From the holdings of the Department of Special Collections , Princeton University Library, Princeton, NJ

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyrightlaw. We do not charge any permission or usage fees for the publication of images of material inour collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff.For more information regarding copyright, credit, and citation please review the information onthe Copyright, Credit and Citation Guidelines page of our website.
https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. https://library.princeton.edu/special-collections/ask-us

---

Master's Theses, 2022-2026 Computer Science, 2023

2023

QUICstep: Circumventing QUIC-based Censorsh

Jia, Watson

Princeton, NJ : Princeton University

---

http://arks.princeton.edu/ark:/88435/dsp014t64gr47d

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. https://www.copyright.gov/title17/title17.pdf

QUICstep: Circumventing QUIC-based Censorship

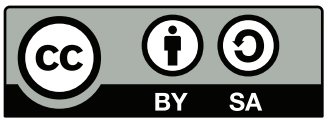
Watson Jia

a master's thesis
presented to the faculty
of princeton university
in candidacy for the degree
of Master of Science in Engineering

recommended for acceptance
by the Department of
Computer Science

Adviser: Prateek Mittal

May 2023



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License (https://creativecommons.org/licenses/by-sa/4.0/)

Abstract

Governments around the world limit free and open communication on the Internet through censorship. To reliably identify and block access to certain web domains, censors inspect the plaintext TLS SNI field sent in TLS handshakes. With QUIC rapidly displacing TCP as the dominant transport-layer protocol on the web, censorship regimes have already begun prosecuting network traffic delivered over QUIC. With QUIC censorship poised to expand, censorship circumvention tools must similarly adapt. We present QUICstep, a censorship-resilient, application-agnostic, performant, and easy-to-implement approach to censorship circumvention in the QUIC era. QUICstep circumvents TLS SNI censorship by conducting a QUIC-TLS handshake over an encrypted tunnel to hide the SNI field from censors and performs *connection migration* to resume the QUIC session in plain sight of the censor. Our evaluation finds that QUICstep successfully establishes QUIC sessions in the presence of a proof-of-concept censor with minimal latency overhead.

Contents

Abstract 3

1 Introduction 8

2 Background and Related Work 11
- 2.1 QUIC . . . 12
- 2.2 Censorship circumvention . . . 14

3 Design and Implementation 16
- 3.1 Threat model . . . 17
- 3.2 System design . . . 17
- 3.3 Implementation . . . 20

4 Evaluation 21
- 4.1 Feasibility evaluation against a proof-of-concept QUIC censor . . . 23
- 4.2 Performance evaluation . . . 24
- 4.3 Measuring server support for QUIC connection migration . . . 25

5 Discussion 28
- 5.1 Deployment challenges . . . 29
- 5.2 Other QUICstep-like systems . . . 32

6 Conclusion 34

References 39

Listing of figures

2.1 This figure demonstrates our adversary model and how QUICstep can be leveraged to circumvent censorship. (a) demonstrates an adversary censor capable of monitoring client traffic, blocking or disrupting 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. . . 14

4.1 Diagram of setup for feasibility and performance evaluations. Though QUICstep works in theory with any encrypted tunnel, for our proof-of-concept we implement QUICstep using a Wireguard tunnel. We selectively send certain types of traffic (DNS traffic, QUIC-TLS handshake traffic) over the Wireguard tunnel and the rest over the native network. 22

4.2 The CDFs for the latency of 250 HTTP/3 GET requests for a 1 MB or 10 MB file using QUICstep, a censorship-vulnerable low latency link, and a secure yet high latency tunnel. The Wireguard proxy is varied between two locations, Ohio and Oregon. We can see QUICstep has minimal additional latency when compared to a native network connection while conducting the entire request through a proxy incurs much more latency. 25

4.3 Pie charts illustrating the current state of server-side support for QUIC connection migration, among the top 10K domains of the Majestic Million domain name list as of April 2023, are shown. All HTTP/3 connections were conducted using the Chromium QUICHE and Firefox Neqo clients. . . . 27

To Mom, Dad, Jasmine, Jessica, Ron, and Mary.

Acknowledgments

The research conducted in this thesis led to *QUICstep: Circumventing QUIC-based Censorship* by Watson Jia, Mona Wang, Liang Wang, and Prateek Mittal of which an earlier version was made available on arXiv as a preprint and is currently under review. The analyses and text in this thesis are based on the preprint, and so this thesis can be considered an updated version of the preprint.

I want to thank my adviser, Professor Prateek Mittal, for encouraging me to pursue this research topic. His continued advice, guidance, and support have greatly shaped this thesis. I want to thank Mona Wang for all her guidance starting from day one. This work would not be possible without her patience and support over the course of many Zoom and Slack meetings. I want to thank Liang Wang for all our insightful discussions. His inputs were valuable in thinking through all of the high-level questions and low-level details. I also want to thank Professor Jonathan Mayer for serving as my reader.

I want to thank my family - my mother, father, and sisters Jasmine and Jessica - for their unconditional love and support. I also want to thank Ron and Mary who I owe a debt of gratitude for all the time and sacrifices they have made over the years. I also want to thank all my friends who kept me sane throughout this process.

1

Introduction

Internet censors implement a variety of techniques to selectively identify connections to censored websites using deep packet inspection (DPI) middleboxes [1]. Today, one widely used technique is to inspect the server name indication (SNI) or other fields present in TLS handshakes [2]. Censorship circumvention tools have been proposed to enable access to censored websites even in the presence of blocking. However, tools such as Tor and VPNs

can be relatively high-latency and low-bandwidth when compared to native network connections. In addition to performance considerations, operating networks as part of VPN deployments can be expensive.

