	0.750	0.834	0.776

Table 4: Ratio of QUICstep page load time to VPN page load time with different client locations, proxy locations, and proxy rate limits. For each client location the proxies are listed in order of geographical distance from the client.

Rate limit	Website size	Native (ms)	VPN (ms)	QUICstep (ms)	QUICstep / VPN	QUICstep / Native	QUICstep - Native (ms)
5 Mbps	10 KB	98.97	926.26	790.41	0.853	7.986	691.44
	100 KB	145.11	1411.53	837.77	0.594	5.773	692.66
	1 MB	277.4	3128.94	971.22	0.310	3.501	693.82
	10 MB	1272.28	18894.73	1962.12	0.104	1.542	689.84
10 Mbps	10 KB	97.22	928.26	791.27	0.852	8.139	694.05
	100 KB	138.41	1409.3	830.78	0.589	6.002	692.37
	1 MB	262.29	2559.01	956.57	0.374	3.647	694.28
	10 MB	1040.3	10431.35	1737.8	0.167	1.670	697.50
None	10 KB	104.99	924.11	790.55	0.855	7.530	685.56
	100 KB	157.57	1404.99	844.25	0.601	5.358	686.68
	1 MB	351.62	2413.92	1038.42	0.430	2.953	686.80
	10 MB	999.588	3875.32	1674.96	0.432	1.676	675.37

Table 5: Latency (ms) with different rate limits and website sizes.