Surfing the Web Quicker Than QUIC via a Shared Address Validation

Erik Sy
Department of Informatics
University of Hamburg
Hamburg, Germany
sy@informatik.uni-hamburg.de

***Abstract*—QUIC is a performance-optimized secure transport protocol and a building block of the upcoming HTTP/3 standard. To protect against denial-of-service attacks, QUIC servers need to validate the IP addresses claimed by their clients. So far, the QUIC protocol conducts address validation for each hostname separately using validation tokens. In this work, we review this practice and introduce a new QUIC transport parameter to allow a shared address validation across hostnames. This parameter indicates to the client, that an issued validation token can be used to abbreviate the address validation when connecting to specific other hostnames. Based on trust-relations between real-world hostnames we evaluate the performance benefits of our proposal. Our results suggest that a shared address validation saves a round-trip time on almost 60% of the required handshakes to different hosts during the first loading of an average website. Assuming a typical transatlantic connection with a round-trip time of 90 ms, we find that deploying our proposal reduces the delay overhead to establish all required connections for an average website by 142.2 ms.**

***Index Terms*—QUIC transport protocol, address validation token, performance enhancement**

I. Introduction

The world wide web is closely tied to the HTTP network protocol. The upcoming version HTTP/3 will replace the traditional TLS over TCP stack with the new QUIC protocol [3]. Thus, it is likely that QUIC will be widely adopted on the web within the forthcoming years. Further use cases include, but are not limited to the Domain Name System (DNS) [6].

A goal of the QUIC protocol is to reduce the delay required to establish secure connections. Faster connection establishments contribute to shorter page loading times, that correlate with an improved experience for web users [14]. To achieve this goal QUIC provides the feature of a zero round-trip time handshake that conducts the transport and the cryptographic connection establishment at the same time. However, this feature is only applicable when reconnecting to a QUIC server as it requires cached state of a prior connection. The first connection to a hostname requires up to two additional round-trips. One additional round-trip is caused by the cryptographic connection establishment and the other by the transport handshake.

In this work, we are addressing the performance limitations caused by this additional round-trip of the transport handshake that is used to validate the client's source address. To illustrate the shortcomings of QUIC's current design, we assume that a client connects sequentially to the hostnames *example.com* and *www.example.com*. Furthermore, we assume both of these hostnames are operated by the same entity and a strict address validation is deployed by the responding QUIC servers. We find that the default QUIC behavior causes per connection one additional round-trip for the validation of the client's address. As a result of this process, the client's address has been validated at the expense of two additional round-trips by the same entity via different hostnames.

We propose a performance-optimized approach that allows reusing address validation tokens across hostnames that have a trust-relation to each other. The client connects to the first hostname with an additional round-trip to validate the client's address and receives a validation token. The second handshake to the other hostname uses the validation token received during the first connection to pass the address validation without an additional round-trip. This approach saves in total a round-trip time compared to the default QUIC behavior.

In summary, this paper makes the following contributions:

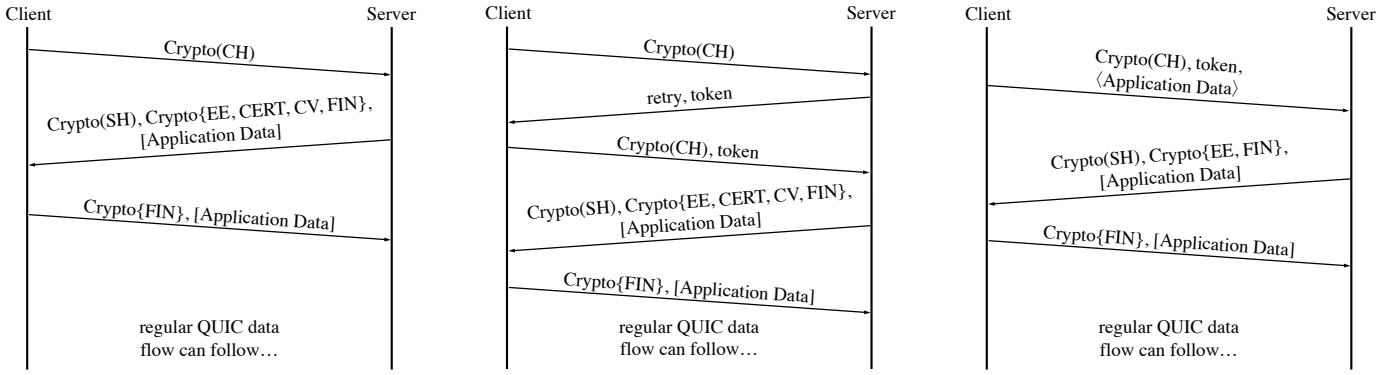
- We propose a new transport parameter for the QUIC protocol that enables a shared address validation across different hostnames.
- We demonstrate the performance gains yielded by our proposal for loading popular websites. Our results indicate that deploying the introduced transport parameter saves a round-trip time on almost 60% of the required connection establishments to different hosts during the first loading of an average website.

