IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 9, 2020, accepted July 20, 2020, date of publication July 31, 2020, date of current version August 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013524

QSOCKS: 0-RTT Proxification Design of SOCKS Protocol for QUIC

MADHAN RAJ KANAGARATHINAM[1], (Senior Member, IEEE), SUKHDEEP SINGH[1], (Member, IEEE), SUJITH RENGAN JAYASEELAN[1], MUKESH KUMAR MAHESHWARI[2], GUNJAN KUMAR CHOUDHARY[1], AND GAURAV SINHA[1], (Member, IEEE)

[1]Samsung Research and Development India-Bengaluru (SRI-B), Bengaluru 560037, India
[2]Department of Electrical Engineering, Bahria University, Karachi 75260, Pakistan

Corresponding author: Sukhdeep Singh (sukh.sandhu@samsung.com)

ABSTRACT Multipath TCP (MPTCP) is an evolution of TCP, capable of using multiple network paths to enhance resilience to network handovers. However, Server-side modification is the key challenge for deployment of MPTCP on a large scale. Therefore, a proxy-based design that uses SOCKSv5 over MPTCP was proposed. Though MPTCP enhances the download experience, it also impacts the browsing experience and Page Loading Time (PLT) due to additional SOCKSv5 protocol signaling overhead. On the other hand, to improve the performance of TCP, Google proposed QUIC (Quick UDP Internet Connection), which addresses the network handover resilience. QUIC also faces server modification as a major challenge. In this article, we propose a novel design of SOCKS over QUIC (QSOCKS), which improves browsing experience while enhancing reliability. QSOCKS ensures 0RTT/1RTT connection time, thereby improving the Page Loading Time (PLT) and Video Loading Time (VLT). We evaluated the performance of QSOCKS through live experiments on the top websites of various web properties located in different regions, using Samsung S9 smartphones. Moreover, we evaluated our proposal for file download scenario in both homogeneous and heterogeneous Wi-Fi & cellular environment. The users not only benefit from the inherent advantages of QUIC but are also privileged with a better browsing experience.

INDEX TERMS Multipath QUIC, SOCKS, QSOCKS, QUIC.

I. INTRODUCTION

Over the last decade, wireless technology has witnessed explosive growth in smartphones, tablets and laptops. These devices have improved the quality of life and opened up new industrial opportunities. According to the Ericsson mobility report, by 2025 5G is going to have 2.6 billion subscriptions, generating 45 percent of the world's total mobile data traffic [1]. A significant amount of today's Internet traffic generated by various protocol includes HTTP: HyperText Transfer Protocol, steaming media, peer-to-peer file sharing, remote access (telnet), file transfer, email (simple mail transfer protocol) traffic. The aforementioned traffic generated is either carried by the Transport Control Protocol (TCP) or the User Datagram Protocol (UDP). TCP provides reliable end-to-end to connection-oriented delivery of packets, whereas UDP provides unordered delivery of IP datagrams. TCP is suitable for the session-oriented protocols and UDP finds an excellent application in carrying multimedia traffic [2]. Due to real-time nature of data (i.e. video and audio), TCP features like re-transmissions, flow control, and reordering are not appropriate [2].

A new general-purpose protocol "Quick UDP Internet Connections (QUIC)" was proposed by Google in 2012. QUIC [3] is a multiplexed, low latency, reliable and encrypted data transfer protocol. It improves the performance of HTTPS-based applications. QUIC is implemented on top of UDP and can multiplex the application streams on a single connection. In contrast to Multipath TCP (MPTCP) [4] and Stream Control Transmission Protocol (SCTP) [5], [6], QUIC does not require any changes to the operating system. The first existing Internet-scale deployment of QUIC, shows high performance and confirms the design decisions with middlebox interference [3]. The IETF has also recently

The associate editor coordinating the review of this manuscript and approving it for publication was Eyuphan Bulut.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

initiated QUIC standardization [7]. The QUIC started as a TCP replacement to transport HTTP/2 and is becoming a universal transport protocol. To prevent the middleboxes from interfering, QUIC encrypts both data and all header fields. Furthermore, the encryption ensures that the protocol will not be ossified by deployed middleboxes [6]. QUIC adopts a flexible packet format and leverages the congestion control of modern TCP stack. Moreover, when downloading different objects from the same server, QUIC supports stream multiplexing which prevents any overhead due to head-of-line blocking [6].

