Extended Abstract: Using TURN Servers for Censorship Evasion

Afonso Vilalonga
Universidade NOVA de Lisboa & NOVA LINCS
Portugal
j.vilalonga@campus.fct.unl.pt

Kevin Gallagher
Universidade NOVA de Lisboa & NOVA LINCS
Portugal
k.gallagher@fct.unl.pt

Osman Yağan
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA
oyagan@andrew.cmu.edu

João S. Resende
Universidade do Porto
Portugal
jresende@fc.up.pt

Henrique Domingos
Universidade NOVA de Lisboa & NOVA LINCS
Portugal
hj@fct.unl.pt

Abstract

As global online censorship intensifies, developing new evasion systems is essential for improving access to uncensored content in highly censored regions. In addition to creating effective censorship evasion systems, the Bridge Distribution Problem also requires the development of systems that allow users to access the necessary information to connect to censorship evasion proxies, such as IP addresses, without censors blocking the communication channels. This extended abstract introduces a novel system that addresses both challenges by using TURN servers as relays to transmit user traffic to censorship evasion proxies or by allowing users to acquire the necessary information to connect to a censorship evasion proxy from brokers where proxy information is available. It also includes a hypothesis on how using TURN servers might help address these challenges.

Keywords

Censorship evasion, Rendezvous channels, TURN servers

1 Introduction & Background

Online censorship has significantly increased in recent years. According to Access Now, 2023 witnessed 283 internet shutdowns across 39 countries—a 29% rise compared to 2022 [22]. This growing trend is further exemplified by the large-scale implementation of censorship systems in various nations [2, 14, 16, 19], with China's Great Firewall (GFW) being the most prominent example [11, 15].

Various techniques have been developed to help users circumvent restrictions in censored regions. These methods range from standard proxies and VPNs, which are often easier to detect and block, to more advanced obfuscation techniques like traffic encapsulation [3, 5, 25] or fully encrypted protocols [4, 23]. However, different facets of the censorship evasion problem exist. On one hand, it is crucial to develop systems that can bypass censorship while remaining undetected. On the other, it is equally important to devise methods for distributing the necessary information, such as IP addresses, to users so they can connect to censorship evasion proxies. A key principle for this distribution is that the communication channel used must remain unblocked, even if the censor is fully aware of its existence (i.e., by using rendezvous channels [27]). This latter challenge represents one of the two subproblems within the broader "Bridge Distribution Problem", the first being how to develop rendezvous channels, and the second being how to ensure that the proxies do not fall into the hands of malicious users [26].

In this extended abstract, we present a novel censorship evasion system based on traffic encapsulation and traffic splitting, capable of serving as both a censorship evasion tool and a rendezvous channel. The core novelty lies in using TURN (Traversal Using Relays around NAT) [21] servers as censorship evasion proxies. Users route their traffic to N TURN servers ($1 \leq N \leq M; M \geq 1$), which relay the traffic to an aggregator server. This aggregator server reassembles the traffic and forwards it to the intended destination. Although traffic splitting is optional, we hypothesize that it could enhance the effectiveness of censorship evasion (see Section 3).

The rationale for using TURN servers is threefold. First, TURN is a protocol that allows a host behind a NAT, referred to as the "TURN Client", to request that another host, the "TURN Server", act as a relay for the client's traffic when a direct connection to its intended destination is not possible. This makes TURN particularly popular in WebRTC [1] and VoIP connections [6]. Estimating the number of applications that use TURN is challenging because many do not disclose this information, and some use closed-source code. However, the large number of TURN service providers [6, 10, 18, 29] suggests a high demand for these servers. Additionally, well-known applications such as Facebook Messenger have been shown to use TURN servers to relay media traffic during calls between users, making the TURN protocol a potential carrier protocol for a censorship evasion system. Second, TURN servers can be used out-of-the-box without any specific configuration by our system, allowing access to a larger pool of servers. These servers can range from user-deployed instances [7] to those owned by cloud providers and TURN service providers (e.g., Cloudflare [6], Metered Video [18], and others [10, 29]). Third, leveraging TURN servers from these providers may help address the previously described subproblem of the Bridge Distribution Problem by using them as rendezvous

Free and Open Communications on the Internet 2025(1), 13–15

channels. We hypothesize that these servers are less likely to be blocked in heavily censored regions due to their importance, enabling us to use them as relays to obtain information for clients seeking to connect to censorship evasion systems [26, 27]. We will elaborate on this further in Section 3.