This proposal was presented to the IETF's QUIC working group, which aims to integrate the proposed shared address validation in the QUIC protocol [13].

The remainder of this paper is structured as follows: Section II describes the performance problem of the QUIC transport protocol that we aim to solve. Section III summarizes the proposed shared address validation and evaluation results are presented in Section IV. Related work is reviewed in Section V, and Section VI concludes the paper.

II. Problem Statement

In this section, we review the connection establishment of the IETF QUIC protocol [7]. Subsequently, we describe the problem that we aim to solve and our threat model.



a) Initial handshake b) Initial handshake with retry c) 0-RTT handshake

Fig. 1. Handshakes in IETF QUIC protocol, where the brackets indicate different levels of encryption. Round brackets indicate no encryption and curved brackets denote encryption based on the handshake traffic secret. Square brackets signal encryption using the application traffic secret and angle brackets indicate the use of the early traffic secret for encryption.

A. Connection Establishment with IETF QUIC

Google developed and deployed the initial design of the QUIC protocol [8], which we will refer to as Google QUIC (gQUIC). In 2016, the Internet Engineering Task Force (IETF) started to standardize QUIC [7], which is still a work in progress. The IETF's draft of QUIC differs significantly from gQUIC, e. g., it uses TLS 1.3 [9] to conduct the cryptographic handshake. However, both QUIC variants aim to improve the performance of HTTPS traffic by conducting the cryptographic and the transport handshake concurrently, while the widespread approach using TLS over TCP conducts these handshakes sequentially. Figure 1 shows a schematic of QUIC's connection establishment.

a) Initial Handshake with and without Retry: As shown in Figure 1 a), the client starts with the ClientHello (CH) message containing lists of supported cipher suites, protocol versions, TLS extensions, and public keys suitable for key exchange. All messages that are part of the cryptographic connection establishment follow the TLS 1.3 [9] protocol and are emphasized with *Crypto* in Figure 1. The server responds with its unencrypted ServerHello message, which is indicated by round brackets in Figure 1. Subsequently, the server computes the handshake traffic secret and sends the Encrypted Extensions (EE), the server's certificate (CERT), the Certificate Verify (CV), and the handshake finished (FIN) messages encrypted with this secret, shown as curved brackets in Figure 1. In the CV message, the server provides a fresh proof for its ownership of the certificate's private key. Whereas FIN messages signal a successful handshake and contain hashes of the exchanged handshake messages to verify that both peers observed the same messages. The server can now calculate the application traffic secret and may send encrypted application data. This encryption type is indicated with square brackets in Figure 1.

Upon receiving the ServerHello message, the client computes the handshake traffic secret. This enables the client to decrypt the received EE, CERT, CV, and FIN messages. The client can now authenticate the server's identity based on the provided CERT and CV messages. Then, the client validates the hashes contained in the server's FIN message and calculates the application traffic secret. Finally, the client responds with its own FIN message and may send encrypted application data to the server.

After a connection is successfully established, the server can provide the client with TLS resumption tickets and address validation tokens that can be used on subsequent connections. Resumption tickets allow resuming previous connections via abbreviated handshakes that decrease the delay overhead and save expensive cryptographic computations during connection establishment. A validation token is an encrypted and authenticated data block which is opaque to the client. It usually contains the client's visible IP address as seen by the server. If the client provides such a token during a subsequent connection establishment, this allows the server to compare the client's claimed IP address with the previously observed clients' IP address in the token.