One missing feature of QUIC is the ability to exploit the different paths that exist between a client and server [8]. Nowadays, smart devices have the capability to communicate through two interfaces, such as cellular and Wi-Fi. The user expects that the device should be capable to combine them. Furthermore, a growing fraction of hosts are dual-stack, the IPv4 & IPv6 paths often differ and have different performance [9], [10]. The widely used protocol that exploits multi-pathing is Multipath TCP (MPTCP) [4]. MPTCP is an extension of TCP and enables a TCP connection to send data over multiple paths [4]. Moreover, MPTCP has the capability to aggregate the bandwidth of the different paths [11], [12]. The deployment of MPTCP on Smartphones [13] shows that multipath enables seamless handovers. This gives rise to the need for multipathing in QUIC to utilize both Wi-Fi & LTE and for better resilience to path failures.

There are two motivations behind adding multipath capabilities to QUIC. The first one is to pool the resources of different paths in order to transport the data of a single connection [14]. This allows the dual-stack host to automatically select the best path when the quality of the IPv4 and IPv6 paths differ. The second motivation is the capacity to recover quickly from connectivity failures. In smartphones with two wireless interfaces (Wi-Fi and cellular), one wireless interface may go down or perform poor at any time and users expect that their applications switch to the other interface seamlessly [15].

Motivation: Proxifying QUIC: To enable MPTCP, both client and server need to be MPTCP enabled. However, most of the servers do not support MPTCP by default. To overcome this disadvantage, Samsung partnered with Mobile Network Operator (KT Corporation) to deploy MPTCP, a proxy based approach in all android applications. MPTCP uses SOCKSv5 [16] to proxify and make it server-agnostic. SOCKSv5 introduces connection overhead, consuming multiple round-trip-times to establish a connection. It amplifies the impact when an application uses HTTPS. Measurements carried out in work [17], showed many popular android applications use short-lived connections. We conducted a similar experiment and we observed that most of the applications exchange a small amount of data (as shown in Fig. 1). Increasing the setup time by SOCKS and MPTCP handshake will evidently affect the user experience. SOCKS increases the time required to establish each TCP connection by several round-trip-times between the client and the SOCKS server.

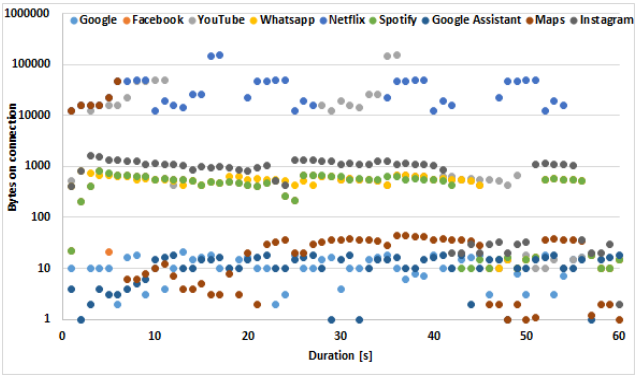


FIGURE 1. Duration and Data Transferred by popular Android APPs.

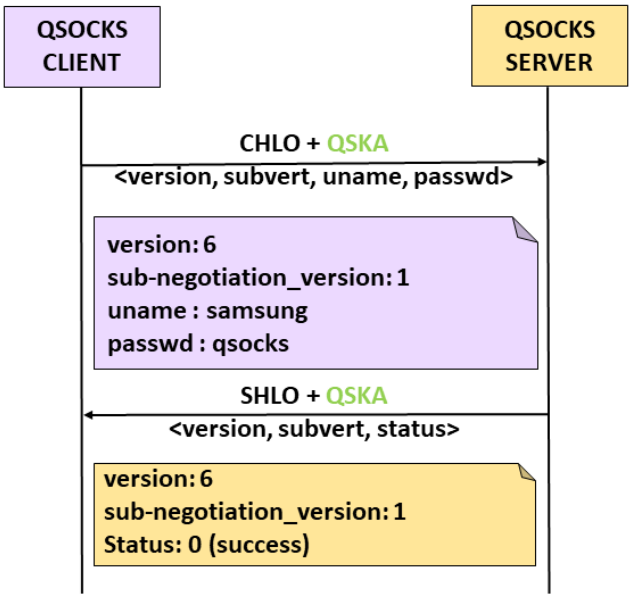


FIGURE 2. Timeline of Connection Establishment in MPTCP. SOCKSv5 requires at-least 4 RTT for each connection to transfer the data to actual destination server.