Most past advances in censorship circumvention have focused on TCP-based protocols or fooling TCP session state management on middleboxes [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. However, the Internet is rapidly moving towards next-generation transport protocols like QUIC. The next generation of web protocols, HTTP/3, is built on top of QUIC. Already, QUIC is supported in many major browsers, and 25% of the top 10 million websites already support HTTP/3 [16, 17, 18, 19]. In early 2022 one firm reported up to 16% of Internet traffic in North America was over QUIC, and 30% of traffic in Europe, the Middle East, and Africa [20]. These numbers are growing, and as this shift occurs, censors will soon adopt new methods to manage QUIC traffic, and circumvention methods must also adjust to the new landscape.

We propose QUICstep, a censorship circumvention technique that circumvents network censorship by leveraging the connection migration capability of QUIC, which is originally designed for seamless mobility to allow the client's IP address to change within the life of a connection. QUICstep conducts a QUIC-TLS handshake over an encrypted tunnel and leverages connection migration to seamlessly migrate the session to the default, native network connection upon completion of the handshake, after which the client resumes their QUIC session directly with a censored server. The encrypted tunnel hides any plaintext handshake fields from the censor, allowing a client to establish a QUIC session even in the presence of censorship. Connection migration enables client-server communication over the lower-latency default network connection, and QUICstep introduces

minimal overhead by performing only one connection migration and minimizing the number of packets sent over a higher latency encrypted tunnel. QUICstep additionally imposes no other protocol requirements other than QUIC connection migration, enabling compatibility with a wide variety of system and application environments.

We implement and evaluate a prototype of QUICstep against a proof-of-concept censor we developed and demonstrate that QUICstep allows a client to establish a QUIC session with a server in the presence of censorship. We also evaluate the performance of a QUICstep connection by comparing its latency to that of a censorship-vulnerable native connection and a fully-proxied connection. We show that there is minimal additional latency using our approach when compared to a native connection, and significant performance advantages when compared to a fully-proxied connection. We additionally measure the current state of QUIC connection migration support among popular web domains. Finally, in our discussion, we provide suggestions for future work that will enable widespread deployment of QUICstep as the Internet moves towards QUIC.

2

Background and Related Work

We provide an overview of QUIC, including a summary of connection migration, QUIC-TLS handshakes, QUIC censorship, and usage of QUIC in building privacy enhancing tools. Censorship and censorship circumvention are areas that are composed of social, economic, and technical elements - we focus on the technical aspects of censorship. We also summarize recent related work leading up to the problem we address in this paper.

2.1 QUIC

QUIC is a transport layer protocol based on UDP and supports multiplexing of application-layer data streams [21]. QUIC was developed to improve the performance of web applications compared to TCP and forms the basis of HTTP/3. QUIC is being rapidly adopted and major large services and CDNs are spearheading its deployment. The meteoric rise of QUIC is likely to continue as HTTP/3 has been standardized as an RFC [22].

Connection migration. QUIC connections can be migrated across networks as they are no longer tied to the TCP 4-tuple; rather, QUIC connections utilize a set of connection identifiers for servers to uniquely identify clients. The decoupling of QUIC connections from IP addresses and ports allows connections to be maintained even as clients move between different networks, which is known as *connection migration*. When a client detects a network change, the client performs a round-trip path validation to ensure that the server is still reachable before resuming the connection. Connection migration can only occur after a session has been fully established between a client and server in a QUIC-TLS handshake. QUIC's connection migration enables massive performance improvements for mobile users, as it persists connections even when clients move across networks. Our paper makes a novel use of this feature originally designed for mobile performance, in the context of circumventing censorship.

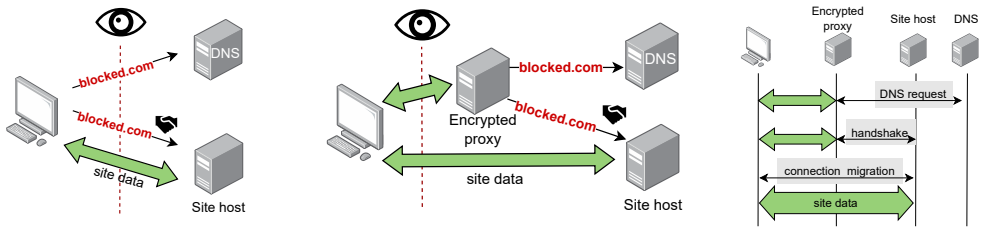
QUIC-TLS handshake. QUIC is designed as a secure-by-default protocol that mandates encryption of data in transit. To achieve this, QUIC is integrated with TLS to provide transport layer security. Compared to traditional TLS over TCP, QUIC-TLS has several security enhancements by rolling together the QUIC and TLS handshake, in addition to a

performance gain resulting from eliminating a round-trip during the handshake. Notably, QUIC-TLS also encrypts the initial packets exchanged during a handshake which can foil naive SNI censorship by DPI middleboxes. Recently in Turkey, after the blocking of a social media website when users criticized the government handling of the 2023 Turkey-Syria earthquake fallout, the site developers reported that users could circumvent SNI-based censorship by forcing a QUIC connection on the domain [23].