Depending on its configuration, the server may per default or dynamically decide to strictly validate the client's IP address before proceeding with the cryptographic handshake. Figure 1 b) shows an initial handshake that validates the client's source address with an additional round-trip. Here, the server sends a retry message and an address validation token to the IP address claimed with the client's first message. To proof the ownership of the claimed IP address, the client is required to provide the received token along with its ClientHello message. Upon receiving the token from the client, the server validates it, which involves a comparison of the IP address stored in the token with the one claimed by the client. If the token is valid, the client-server pair will proceed with a standard initial connection establishment starting from the ServerHello message (see Figure 1 a).

b) 0-RTT Connection Establishment: Furthermore, the QUIC protocol provides zero round-trip time (0-RTT) connection establishments as shown in Figure 1 c). Here, the client can send data encrypted with the early traffic secret without waiting for a response from the server using TLS resumption. Thus, the client requires a previously retrieved resumption ticket and a pre-shared key (PSK) to encrypt these early data and to signal the used PSK to the server. Optionally,

the client can provide an address validation token as shown in Figure 1 c) to anticipate a server's retry request.

Upon receiving these messages, the server starts to validate the client's token and IP address. In case of a positive validation result, the server begins with its own cryptographic operations. To derive the early traffic secret, the server uses also information provided in the *pre_shared_key* extension of the ClientHello message. Assuming, that the server can successfully decrypt the provided early data, it will signal this with the *pre_shared_key* extension in the ServerHello message. After sending the ServerHello message, the server will derive the handshake traffic secret and use it for encrypted transmission of the EE and FIN messages. Note, that this handshake does not require the server to provide a CERT and CV message to authenticate its claimed identity. Instead, the peers authenticate each other by successfully resuming the previous TLS session using the pre-shared key. This abbreviated authentication during resumed connections significantly decreases the delay and saves expensive cryptographic operations. Subsequently, the server can derive its application data secret and respond with encrypted application data to the client's early data.

Upon receiving the server's messages, the client will derive the handshake traffic secret and decrypt the server's EE and FIN message. Then, the client reviews the validation data contained in the server's FIN message to validate, that the exchanged messages have not been tampered with during transit. If the client did not determine a modification of the received data, it will provide the server its own FIN message. Finally, the client can derive its application traffic secret and the regular data flow follows.

Note, that application data encrypted with the early traffic secret does not provide forward-secrecy and might be replayed in other connections. For a comprehensive discussion on the weaker security guarantees of early data, we refer readers to RFC 8446 [9].

B. Performance Limitation of Address Validation

A QUIC server can validate the IP address claimed by a client by sending it a retry message with a token. Only, if the client returns this issued token, the server proceeds with the cryptographic handshake. This practice protects the server from spending expensive cryptographic operations on malicious handshake attempts claiming an illegitimate IP address. As a drawback, this IP spoofing defense increases the delay overhead of the connection establishment by a round-trip time.

To avoid this additional delay during the establishment of a new connection, a client can present a token from a previous connection to the same hostname along with the ClientHello message. If the server accepts this token as valid for the IP address claimed by the client, then it will directly proceed with the cryptographic connection establishment. This practice requires the client to retrieve a token in a previous QUIC connection to the same hostname.

QUIC does not consider using a retrieved address validation token to connect to new hostnames not matching the connection that has been used to retrieve the token. As a result, connections to every fresh hostname cannot present an address validation token leading to the described additional delay overhead if the server enforces address validation. This is a performance limitation if trust-relations between different hostnames are available. To illustrate this limitation, we assume a trust-relation between the hostnames *example.com* and *www.example.com*, which mutually accept their issued address validation tokens. Assuming the same network latency to both hostnames, a client establishing fresh connections to both hostnames experiences an additional delay overhead of twice the round-trip time due to address validation. However, if the client sequentially connects to these hostnames and reuses a token obtained in the first connection to establish the latter connection, this bisects the additional delay overhead to a single round-trip time.