This additional delay can be significant for applications that rely on short TCP connections. Fig. 2 depicts the additional round-trip-time in MPTCP. To tackle the TCP initial connect overhead, QUIC uses 0-RTT connections that accelerates the latency during the establishment of a connection.

To efficiently combine Wi-Fi and cellular, Multipath QUIC [8] (MPQUIC) was proposed. With the similar limitation as MPTCP, MPQUIC needs server side deployment. The deployment might take a decade and hence proxifying MPQUIC is inevitable. The current SOCKS protocol was designed primarily to proxy TCP by signaling control messages in each TCP connect. Moreover, SOCKS and QUIC does not work hand-in-hand. In QUIC, we establish the connection only once with the SOCKS server and multiplex every request as an individual stream. Hence, a new design for SOCKS over QUIC is required. The design also overcome the disadvantage of several round-trip-times during session establishment in SOCKSv5.

Contributions: Considering the above motivations, we propose QSOCKS which enhances the capabilities of QUIC. We pave the way for Multipath QUIC over QSOCKS which would evidently replace MPTCP by overcoming its shortcomes. The major contributions of our paper are:

1) A first-of-its-kind method of QSOCKS, a SOCKS based QUIC proxy design which enhances the connection establishment using 0-RTT/1-RTT proxy session.
2) In addition to the handshake improvement, QSOCKS also enhances the security by making QSOCKS AUTH and QSOCKS REQUEST incognito using 0-RTT keys. In contrast, the SOCKSv5 AUTH and REQUEST are unencrypted and are prone to security risks.
3) QSOCK's resilience to network handover by leveraging the properties of QUIC.
4) To show the efficiency of our proposal, we analyze the performance of QSOCKS through Live air experiments using Samsung S9 devices. We compare QUIC Booster (QSOCKS + MPQUIC) with GigaLTE (SOCKSv5 + MPTCP) [19]. The simulation results show that QSOCKS outperforms the SOCKSv5 in various environments.

A. IMPACT OF QSOCKS OVER NEXT GENERATION NETWORKS

- QSOCKS is a novel solution that explores the avenue of the world's first deployment of MPQUIC and QSOCKS in smartphones. This is a first-of-its-kind proxy-based architecture which is best-fit for short-lived android applications.
- This paper suggests an interesting idea on Google proposed QUIC with the motto of making services faster on smartphones than ever before.
- We aim to standardize it in IETF and make QSOCKS available to mobile vendors for future low-latency communication, IoT networks and Next Generation Mobile Networks.

The remainder of this paper is organized as follows: Section II presents an overview of existing Multipath protocols, QUIC, Mutipath QUIC and SOCKS protocols. The proposed QSOCKS and the system model is outlined in Section III. Section IV demonstrates detailed evaluation of experimental results. Finally, concluding remarks are provided in Section V.

II. BACKGROUND

In this section, we present an overview of Multipath protocol, QUIC and SOCK Protocols.

A. MULTIPATH PROTOCOL

TCP with multipath enables data scheduling and transmission over different paths. Several studies as in [20]–[24] were carried out to implement TCP over multipath to maximize resource usage and increase redundancy. Multiple-connection TCP (MCTCP) [23] encodes the control information by adding a shim layer. The protocols MMPTCP [21] and MPUDP [22] are specifically designed for data centers and virtual private networks. Multipath TCP (MPTCP) [4] is a widely known transport protocol. MPTCP enables the simultaneous use of multiple paths between peers. MPTCP is designed to successfully establish the initial path and after which subsequent paths are added for multiple transmissions. The MPTCP design suffers from complex TCP header with additional sequence numbers and checksum [6].

B. QUIC PROTOCOL

The single-path QUIC (Quick UDP Internet Connections) is a general-purpose transport protocol developed by Google to accelerate HTTP/S traffic and make it more secure. The advantages of QUIC protocol over TCP+TLS/HTTP2 are: (i) reduce connection establishment latency; (ii) improved congestion control; (iii) stable connection when networks are switched and (iv) forward error correction [3].

QUIC uses UDP as its basis [3]. Each QUIC stream is separately flow controlled. In case of packet loss or a broken steam, it does not affect other streams. The protocol continues to serve other streams independently. The implementation of QUIC on top of UDP makes development and testing simple, can be easily upgraded and shipped with the applications [6].