The remainder of this extended abstract is organized as follows: In Section 2, we explore our planned system design, and in Section 3, we outline our hypothesis regarding how the system might help circumvent censorship and potentially provide a solution to the first subproblem of the Bridge Distribution Problem.

2 System Design

Our proposed system design is straightforward: we aim to enable a client application to route traffic through a TURN server, with an added capability for traffic splitting across multiple TURN servers. This system will be implemented and tested in two configurations: as a pluggable transport and as a standalone solution. Both configurations rely on three core components: the client-side software, the TURN proxy, and the bridge. Here, we use the term "bridge" to refer to both the Tor pluggable transport bridge and the server-side aggregator software in the standalone version. The client-side software differs slightly between the pluggable transport and standalone versions. In the pluggable transport design, the client-side software only accepts incoming connections from the Tor daemon. Conversely, the standalone version operates as a TCP server that allows regular applications to connect, though only SOCKS5 connections are permitted. The client-side software connects to one or more TURN proxies, enabling traffic splitting. These proxies then forward the traffic to a single bridge, where the split traffic is reassembled. Finally, the bridge directs the traffic either to the Tor network (in the pluggable transport version) or to the destination specified by the SOCKS5 connection's target address (in the standalone version).

For data transmission between the client-side software and the bridge, we adopt the turbo tunnel architecture [12], which provides a reliability layer and allows traffic to be split across multiple TURN proxies and reassembled at the bridge, provided that the bridge remains the same for that specific traffic stream. We plan to offer three options for data encapsulation within TURN messages: direct transmission of application data, encapsulation within DTLS datagrams via WebRTC data channels, and encapsulation within video frames in WebRTC media streams. The second and third options mimic the encapsulation strategies of known WebRTC-based censorship evasion systems [5, 25]. Additionally, we plan to test the system using various types of underlying transport protocols between the TURN client and the TURN server. Currently, the TURN protocol allows connections between the TURN client and the TURN server to be established using UDP, TCP, TLS-over-TCP, or DTLS-over-UDP [20, 21]. Finally, since the TURN server can inspect the content of the messages, we need to establish a symmetric key between the client and the bridge when WebRTC encapsulation is not used.

3 Hypothesis

3.1 Threat Model

We consider a censor capable of passively observing and classifying network traces, specifically between the client-side software and the TURN proxy. Our focus is passive attacks targeting the traffic exchanged between the client-side software and the TURN proxy, where the censor attempts to fingerprint our system to block it (i.e., Deep Packet Inspection (DPI) [24]). Additionally, we assume that the censor actively searches for censorship evasion system proxies and attempts to block them through any form of enumeration attacks [8, 9, 13]. This threat model aligns with descriptions of real-world censors and the attacks they carry out on deployed censorship evasion systems.

3.2 Bridge Distribution Problem

The motivation for using our system to address the first subproblem of the Bridge Distribution Problem stems from the availability of geographically distributed TURN servers that numerous TURN server providers provide for developers [6, 10, 18, 29]. While estimating the exact number of applications using TURN without empirical testing is challenging—a study we plan to conduct—the substantial number of TURN providers indicates an active demand for these services. Furthermore, TURN servers are essential for WebRTC applications to establish peer connections when direct peer-to-peer connections are blocked by NAT or firewall restrictions. We argue that WebRTC is a critical protocol stack for many heavily censored regions, as Snowflake, a WebRTC-based censorship evasion system, has remained unblocked in these areas. Consequently, to support such WebRTC applications, TURN servers must remain accessible to accommodate all potential users, which increases the likelihood that some TURN service providers will remain unblocked in these regions. Our approach, therefore, is to leverage these TURN server providers to acquire TURN servers and use them as relays, enabling users to access the necessary information to connect to a censorship evasion system through them.