To put this into perspective, a typical latency in North America is < 45 ms and < 90 ms for transatlantic connections [15]. However, several regions in the world exist which suffer from high network delays, often exceeding 300 ms [5]. In total, this example illustrates that using address validation tokens across different hostnames can provide significant performance gains, especially for connections experiencing high latencies.

C. Threat Model

To clarify the security aspects of the described problem, we define our threat model in the following.

The considered adversary is able to spoof the source address of its packets. However, we assume the address validation tokens to be cryptographically secure, thus attackers cannot generate valid tokens. In total, our adversary affects the security objective of availability. By spoofing the IP address of a victim's endpoint, the attacker causes the QUIC server to send its response to the victim, which is also known as a reflection attack. Moreover, the attacker can also directly attempt to exhaust the resources of the server by requesting connections from various spoofed IP addresses.

III. Shared Address Validation across Hostnames

This section describes our approach of a shared address validation across hostnames for the QUIC protocol. For that, we introduce the new transport parameter *validation_group* that aims to enable a shared address validation across hostnames.

Deviating from the standard initial handshake, the server unilaterally includes the *validation_group* value in its list of transport parameters. In detail, the *validation_group* presents a one-bit value, which is set to one if this feature is supported and zero otherwise.

The client finds the declared *validation_group* value in the server's EE message. If this value is set to zero, then the client reasons that the server does not support this feature and proceeds with its default behavior.

In case the *validation_group* value is set to one, the client concludes that the server supports a shared address validation across the hostnames, for which the presented TLS certificate is valid. We define a validation group to be a list of hostnames

for which a single TLS certificate is valid and that mutually support address validation using tokens issued by any member of the same group. To support a shared address validation, the client associates received tokens during that connection with the validation group at hand.

To open a new connection to any member of a validation group, the client can use cached tokens associated with the same group. Tokens received during such a new connection must be associated with the same validation group.

The QUIC server does not provide a TLS certificate during a resumed connection establishment. In this case, the tokens received during the connection should be associated with the TLS certificate of the original connection, for which the server's identity was authenticated using a TLS certificate.

To mitigate tracking via address validation tokens [11], a client-side expiration mechanism is required for them. The lifetime of these tokens presents a performance versus privacy trade-off, where shorter lifetimes lead to better privacy protection. Based on an analysis of characteristic Internet traffic, the authors of [10] recommend a lifetime of ten minutes to address this trade-off in web browsers.

To support a shared address validation across hostnames, the cryptographic secret used to generate and decrypt the tokens needs to be shared across the servers serving these hostnames. Note, that address validation tokens are designed for single-use only. If a member of a validation group is not properly configured and fails to validate a legitimate token, then this process uses up a cached token of the respective validation group. In the worst case, this behavior can lead to reduced performance compared to the status quo. For example, if the client needs to respond to a retry message of a server because it used up its cached tokens on not properly configured servers.

IV. Evaluation

In this section, we evaluate the performance benefit of our proposal for web browsing. For this purpose, we analyze the shared address validation across hostnames for the Alexa Top 1K Sites [1] using an analytical model. We start by describing our assumptions for this evaluation. Subsequently, we introduce the analyzed data set and summarize our results.

A. Assumptions

To analyze the loading behavior of popular websites, we assume that each established connection enforces a strict address validation. This assumption describes the current practice of TCP Fast Open [4] and gQUIC [8]. However, IETF QUIC provides an operation mode with a relaxed address validation known as stateless retry. Nonetheless, we believe the strict address validation to become the most popular operation mode of QUIC servers. As the draft on IETF QUIC is still in an early stage, we are not aware of online services deploying IETF QUIC at a production level. Thus, we cannot determine the configuration of real-world deployments of IETF QUIC to assess the accuracy of this assumption at the moment.

Moreover, this evaluation assumes that the investigated websites deploy the QUIC protocol to support their HTTPS connection establishment. To justify this assumption, we point out that numerous browser vendors and content delivery networks contribute to the draft of the QUIC protocol [7] and intend to deploy this protocol in their products. Furthermore, the upcoming HTTP/3 uses the QUIC protocol [3] by default, thus it will be deployed to serve websites. In total, it seems likely that the QUIC protocol will be widely deployed on the Internet within the forthcoming years.