QUIC provides encryption and authentication for all the packets except for handshake and resets packets. The authentication headers prevent middleboxes form any manipulations. QUIC achieves low latency by sharing the cryptographic information with transport information in a single handshake. The client is aware of initial keys and can establish new QUIC connections with 0-RTT. Moreover, QUIC uses a 64-bit connection ID for every connection. If a device switches from Wi-Fi to cellular or undergoes NAT re-bindings (changes it IP), it can seamlessly migrate the existing connection over the new IP address [25].

Previous research work of authors in [26], [27] enhances the capability of QUIC by adding cross layer approach which in turn adapts to the dynamic network conditions. There is further scope of improvement in terms of proxification which we aim to achieve with the help of our proposed work in this paper.

C. OVERVIEW OF MULTIPATH QUIC

Authors in [6], [8] enhanced the performance of QUIC by enabling Multipath (MPQUIC). The MPQUIC considers path management, packet scheduling, and congestion control. The architecture of MPQUIC [6] is explained below: Path Manager: Like MPTCP or QUIC, MPQUIC performs the cryptographic handshake over the first path. Each path created in MPQUIC/MPTCP is referred to as a subflow. MPQUIC uses Path Manager to control the subflows (creation and deletion). Each packet has a PATH ID in the public header and each subflow is identified by that PATH ID. The presence of PATH ID allows MPQUIC to use multiple flows even during network address translations by middleboxes in the path. Since the QUIC stream can be bi-directional, the paths created by

the client will have an odd PATH ID to avoid PATH ID clashes. Unlike MPTCP that requires a three-way handshake for its subflow establishment, MPQUIC can leverage the 0-RTT path establishment for all subflows after establishing the initial subflow.

Packet Scheduler: The path manager establishes multiple subflows and MPQUIC Packet Scheduler schedules the packets from streams on subflows. The Packet Scheduler considers both subflows' properties and stream requirements. The transmission strategy in MPQUIC is more flexible, as the QUIC frames are not constrained to particular subflow. Hence, the QUIC frames are independent of the packet containing them. In MPQUIC, the scheduler duplicates the data over new subflow when a subflow is created, as the newer subflow's characteristics are not known yet.

Packet Number and Timers: Similar to QUIC, for reliable communication, MPQUIC considers unique packet numbers and stream offsets for each subflow. The Packet scheduler detects any under-performing or broken subflows by monitoring the subflow statistics (e.g. round-trip-time and timeouts) and speed up the handover process for mobility cases.

Congestion Control: The congestion control is used to achieve fairness on the shared bottleneck. By default, MPQUIC adapts OLIA coupled congestion control scheme [28] to control the congestion [8]. However, in wireless environment, MPQUIC adopts the CUBIC congestion control scheme [29]. The CUBIC scheme is used in our experiments as it adapts to bandwidth-delay better in wireless environment.

D. SOCKS PROTOCOL

SOCKS is an Internet protocol that uses a network firewall to separate an organizational internal network structure from an exterior network [16], [18]. The firewall system works as an application-layer between network and offers controlled TELNET, FTP, and SMTP access [16]. Sock protocol provides a convenient and secured utilization of network firewalls for client-server applications in both the TCP and UDP domain. This protocol does not enable network getway services like ICMP message forwarding. Moreover, SOCKS work as a ''shim-layer'' between the application and the transport layer [16].

The SOCKS Version 4, provides unsecured firewall traversal for TELNET, FTP, HTTP, GOPHER and TCP-based client-server applications. SOCKS Version 5, overcomes the weaknesses of SOCK Version 4. The SOCK Version 5, includes UDP, provides strong authentication schemes and extends the addressing scheme by considering the domain-name & V6 IP addresses [16]. The SOCKS protocol implementation involves the appropriate use of encapsulation routines in the SOCKS library for TCP-based client applications [16]. The Procedure for TCP-based clients is given in [16]. The SOCKS implementation with Multipath TCP, enables simultaneous utilization of different access links, which increase the throughput of TCP connections. However, it increases the TCP connection establishment time by several round-trip-times between client and SOCKS server [18]. SOCKSv6 is proposed in IETF [30] for enhancing TCP proxification using TCP Fast Open (TFO) option. This does not work for QUIC as its designed only for TCP. Additionally, in case of HTTPS connection, it also includes multiple RTTs of SSL handshakes, increasing the overall Page Loading Time (PLT).