However, the encryption of QUIC-TLS handshake packets does not offer strong confidentiality assurances since the keys used to encrypt these initial packets are accessible to anyone observing the connection. Eavesdroppers can extract these keys and decrypt the handshake packets to inspect the contents of these packets, including the SNI field. While this process requires additional effort by network censors, network-level adversaries can still scrutinize QUIC-TLS handshake packets and impose censorship based on the TLS SNI field, thereby making HTTP/3 connections vulnerable to censorship.

QUIC-based censorship. QUIC is not yet universally deployed, but studies have measured and found censorship, incidental or otherwise, of QUIC and HTTP/3 traffic in countries like China, India, and Iran [24]. More recently, evidence of QUIC censorship has been recorded in Russia, China, India, and Uganda - in particular, Russian ISPs may be deploying deep packet inspection with SNI blacklists [25].

QUIC and other privacy use cases. Other work has explored the usage of QUIC properties in building privacy-enhancing technologies. MIMIQ utilizes QUIC's connection migration to obscure IP addresses from network observers [26]. CoMPS utilizes connection migration to split encrypted traffic across various network paths as a web fingerprinting defense [27]. TurboTunnel proposes incorporating transport-layer protocols like QUIC to



(a) Adversary model **(b)** QUICstep design **(c)** QUICstep network requests

Figure 2.1: This figure demonstrates our adversary model and how QUICstep can be leveraged to circumvent censorship. (a) demonstrates an adversary censor capable of monitoring client traffic, blocking or disrupting 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.

build session and reliability guarantees into censorship circumvention systems [28].

2.2 Censorship circumvention

Many approaches proposed to circumvent network censorship involve various forms of encrypted tunneling, including use of Tor or VPN. In the face of overzealous IP blocking, active probing, and other techniques to prevent connections to VPNs or Tor nodes, researchers at Tor and elsewhere have developed various pluggable transports to obfuscate circumvention traffic, including obfs4, meek, and Snowflake [11, 13, 29]. Other reliable circumvention techniques rely on hiding or otherwise obfuscating the server name from the TLS handshake, including domain fronting and domain hiding [14]. However, some CDN providers no longer support connections with mismatching SNI and HTTP hostnames, or connections with both SNI and ESNI fields [30, 31].

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 es-

tablished, 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 [32, 33]. 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 using TLS session resumption [34]. However, this approach focuses on session resumption in TLS 1.2, and it is not yet clear whether BlindTLS is compatible with session resumption in TLS 1.3. In contrast with BlindTLS, our approach is compatible with TLS 1.3 and is designed to be independent of TLS version. Moreover, SNI censorship circumvention literature has focused on the traditional TCP setting, and the web's transition to QUIC opens up new research areas. Our work is the first to thoroughly investigate leveraging QUIC connection migration to circumvent SNI censorship.

CoMPS. Finally, a QUICstep-like architecture was briefly mentioned by Wang et. al as a potential downstream use case for CoMPS, a connection-migration traffic splitting framework. The main purpose of CoMPS was to improve robustness against website fingerprinting, so the QUICstep-like architecture was not evaluated for deployment feasibility or performance for the censorship circumvention use case [27]. In this paper, we implement and evaluate the proof-of-concept, and discuss practical deployment challenges that may arise for circumvention developers.

3

Design and Implementation

In this section, we describe the design of QUICstep. Our primary design goals are to circumvent SNI censorship while minimizing overall latency overhead. Additionally, our design aims to minimize modifications to client-side software and avoid requiring modifications to server-side software. Moreover, our design aims to maximize flexibility and compatibility by being agnostic to underlying client technologies and protocols, including

but not limited to TLS version, encrypted tunnel, operating system, and browser. Our contribution is an easy-to-implement, performant proof-of-concept censorship circumvention tool whose only underlying technical requirement is client and server support for QUIC connection migration.

3.1 Threat model

A client in a censored network seeks to access a censored domain, which is hosted outside the censored network, through HTTP/3.The censor uses DPI techniques such as DNS and SNI filtering to identify and prevent such access. Specifically, the censor has the capability to decrypt QUIC-TLS initial packets and reveal the TLS SNI field. We assume that the client already has access to some encrypted tunnel (e.g. VPNs, Tor) outside of the censored network, and that this tunnel is not blocked by the censor. We assume that the censor is unable to break the security guarantees of QUIC or encrypted tunnels, and the censor will not attack the availability of certain classes of web traffic as part of censorship (e.g. blocking all QUIC traffic). While a censor could choose to do so, this would incur additional undesirable social and economic costs as this would likely affect non-censored web traffic. Figure 2.1a illustrates a censored network that is a representation of our threat model.

3.2 System design

The design of QUICstep seeks to achieve the following goals:

- *Censorship-resilience.* The censor is not able to learn the censored domain the client is visiting by inspecting or analyzing the network traffic produced by our approach.

- *Application-agnostic.* Our approach shall 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.

- *Low performance overhead.* Our approach shall introduce minimal performance overhead compared to existing censorship circumvention techniques.

Driven by these goals, we present QUICstep. Figure 2.1b depicts the architecture of QUICstep at a high level. The following are features of QUICstep, each presented in the context of the above design goals.

Hide handshakes using encrypted network paths. In QUICstep, the client maintains two distinct network paths to the QUIC server that hosts the censored domain: a low-latency, non-encrypted (direct) path, and a high-latency encrypted tunnel. [*] To circumvent SNI censorship, the client conducts a QUIC-TLS handshake with a QUIC server over the encrypted tunnel, hiding the TLS SNI value (and DNS requests) from the censor. Assuming that the censor is not able to violate the confidentiality guarantees of the encrypted tunnel, QUICstep elides censorship based on sensitive values like DNS hostname or TLS SNI.