3.3 Censorship Evasion

For censorship evasion, our goal is to understand the fingerprint of our system in comparison to other applications that use TURN servers. This knowledge will help us develop a system that blends with regular TURN server usage. We also aim to assess the prevalence of TURN usage in WebRTC-based applications, as well as the types of data these applications typically transmit over TURN servers. To achieve this, we will survey applications to identify those that use TURN servers, beginning with WebRTC-based applications and potentially expanding to other types. We hypothesize that traffic splitting and the proxy ephemerality nature of the system could mitigate active probing and IP blocking of proxies [17, 28] because (1) even if some proxies are blocked, the system continues functioning as long as the user uses multiple proxies, and (2) it allows the system to migrate to new proxies on the fly if existing ones are blocked. Active probing might also be countered, as our TURN servers are no different from regular TURN servers, which will produce the same responses to identical probing requests. Traffic splitting may also obscure specific characteristics that could otherwise be used to fingerprint the system. However, it may also introduce distinct fingerprints of its own, which we plan to evaluate.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive feedback. This work was supported by the FCT Ph.D. scholarship grant Ref. (PRT/BD/154787/2023), awarded by the CMU Portugal Affiliated Ph.D. program.

References

[1] H. Alvestrand. 2021. Overview: Real-Time Protocols for Browser-Based Applications. https://www.rfc-editor.org/rfc/rfc8825.html. Accessed: 2024-11-04.

[2] Ghadeer Awwad and Kentaro Toyama. 2024. Digital Repression in Palestine. In *CHI*. ACM. https://dl.acm.org/doi/pdf/10.1145/3613904.3642422

[3] Diogo Barradas, Nuno Santos, Luís Rodrigues, and Vítor Nunes. 2020. Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC. In *Computer and Communications Security*. ACM. https://www.gsd.inesc-id.pt/~nsantos/papers/barradas_ccs20.pdf

[4] Ben Lynne (brl). n.d.. Obfuscated OpenSSH. https://github.com/brl/obfuscatedopenssh. Accessed: 2024-11-01.

[5] Cecylia Bocovich, Arlo Breault, David Fifield, Serene, and Xiaokang Wang. 2024. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In *USENIX Security Symposium*. USENIX. https://www.usenix.org/system/files/sec24fall-prepub-1998-bocovich.pdf

[6] Cloudflare. n.d.. TURN Service. https://developers.cloudflare.com/calls/turn/ Accessed: 2024-11-04.

[7] Coturn. n.d.. Coturn. https://github.com/coturn/coturn Accessed: 2024-11-04.

[8] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *8th USENIX Workshop on Free and Open Communications on the Internet (FOCI 18)*. USENIX Association. https://www.usenix.org/conference/foci18/presentation/dunna

[9] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *Proceedings of the 2015 Internet Measurement Conference*. Association for Computing Machinery. https://doi.org/10.1145/2815675.2815690

[10] ExpressTURN. n.d.. ExpressTURN. https://www.expressturn.com/ Accessed: 2024-11-04.

[11] Yuzhou Feng, Ruyu Zhai, Radu Sion, and Bogdan Carbunar. 2023. A Study of China's Censorship and Its Evasion Through the Lens of Online Gaming. In *USENIX Security Symposium*. USENIX. https://www.usenix.org/system/files/usenixsecurity23-feng.pdf

[12] David Fifield. 2020. Turbo Tunnel, a good way to design censorship circumvention protocols. In *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association. https://www.usenix.org/conference/foci20/presentation/fifield

[13] Sergey Frolov, Jack Wampler, and Eric Wustrow. 2020. Detecting Probe-resistant Proxies. In *Network and Distributed System Security*. The Internet Society. https://www.ndss-symposium.org/wp-content/uploads/2020/02/23087.pdf