E. PROXIFICATION OF MULTIPATH PROTOCOL

The Multipath protocols (MPTCP/MPQUIC) can combine both cellular and Wi-Fi networks. However, the server-side deployment might take a longer time. SOCKS is a protocol used by proxy servers for session management. SOCKS handshake further increases the connection establishment time for TCP. QUIC accelerates the HTTP traffic, reduces the connection establishment time and provides a stable connection. However, SOCKS and QUIC do not work hand-in-hand. Motivated from the above studies, we proposed QSOCKS which enables (1) Proxy design; (2) 0-RTT/1-RTT connection establishment; (3) Enhanced security; (4) Improved page Loading Time (PLT)/Video Loading Time (VLT).

III. PROPOSED QSOCKS

In this section, we introduce the design of QSOCKS. The proposed approach reduces round-trip-time during proxification and thus improves the connection establishment time.

A. DESIGN GOALS

We aim to develop a solution considering the following goals:

(i) SOCKS Overhead: Reduce the latency of SOCKS conversations by leveraging 0RTT properties of QUIC.
(ii) Non-Blocking Request: The client asynchronously sends the data along with SOCKS Request without waiting for the response.
(iii) Security: All the SOCKS Authentication and the Connection requests must be secured.
(iv) Confidentiality: Assures end-to-end encryption between client and server.
(v) Reliability: Guarantees the control messages to be reliably delivered with high priority.
(vi) Robust: The protocol should be extensible via options without breaking backward-compatibility.

B. DESIGN

In this subsection, we present the design and implementation of QSOCKS. The proposed QSOCKS is a SOCKS extension for QUIC protocol. QSOCKS consists primarily of two phases: QSOCKS Authentication, and QSOCKS Request.

1) QSOCKS AUTHENTICATION

QSOCKS integrates its SOCKS authentication with the QUIC Crypto handshake. QSOCKS introduces QSOCKS METHOD ('QSKM') and QSOCKS AUTH ('QSKA') tags in the Client Hello (CHLO) message as shown in Fig. 3.

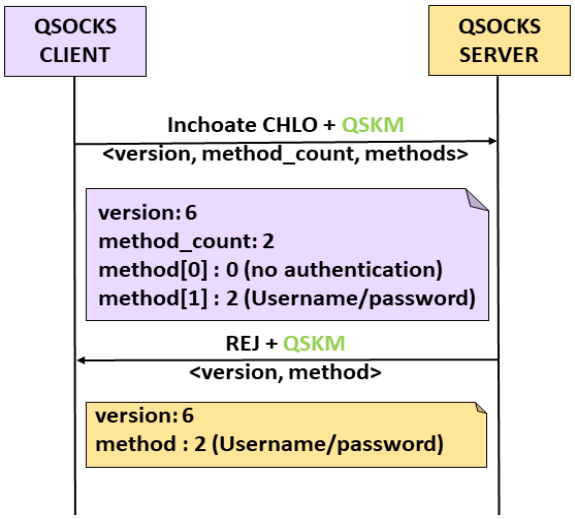


FIGURE 3. QSOCKS METHOD exchange using QSKM tag in QUIC Inchoate CHLO.

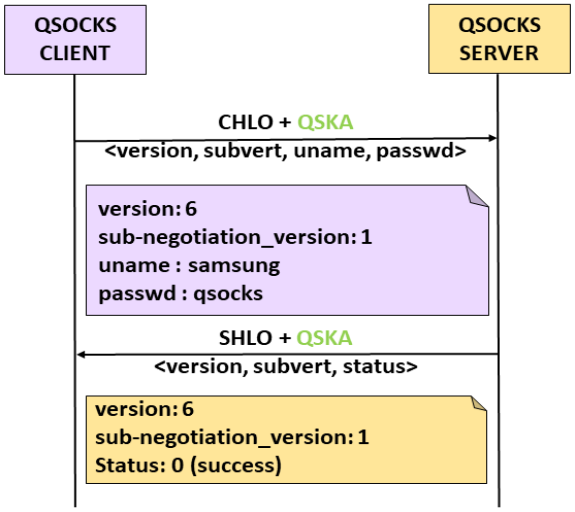


FIGURE 4. QSOCKS AUTH sub-negotiation using QSKA tag in QUIC CHLO.