Improve performance by switching to unencrypted network paths. After the completion of the QUIC-TLS handshake, a QUIC session is established between the server and the client. The client then immediately migrates the connection to the unencrypted network path to continue the communication directly with the server. The rest of the packets will be sent over a *native* QUIC connection with no extra tunnels, which minimizes the

[*] By "non-encrypted" we mean no extra encryption layer beyond what is already provided by the application protocols themselves.

latency of QUICstep.

Seamless path switching via connection migration. The connection migration feature of QUIC ensures that path switching between the unencrypted and encrypted paths only introduces one RTT of latency (i.e., path validation), and will not disrupt the ongoing session between the client and the server.

In comparison to a native QUIC connection, QUICstep incurs some additional latency due to conducting the handshake over the encryption tunnel and path validation during path switching. However, we expect that this latency overhead will be amortized since most of the time the client communicates directly with the server over a native connection. We also expect that QUICstep will be more performant than other tunnel-based censorship circumvention approaches such as Tor and VPNs.

Flexibility in a mixed TCP-QUIC world. Although the adoption of QUIC-based protocols continues to grow, security and protocol upgrades can take a long time to percolate throughout the entire Internet. We may expect that most web connections will be driven by a mixed combination of TCP and QUIC connections in the long-term, with web services slowly porting over to QUIC for the protocol upgrade and performance gain. QUICstep in practice also works well for this mixed-protocol world.

QUICstep imposes no restrictions to upper-layer protocols or client applications being used, and does not necessitate any changes apart from requiring that the client and server employ QUIC implementations that enable connection migration. We will demonstrate that QUICstep can be easily instantiated with only four routing and firewall rules and one shell script.

3.3 Implementation

During a QUIC-TLS handshake, a client will first make a DNS query to a censored domain. Since DNS hostnames are sensitive and often in plaintext, we add routing rules using `ip rule` to route DNS queries over the encrypted tunnel to circumvent DNS censorship. We then route QUIC-TLS handshake packets through the encrypted tunnel to circumvent TLS-SNI censorship. To do this, we perform a firewall match for the transport layer protocols associated with QUIC and the headers present in a QUIC-TLS handshake using `iptables`. We believe this implementation to be robust given that DNS and QUIC traffic adhere to their specifications. After the handshake, both sides start sending packets that are non-handshake packets - these packets are not matched by the firewall and are instead routed over the native network. This induces the connection migration process from either peer who recognizes the network change, causing a round-trip path validation. To easily spin up an encrypted tunnel along with these routing and firewall rules, we chose Wireguard as our VPN provider due to its ease and control over configuring network interfaces. We use a shell script to instantiate a Wireguard interface with our desired routing and firewall rules.

Figure 2.1c illustrates the implementation of QUICstep. The code for this proof-of-concept implementation, and for our evaluations, is made available at https://github.com/inspire-group/quicstep.

4

Evaluation

We evaluate our proof-of-concept QUICstep implementation. We first evaluate QUICstep in its ability to circumvent a proof-of-concept censor that blocks QUIC-TLS handshake packets. We then evaluate the performance and show that the latency overhead of QUICstep is amortized over the whole connection. Finally, we evaluate the current state of support for QUIC connection migration among popular web domains.

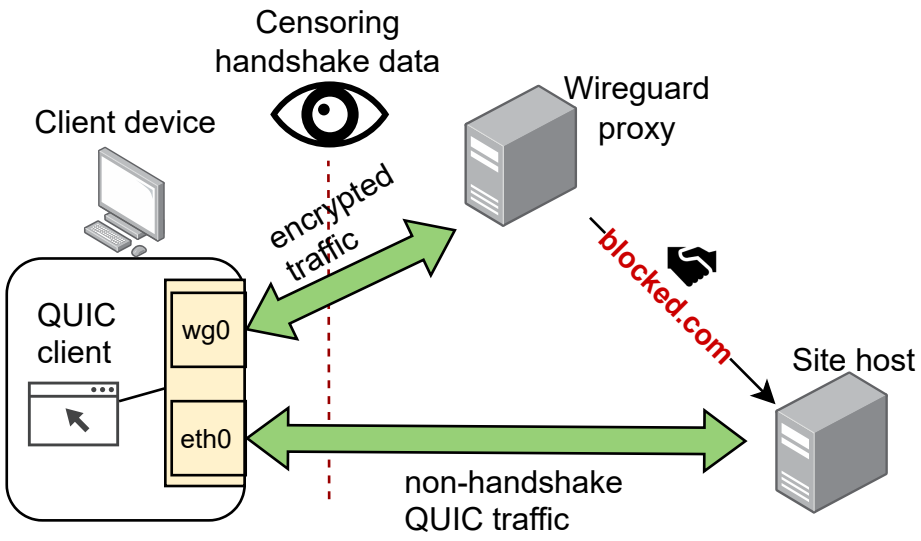


Figure 4.1: Diagram of setup for feasibility and performance evaluations. Though QUICstep works in theory with any encrypted tunnel, for our proof-of-concept we implement QUICstep using a Wireguard tunnel. We selectively send certain types of traffic (DNS traffic, QUIC-TLS handshake traffic) over the Wireguard tunnel and the rest over the native network.