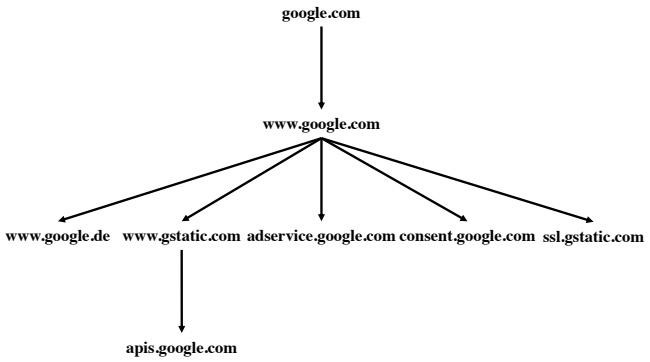


Fig. 2. Domain tree of the website google.com.

In addition, to evaluate our proposal for popular websites we require information on real-world hostnames that trust each other with regard to the availability of their servers. To approximate this information, we assume that hostnames trusting each other with respect to the confidentiality of their communications are also willing to have a trust-relation concerning the availability of their servers. We define that a trust-relation with respect to confidentiality exists, if hostnames share a secret cryptographic state with each other such as the private key of their TLS certificate or they enable the resumption of TLS sessions across their hostnames. From our perspective, it seems reasonable that such closely cooperating hostnames, which are often operated by the same entity, are willing to trust each other in terms of the validation of their clients' IP addresses.

B. Data Set

This paper uses a data set on trust-relations of the Alexa Top 1K Sites, that is described and evaluated in [12]. The data set aggregates a scan of the Alexa Top 1K Sites [1] performed on the 8th of November 2018. In total, the scan successfully retrieved the domain trees of 839 websites. A domain tree describes an overview of the sequence of established connections to different hostnames during the retrieval of a website (see Figure 2). Furthermore, the data set collected real-world TLS trust-relations between the different hostnames within each domain tree. For that, the data set defines a TLS trust-relation between two hostnames if they either share the same TLS certificate or enable the resumption of TLS sessions between their hostnames.

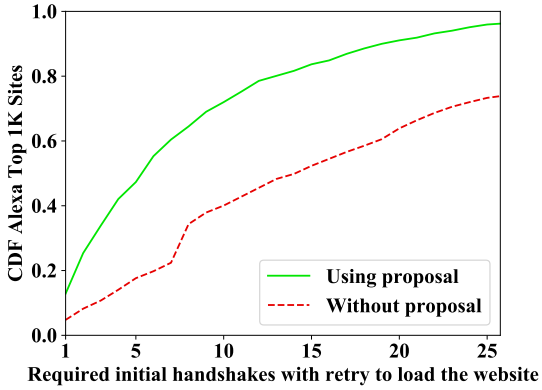


Fig. 3. A cumulative distribution function (CDF) of Alexa Top 1K Sites is shown in dependence on the number of required initial handshakes with retry to retrieve the website. The green line marks the values considering the introduced shared address validation, while the red, dashed plot notes the current default. Note, that this plot is cut off at 25 handshakes.

C. Results

In this section, we analyze the performance benefits of our proposal for the first loading of popular websites. First, we investigate the number of connections necessary for the retrieval of an average website, that can each save a round-trip time during the connection establishment by deploying shared address validation. Subsequently, we observe that most websites require a client to sequentially establish connections to different hosts. Based on this practice, we analyze our proposal by measuring the aggregated delay overhead for establishing all QUIC connections necessary for the retrieval of the respective website successively. This evaluation of successive retrievals allows approximating the benefit of our proposal for the loading time of a website.

a) Share of Abbreviated Handshakes: The analyzed data set [12] indicates, that TLS trust-relations are a common practice on the web. In total, the average Alexa Top 1K Site requires the client to establish 20.24 encrypted connections to different hostnames. Figure 3 plots the cumulative distribution function of the Alexa Top 1K Sites over the number of required initial handshakes with retry for their retrieval. The red, dashed plot represents the status quo and indicates that 95.2% of these websites require more than a single secure connection. Note, that 73.3% of the investigated sites can be loaded with at most 25 initial handshakes with retry. Thus, we cut off this plot after 25 of these handshakes for reasons of clarity.