[14] Devashish Gosain, Kartikey Singh, Rishi Sharma, Jithin S, and Sambuddho Chakravaty. 2024. Out in the Open: On the Implementation of Mobile App Filtering in India. In *Passive and Active Measurement Conference*. Springer. https://pam2024.cs.northwestern.edu/pdfs/paper-58.pdf

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

[16] Divyank Katira, Gurshabad Grover, Kushagra Singh, and Varun Bansal. 2023. CensorWatch: On the Implementation of Online Censorship in India. In *Free and Open Communications on the Internet*. https://www.petsymposium.org/foci/2023/foci-2023-0006.pdf

[17] Anna Harbluk Lorimer, Rob Jansen, and Nick Feamster. 2024. Extended Abstract: Traffic Splitting for Pluggable Transports. In *Free and Open Communications on the Internet*. https://www.petsymposium.org/foci/2024/foci-2024-0004.pdf

[18] metered. n.d.. Open Relay: Free WebRTC TURN Server. https://developers.cloudflare.com/calls/turn/ Accessed: 2024-11-04.

[19] Sadia Nourin, Van Tran, Xi Jiang, Kevin Bock, Nick Feamster, Nguyen Phong Hoang, and Dave Levin. 2023. Measuring and Evading Turkmenistan's Internet Censorship. In *The International World Wide Web Conference*. ACM. https://dl.acm.org/doi/abs/10.1145/3543507.3583189

[20] M. Petit-Huguenin and G. Salgueiro. 2014. Datagram Transport Layer Security (DTLS) as Transport for Session Traversal Utilities for NAT (STUN). https://datatracker.ietf.org/doc/html/rfc7350. Accessed: 2024-11-04.

[21] R. Mahy and P. Matthews and J. Rosenberg. 2010. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). https://datatracker.ietf.org/doc/html/rfc5766 Accessed: 2024-11-04.

[22] Zach Rosson, Felicia, and Carolyn Tackett. 2024. The most violent year: internet shutdowns in 2023. https://www.accessnow.org/internet-shutdowns-2023/. Accessed: 2024-11-04.

[23] The Tor Project. n.d.. obfs4 (The Obfuscator). https://gitweb.torproject.org/pluggabletransports/obfs4.git/tree/doc/obfs4-spec.txt. Accessed: 2024-11-02.

[24] Michael Carl Tschantz, Sadia Afroz, Name withheld on request, and Vern Paxson. 2016. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *2016 IEEE Symposium on Security and Privacy (SP)*.

[25] Afonso Vilalonga, Joao S. Resende, and Henrique Domingos. 2023. TorKameleon: Improving Tor's Censorship Resistance with K-anonymization and Media-based Covert Channels . In *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE Computer Society. https://doi.ieeecomputersociety.org/10.1109/TrustCom60117.2023.00203

[26] Afonso Vilalonga, João S. Resende, and Henrique Domingos. 2024. Looking at the Clouds: Leveraging Pub/Sub Cloud Services for Censorship-Resistant Rendezvous Channels. In *Free and Open Communications on the Internet*. https://www.petsymposium.org/foci/2024/foci-2024-0010.pdf

[27] Paul Vines, Samuel McKay, Jesse Jenter, and Suresh Krishnaswamy. 2024. Communication Breakdown: Modularizing Application Tunneling for Signaling Around Censorship. *Privacy Enhancing Technologies* (2024). https://petsymposium.org/popets/2024/popets-2024-0027.pdf

[28] Ryan Wails, George Arnold Sullivan, Micah Sherr, and Rob Jansen. 2024. On Precisely Detecting Censorship Circumvention in Real-World Networks. In *Network and Distributed System Security*. The Internet Society. https://www.robgjansen.com/publications/precisedetect-ndss2024.pdf

[29] xirsys. n.d.. xirsys. https://xirsys.com/ Accessed: 2024-11-04.