Setup. The QUIC client is running on an Ubuntu 20.04 virtual machine located in the Greater New York area. The QUIC web server is located on an AWS EC2 instance in the Virginia region. The web server serves a 1MB or 10MB file composed of randomly generated content. We use Chromium QUICHE to implement the client and server due to its support for connection migration [35]. We choose WireGuard as the encrypted tunnel. A WireGuard proxy is running in AWS instances hosted at varied locations. The client's virtual machine has two network interfaces: one routes traffic on the direct unencrypted path via native QUIC, and one routes traffic via the WireGuard tunnel. We show the evaluation setup in Figure 4.1.

4.1 Feasibility evaluation against a proof-of-concept QUIC censor

We evaluate the ability of QUICstep to circumvent a censor who attempts to block client access to censored domains based on handshake packets. Ideally, we would evaluate QUICstep on vantage points located in networks using real deployed DPI middleboxes conducting SNI-based censorship of QUIC traffic. In the recent past, the most concrete observation of QUIC-SNI censorship has been in Russia where ISPs likely have the ability to decrypt QUIC handshake packets and match the SNI field to a blacklist [25]. However, in the time since the recent armed conflict between Ukraine and Russia, these original vantage points are no longer accessible, and it is currently difficult to ethically conduct censorship testing in Russia with the present geopolitical situation. Due to these difficulties, we implemented our own proof-of-concept (PoC), real-time QUIC censor for the purposes of testing the censorship resilience of QUICstep. We leave a more comprehensive QUIC censor implementation or an evaluation of QUICstep within real censored vantage points for future work.

The PoC censor attempts to block all QUIC connections by dropping QUIC handshake packets. We argue that if QUICstep is able to circumvent a coarse-grained censor, then it will be able to circumvent a finer-grained censor that performs SNI filtering on QUIC traffic. We implemented this censor in the form of a local proxy with an additional firewall rule that matches and drops all QUIC handshake packets while forwarding other network traffic as usual. The client's traffic is directed through this proxy before leaving the testbed network.

Our tests showed that the client failed to establish a QUIC session with the web

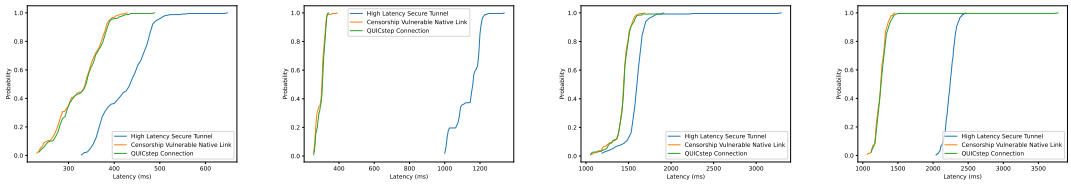
server using native QUIC connections, while QUICstep succeeded. Our solution, QUICstep, uses an encrypted tunnel to conceal the fact that a client is attempting to initiate a connection to a censored domain, which provides reliable resistance against SNI censorship.

4.2 Performance evaluation

Our goal in the performance evaluation is to demonstrate that the additional overhead of conducting a QUIC handshake over an encrypted tunnel and performing a single connection migration (generally, the cost of one network round-trip for path validation purposes) is amortized over the entire network request. We compare the latency of HTTP/3 GET requests when using a low-latency, censorship-vulnerable link (native QUIC), a high-latency, secure tunnel (WireGuard), and QUICstep to examine the additional latency incurred by our design.

We measure the latency of 250 HTTP/3 GET requests to fetch the 1 MB or 10 MB files, and calculate the cumulative distribution function (CDF) of the latency for each network connection. We also varied the location of the Wireguard proxy between two proxy node locations in Ohio and Oregon. We present the results in Figure 4.2.

Recall that QUICstep causes an increase in latency during a QUIC handshake due to two reasons: first, all handshake packets go through an encrypted tunnel via a VPN proxy, and second, path validation occurs after the connection migration. However, we expect this additional latency to be amortized because the rest of the QUIC session continues directly between the client and the server. The amortized effect is captured in Figure 4.2 — we can see that in all cases the CDF of latency for QUICstep closely follows that of a low



(a) Ohio, 1 MB **(b)** Oregon, 1 MB **(c)** Ohio, 10 MB **(d)** Oregon, 10 MB

Figure 4.2: The CDFs for the latency of 250 HTTP/3 GET requests for a 1 MB or 10 MB file using QUICstep, a censorship-vulnerable low latency link, and a secure yet high latency tunnel. The Wireguard proxy is varied between two locations, Ohio and Oregon. We can see QUICstep has minimal additional latency when compared to a native network connection while conducting the entire request through a proxy incurs much more latency.

latency link, while conversely, the CDF of a high latency tunnel is noticeably shifted to the right of both the QUICstep connection and low latency link CDFs. This shows that tunneling all network traffic through an encrypted tunnel incurs significantly more latency than QUICstep. The benefits of amortizing the additional latency in QUICstep are especially advantageous when the latency of the encrypted tunnel is higher, as seen in subfigures 4.2b and 4.2d where the location of the proxy is varied.

The latency advantage of QUICstep over a proxied connection is clearer as QUICstep allows the client to communicate directly with the server over native QUIC after completing the handshake, as opposed to sending all traffic through the proxy. In practice, it may not always be the case that the client communicates with a geographically close server since CDNs and load balancers will choose web servers close to the proxy node. However, by design, QUICstep will still perform faster than a fully proxied connection.