The green plot represents the results of our analytical model using a shared address validation in case of a TLS trust-relation between the corresponding hostnames. If such a trust-relation exists, the client will reuse a received address validation token across this group of hostnames that trust each other. Thus, our model converts an initial handshake with retry (status quo) to an initial handshake without retry, if the client previously connected to any member of the same validation group to retrieve corresponding address validation tokens. Note, that our model investigates each website visit separately. Each converted handshake reduces the delay overhead of the specific connection establishment by a round-trip. Figure 3 shows that the use of a shared address validation shifts the distribution of the Alexa Top 1K Sites towards less required initial handshakes with retry. For example, the share of sites requiring a single initial handshake with retry increased from 4.8% to 12.9% by using our proposal. Furthermore, we find that 42.1% of the investigated websites can be retrieved with less than five initial handshakes with retry by using the introduced proposal compared to 14.1% without deploying the proposal. In total, we find that 96.0% of the Alexa Top 1K Sites can be retrieved with at most 25 initial handshakes with retry by deploying a shared address validation.

Our results for the average Alexa Top 1K Site suggest, that a shared address validation reduces the number of initial handshakes with retry from 20.24 to 8.35 for the first visit of an average website. Thus, 11.89 initial QUIC connections can be converted to use an address validation token received from a member of the same validation group. In total, the proposed practice saves for 58.75% of the established QUIC connections a round-trip time. This yields a reduction of 11.89 round-trips during the establishment of the required connections.

b) Delay Overhead: The studied data set [12] provides insights into the sequence of established connections and which retrieved resources triggered the establishment of additional connections. Figure 2 provides the domain tree of *google.com* that marks initiated connections to different hostnames with arrows. We observe that the longest path within the domain tree requires four sequential connection establishments via the hostnames *google.com*, *www.google.com*, *www.gstatic.com*, and *apis.google.com*. Thus, the website retrieval of *google.com* is impacted by about four times the delay overhead of a single connection establishment.

Figure 4 plots the cumulative distribution of Alexa Top 1K Sites over the length of their longest paths of initial handshakes with retry. The red, dashed plot indicates the current status quo. We observe, that the Alexa Top 1K Sites require no more than eight sequential connections for their retrieval. The single most popular configuration requires four sequential connections and is used by 33.1% of the Alexa Top 1K Sites. In total, we find that 63.0% of the Alexa Top 1K Sites can be retrieved with four or less sequential connections.

The plot marked with a green line provides the results for deploying the proposed shared address validation. In our simulation, we convert if applicable an initial handshake with retry to an initial handshake, where the client presents an address validation token obtained from another member of the same validation group. Each of these converted handshakes saves a round-trip time during the corresponding connection establishment. Note, that our evaluation repeats the computation of the longest paths of initial handshakes with retry after simulating the shared address validation. Thus, the longest paths with and without using the introduced shared address validation can deviate from each other.

We find, that the deployment of our proposal leads to a significant reduction of necessary initial handshakes with retry

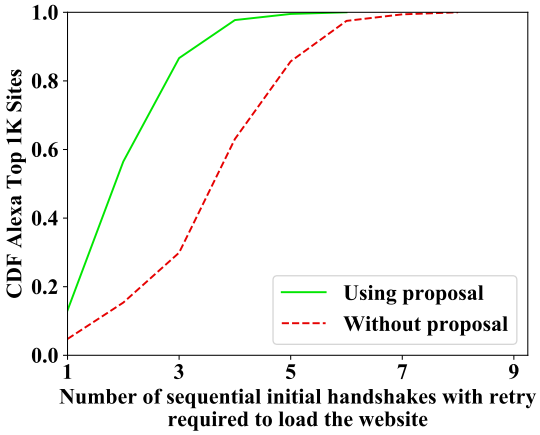


Fig. 4. This plot shows a CDF of the Alexa Top 1K Sites over the number of required sequential initial QUIC connections with retry to load the respective website. The green line represents the values considering the introduced shared address validation, while the red, dashed plot marks the current default.

for the Alexa Top 1K Sites. For example, using a shared address validation reduces the share of websites requiring more than five sequential connections from 37.0% to 2.3%. Furthermore, the number of websites requiring less than three connections increased from 15.4% to 56.5% when using our proposal.