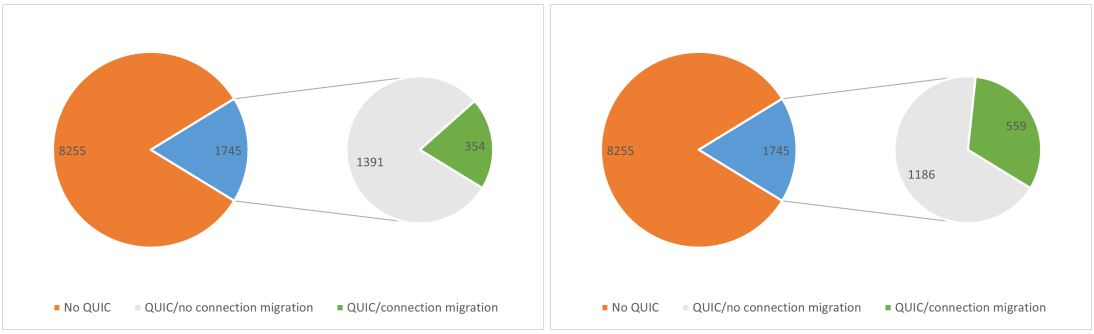
4.3 Measuring server support for QUIC connection migration

We design experiments to quantify the current state of support among popular web domains for QUIC connection migration. Our measurement of server-side support for QUIC

connection migration among popular web domains is to evaluate the current feasibility of deployment of QUICstep. We measured QUIC connection migration support from the top 10K domains from the Majestic Million [36] domain list. To identify server-side support for QUIC, we conduct an HTTP/1.1 GET request to a domain and scrape the `Alt-Svc` HTTP header from the response, denoting support for QUIC by the presence of the `h3` protocol identifier. To identify server-side support for connection migration among QUIC domains, we conduct an HTTP/3 GET request for that domain over QUICstep, denoting support for connection migration if the request succeeds without error. We used the QUIC libraries of two of the most popular web clients, Chromium QUICHE and Firefox Neqo, to conduct all HTTP/3 requests due to their support for connection migration [35, 37].

As of April 2023, we find that out of the top 10K domains from the Majestic Million, 1745 domains support QUIC. With the Chromium QUICHE client, we find that out of 1745 QUIC-supporting domains, 354 support connection migration. With the Firefox Neqo client, we find that out of 1745 QUIC-supporting domains, 559 support connection migration. Figure 4.3 illustrates these results.

Our results demonstrate that roughly one in six of the top 10K domains advertise QUIC support, and even fewer of those domains demonstrate working support for connection migration. Moreover, the discrepancy in the number of domains that support connection migration across different QUIC clients suggests interoperability issues between different QUIC implementations. Since QUICstep requires that both client and server implementations support QUIC connection migration, we expect this limitation to become less of an issue over time as we anticipate continued growth and adoption of QUIC across the web.



(a) Chromium QUICHE **(b)** Firefox Neqo

Figure 4.3: Pie charts illustrating the current state of server-side support for QUIC connection migration, among the top 10K domains of the Majestic Million domain name list as of April 2023, are shown. All HTTP/3 connections were conducted using the Chromium QUICHE and Firefox Neqo clients.

We acknowledge that growth of QUIC does not necessarily imply increased support for connection migration; however, we believe that the recent standardization of QUIC as an RFC will encourage developers of QUIC libraries to support connection migration as a feature and resolve cross-compatibility or interoperability issues between different QUIC implementations.

5

Discussion

QUICstep presents a promising direction for censorship circumvention in a QUIC-first world. In this section, we discuss additional barriers and challenges to large-scale deployment and efficacy of this technique, and propose a roadmap of future work to get there.

5.1 Deployment challenges

The success of QUICstep in the future depends not only on the ongoing deployment and popularity of QUIC, but various other factors that come into play when deploying the technique in practice.

5.1.1 QUICstep usage in web browsers

QUICstep is well-suited to handle a mixed combination of TCP and QUIC traffic that might be encountered in a web session on a major browser. However, since QUIC is a newer protocol that has yet to reach the entire Internet, browsers initiate QUIC connections in a more conservative manner. Specifically, browsers will establish a connection to a host over TCP and only switch to QUIC for subsequent requests if it received an `Alt-Svc` header indicating HTTP/3 support on that host [38]. If a censor blocks the initial TCP connection, then any potential subsequent QUIC connections will not take place. Nevertheless, there are non-practical methods to initiate a QUIC connection without a preceding TCP connection (i.e. *force* a QUIC connection) to a host within web browsers like Chrome and Firefox [39].

We additionally evaluate the ability of QUICstep to circumvent QUIC censorship within a browser. Using Chrome, we manually force QUIC connections to all resources on `google.com` hosted by a QUIC-supporting domain. We also modify our PoC QUIC censor to additionally drop TLS handshake packets to prevent Chrome from reloading resources of `google.com` over TCP upon QUIC connection failure. **We were able to verify Chrome failed to load `google.com` over a forced native QUIC connection, while QUICstep succeeded**.

QUICstep provides censorship resilience in browsers, but this may not be practically realized by end-users with default browser settings. We expect that as QUIC becomes more widely adopted, it may become preferable for browsers to force QUIC connections, either by default or by providing a practical end-user option. Beyond combining QUICstep with other TCP-based censorship circumvention techniques, future work will be needed to make QUICstep as a standalone tool more deployable for censorship circumvention in browsers.

5.1.2 Identifying QUIC connection migration support in the wild

During our measurements of QUIC connection migration support, we found that it was easy to identify server-side support for QUIC, but not for connection migration. We initially sought to evaluate whether a server could advertise support for connection migration based the QUIC `disable_active_migration` transport parameter, but based on our preliminary observation, it is not a reliable way to identify connection migration support. It only tells if a website does not support active connection migration (migration explicitly initiated by the client), but websites that do not support active connection migration may still support passive connective migration (migration triggered by a network change that the client is not aware of, such as NAT rebindings). Moreover, not all QUIC websites advertise this parameter.

Based on our findings, we found that the only reliable way to identify server-side support for connection migration was to attempt to perform connection migration with a server. Our measurements also suggest cross-compatibility or interoperability issues between different client and server implementations of connection migration. Monitoring of QUIC

connection migration support among commonly used QUIC libraries, evaluating interoperability between different connection migration implementations, and investigating and reliably identifying QUIC connection migration support among web domains is an area for future work.

5.1.3 Observability of QUICstep

Although QUICstep usage is theoretically indistinguishable from a regular connection migration, in practice connection migrations due to network changes may be quite rare. As such, this technique is less fingerprintable than other similar proposals [34], but certain network patterns may be correlated to the frequent usage of QUICstep.

To be specific, when using QUICstep, a native QUIC connection is divided into two sub-connections: handshake and non-handshake. To detect QUICstep, DPI may need to correlate network traffic related to these two sub-connections via timing attacks, which can require a large amount of state. Alternatively, a DPI middlebox could block all connections that look like they were migrated (where the associated connection ID was not observed in a handshake), but this would cause false positives for regular clients experiencing a regular network change. An additional complication is that QUIC connections can change connection IDs independent of connection migrations as well. Therefore, several intriguing research questions arise: How often does connection migration happen in practice? Can current real-world DPIs handle mobile network protocols? Can DPIs be designed to handle and control network-mobile clients efficiently? We leave these as future work.

5.1.4 Broad QUIC protocol blocking and monitoring

Researchers have observed that the QUIC protocol is blocked altogether in many regions [25]. We believe this is due to the fact that most deployed network middleboxes do not yet have the capability to decipher or manage QUIC connections, and rely on TCP-based metadata in order to enact network controls. In the context of QUIC, we can consider this a widespread downgrade attack as it forces clients and websites to utilize TCP rather than QUIC. Many DPI middleboxes are also in the midst of developing support for QUIC connection management [40, 41, 42]. As QUIC protocols like HTTP/3 become increasingly widespread and network management tooling adapts to the new QUIC paradigms, we identify a need for a non-downgradable protocol upgrade to QUIC. This way, the collateral damage of blocking all QUIC connections will grow to disincentivize networks from widespread protocol blocking in the future.

Further work investigating the current state of QUIC censorship and its methods in various countries is necessary and will also inform the development and deployment of censorship-resilient networking tools in the age of QUIC. Evaluating our design against real-world QUIC censors to evaluate its efficacy is also an area of future work.

5.2 Other QUICstep-like systems

QUICstep defers connection state management to the default QUIC stacks of the client and server for practicality and simplicity of deployment. If we generalize this system, a trusted provider of censorship-resilient communications (for instance, a trusted VPN or Tor) located outside the censored network could be responsible for maintaining a database

of active QUIC connection IDs and session keys to commonly censored websites. A modified QUIC client could then "migrate" connections from this store. In this hypothetical system, connection IDs can be communicated via any other channel whereas in QUICstep, the connection ID and session keys are communicated through the default handshake via an encrypted proxy.

We may also be able to leverage QUICstep-like techniques to subvert other network controls that are not implemented at the network layer. For instance, geoblocking can be implemented at the application-layer; it may be interesting to evaluate the usage case of QUICstep to evade geoblocking, particularly for access to high-bandwidth applications or resources.

Decoy routing applications. Researchers have proposed various uses of TLS session resumption as a covert channel for decoy routing [32, 33]. A similar application of connection migration, via sharing connection IDs and session keys, could be feasible for decoy routing. There has yet to be significant research into applying the features and benefits of QUIC to decoy routing applications.

6

Conclusion

We present QUICstep, a tool to circumvent SNI censorship in the QUIC setting. Our approach enables a client within a censored network to establish a QUIC session with a blocked server by conducting a QUIC-TLS handshake over an existing encrypted tunnel and performing connection migration to continue the rest of the session in plain sight of the censor. We implement a proof-of-concept of QUICstep using only a few routing and

firewall rules and provide open source access to our implementation. With our proof-of-concept implementation, we demonstrate that QUICstep provides censorship resilience with low performance overhead while being entirely agnostic to other client applications and upper-layer protocols.

References

[1] R. Sundara Raman, P. Shenoy, K. Kohls, and R. Ensafi, "Censored planet: An internet-wide, longitudinal censorship observatory," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 49–66. [Online]. Available: https://doi.org/10.1145/3372297.3417883

[2] K. Bock, G. Naval, K. Reese, and D. Levin, "Even censors have a backup: Examining china's double https censorship middleboxes," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*, 2021, pp. 1–7.

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.

[4] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 97–108.

[5] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, "Censorspoofer: asymmetric communication using ip spoofing for censorship-resistant web browsing," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 121–132.