Our results indicate, that the average Alexa Top 1K Site requires 4.04 sequential initial handshakes with retry. The proposed shared address validation reduces this value to 2.46. In total, our proposal saves 1.58 times the round-trip time until the last connection required for loading an average website is established. This presents a reduction of the longest path of initial handshakes with retry by 39.1% for the average website. Assuming a transatlantic connection with a round-trip time of 90 ms [15], our proposal reduces the delay overhead for establishing all required QUIC connections by 142.2 ms.

V. Related Work

To the best of our knowledge, we are the first to investigate the benefit of a shared address validation across hostnames for transport handshakes. Prior work includes the draft of the QUIC transport protocol, as it is designed to support a shared address validation across servers serving the same hostname.

Related work includes [12], that article investigates trust-relations within the domain trees of the Alexa Top 1K Sites and proposes TLS session resumption across hostnames to increase the number of resumed TLS handshakes during web browsing. However, this work focuses on QUIC's transport handshake and the task of validating the client's source address which are unrelated to TLS session resumption mechanisms.

Moreover, HTTP version 2 (HTTP/2) [2] can be considered as related work as it allows reusing TLS connections across different hostnames if these hostnames can be served from the same server address. In detail, HTTP/2 aims to reduce the number of established connections to yield performance gains. However, our proposal reduces the delay of the connection establishment itself. Thus, our proposal can be applied if HTTP/2 connection reuse is not feasible. For example, if two trusting hostnames are served via different IP addresses or a connection has already timed out.

VI. Conclusions

This work proposes a new transport parameter for the QUIC protocol to support clients to use address validation tokens across hostnames. Our evaluation demonstrates, that such a shared address validation significantly reduces the delay overhead of QUIC's connection establishment on the real-world web.

References

[1] Alexa Internet Inc. (2019) Alexa Top 1,000,000 Sites. [Online]. Available: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

[2] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7540.txt

[3] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-http-19, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-19

[4] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain, "TCP Fast Open," RFC 7413, Dec. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7413.txt

[5] A. Formoso, J. Chavula, A. Phokeer, A. Sathiaseelan, and G. Tyson, "Deep Diving into Africa's Inter-Country Latencies," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2231–2239.

[6] C. Huitema, M. Shore, A. Mankin, S. Dickinson, and J. Iyengar, "Specification of DNS over Dedicated QUIC Connections," Internet Engineering Task Force, Internet-Draft draft-huitema-quic-dnsoquic-06, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-huitema-quic-dnsoquic-06

[7] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-transport-19, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-19

[8] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The QUIC transport protocol: Design and Internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.

[9] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8446.txt

[10] E. Sy, C. Burkert, H. Federrath, and M. Fischer, "Tracking users across the web via tls session resumption," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC '18. New York, NY, USA: ACM, 2018, pp. 289–299. [Online]. Available: http://doi.acm.org/10.1145/3274694.3274708

[11] ——, "A QUIC Look at Web Tracking," *Proceedings on Privacy Enhancing Technologies*, vol. 3, 2019.

[12] E. Sy, M. Moennich, T. Mueller, H. Federrath, and M. Fischer, "Enhanced Performance for the encrypted Web through TLS Resumption across Hostnames," *arXiv preprint arXiv:1902.02531*, 2019.

[13] Sy, Erik. (2019) Saving a round-trip time in the initial handshakes with retry. [Online]. Available: https://github.com/quicwg/base-drafts/issues/2552

[14] M. Varvello, J. Blackburn, D. Naylor, and K. Papagiannaki, "Eyeorg: A platform for crowdsourcing web quality of experience measurements," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 399–412.

[15] Verizon. (2019) IP Latency Statistics. [Online]. Available: https://enterprise.verizon.com/terms/latency/