[6] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "Stegotorus: a camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 109–120.

[7] P. Winter, T. Pulls, and J. Fuss, "Scramblesuit: A polymorphic network protocol to circumvent censorship," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 213–224.

[8] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, 2014, pp. 163–172.

[9] L. Dixon, T. Ristenpart, and T. Shrimpton, "Network traffic obfuscation and automated internet censorship," *IEEE Security & Privacy*, vol. 14, no. 6, pp. 43–53, 2016.

[10] "Tor project, obfsproxy3." [Online]. Available: https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt

[11] "Tor project, obfsproxy4." [Online]. Available: https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt

[12] "Racecar project, obfsproxy5." [Online]. Available: https://racecar.cs.georgetown.edu/software/

[13] "Tor project, meek." [Online]. Available: https://gitlab.torproject.org/legacy/trac/-/wikis/doc/meek

[14] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting." *Proc. Priv. Enhancing Technol.*, vol. 2015, no. 2, pp. 46–64, 2015.

[15] Z. Wang, Y. Cao, Z. Qian, C. Song, and S. V. Krishnamurthy, "Your state is not mine: A closer look at evading stateful internet censorship," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 114–127.

[16] D. Schinazi, F. Yang, and I. Schwett, "Chrome is deploying http/3 and ietf quic," Chromium Blog, Oct 2020. [Online]. Available: https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html

[17] D. Damjanovic, "Quic and http/3 support now in firefox nightly and beta," Mozilla Hacks, Apr 2021. [Online]. Available: https://hacks.mozilla.org/2021/04/quic-and-http-3-support-now-in-firefox-nightly-and-beta/

[18] J. Kehr, "What's quic?" Microsoft Tech Community, Aug 2021. [Online]. Available: https://techcommunity.microsoft.com/t5/networking-blog/what-s-quic/ba-p/2683367

[19] "Usage statistics of http/3 for websites," W3Techs, 2023. [Online]. Available: https://w3techs.com/technologies/details/ce-http3

[20] A. Havang, "Quic is quicly taking over!" Jan 2022. [Online]. Available: https://www.sandvine.com/blog/quic-is-quickly-taking-over

[21] J. Iyengar and M. Thomson, "Rfc 9000: Quic: A udp-based multiplexed and secure transport," *Omtermet Emgomeeromg Task Force*, 2021.

[22] M. Bishop, "Rfc 9114: Http/3," 2022.

[23] S. Kappanoğlu, "turkish ISPs use two methods for blocking access..." https://twitter.com/esesci/status/1630024112071491586. [Online]. Available: https://twitter.com/esesci/status/1630024112071491586

[24] K. Elmenhorst, B. Schütz, N. Aschenbruck, and S. Basso, "Web censorship measurements of http/3 over quic," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 276–282.

[25] K. Elmenhorst, "A quick look at quic censorship," Apr 2022. [Online]. Available: https://www.opentech.fund/news/a-quick-look-at-quic/

[26] Y. Govil, L. Wang, and J. Rexford, "Mimiq: Masking ips with migration in quic," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2020.

[27] M. Wang, A. Kulshrestha, L. Wang, and P. Mittal, "Leveraging strategic connection migration-powered traffic splitting for privacy," *arXiv preprint arXiv:2205.03326*, 2022.

[28] D. Fifield, "Turbo tunnel, a good way to design censorship circumvention protocols." in *FOCI@ USENIX Security Symposium*, 2020.

[29] "Tor project, snowflake." [Online]. Available: https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transports/snowflake/-/wikis/home

[30] E. Doerr, "Securing our approach to domain fronting within azure," Mar 2021. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2021/03/26/securing-our-approach-to-domain-fronting-within-azure/

[31] C. MacCarthaigh, "Enhanced domain protections for amazon cloudfront requests," Apr 2018. [Online]. Available: https://aws.amazon.com/blogs/security/enhanced-domain-protections-for-amazon-cloudfront-requests/

[32] V. Manfredi and P. 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)*, 2018.

[33] A. Devraj, L. Wang, and J. Rexford, "Redact: refraction networking from the data center," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 4, pp. 15–22, 2021.

[34] S. Satija and R. Chatterjee, "Blindtls: Circumventing tls-based https censorship," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*, 2021, pp. 43–49.

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

[36] Majestic, "The majestic million," 2023. [Online]. Available: https://majestic.com/reports/majestic-million

[37] Mozilla, "Neqo," https://github.com/mozilla/neqo/, 2022.

[38] R. Marx, "Http/3: Practical deployment options (part 3)," Sep 2021. [Online]. Available: https://www.smashingmagazine.com/2021/09/http3-practical-deployment-options-part3/

[39] K. Elmenhorst, "How to test http/3 in browsers." [Online]. Available: https://github.com/kelmenhorst/quic-censorship/blob/main/browsers.md

[40] L. Deri, "nDPI Encrypted Traffic Analysis," https://ripe80.ripe.net/presentations/35-nDPI_RIPE_052020.pdf, 2020.

[41] C. Yu, "GQUIC Protocol Analysis and Fingerprinting in Zeek," https://engineering.salesforce.com/gquic-protocol-analysis-and-fingerprinting-in-zeek-a4178855d75f/, 2022.

[42] S. S. FUNCTIONALITY, "From TCP to QUIC. Stingray SG Signatures," https://vasexperts.com/blog/functionality/from-tcp-to-quic/, 2022.