- Client advertises the client-supported SOCKS AUTH (Authentication) methods using the QSKM tag in QUIC CHLO packet. The server selects one among the advertised methods and communicates the selected method using QSKM in QUIC REJ packet to the client. For our current design, we consider a simple set of AUTH methods as follows:
  - X'00' NO AUTHENTICATION REQUIRED
  - X'01' GSSAPI
  - X'02' USERNAME/PASSWORD
  - X'03' to X'7F' IANA ASSIGNED
  - X'80' to X'FE' RESERVED
  - X'FF' NO ACCEPTABLE METHODS
- QSKA is an optional feature that contains the SOCKS AUTH (authentication) sub-negotiation as shown in Fig. 4. The QSOCKS client, based on the selected AUTH method, advertises its credentials via QSKA request in the QUIC Client Hello message (CHLO). The QSOCKS server verifies the supplied credentials and responds via the QSKA response in the QUIC Server Hello message (SHLO).

2) QSOCKS REQUEST

Once the QSOCKS authentication is complete, every new SOCKS session is initiated via a QSOCKS Request. QSOCKS Request is integrated with the payload of a QUIC STREAM frame. The proposed QSOCKS Request header format is depicted in Fig. 5.

In addition to the legacy SOCKS CMD values, QSOCKS introduces two additional options to support TCP (X'04') and UDP (X'05') over QUIC protocol. The SOCKS server will typically evaluate the request based on source and destination addresses, and return one or more reply messages, as appropriate for the request type. This blocks the application data to be sent to actual destination server.

VER	CMD	RSV	ATYP	DST. ADDR	DST. PORT	REQ. TYPE
1	1	X'00'	1	Variable	2	1

o VER - version: X'06'
o CMD
  o TCP CONNECT X'01'
  o TCP BIND X'02'
  o UDP ASSOCIATE X'03'
  o **TCP over QUIC X'04'**
  o **UDP over QUIC X'05'**
o RSV - RESERVED
o ATYP address type of following address
  o IP V4 address: X'01'
  o DOMAINNAME: X'03'
  o IP V6 address: X'04'
o DST.ADDR - desired dst address
o DST.PORT - desired dst port
o **REQ.TYPE**
  o **ASYNC X'00'**
  o **SYNC X'01'**

FIGURE 5. QSOCKS Request Header format.

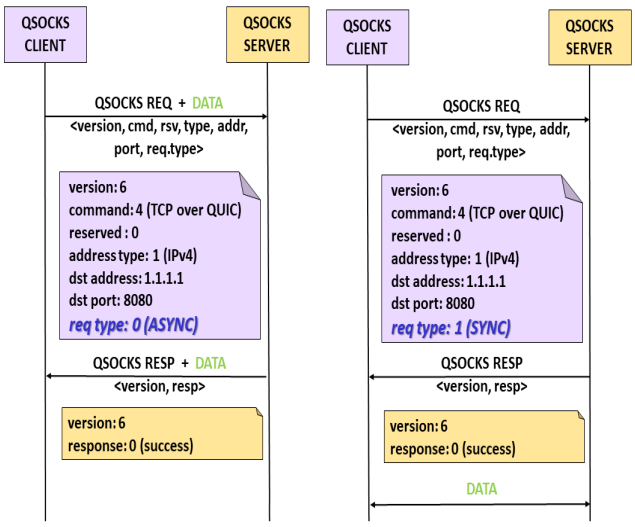


FIGURE 6. Comparison of QSOCKS SYNC and ASYNC Requests. ASYNC Request enables 0-RTT data transfers.

Hence, QSOCKS introduces an additional field in QSOCKS Request, REQ.TYPE. If REQ.TYPE is asynchronous, then it allows the Data to be piggybacked along with the request. If the server responds an error later, the stream is canceled. Fig. 6 compares the asynchronous and synchronous

QSOCKS Request exchange. The user application sessions are made 0-RTT by leveraging asynchronous QSOCKS request.

C. QSOCKS ADVANTAGES OVER SOCKSv5

The key issue in SOCKSv5 is that, it requires 7-RTTs (TCP Handshake, method negotiation, authentication, connection request, connection response, TLS negotiation, TLS grant) for a secured proxy connection and 5-RTTs for a plain proxy connection to exchange data. The QSOCKS is designed to reduce the initial establishment overhead, exchanging data in 0-RTT (illustrated in Fig. 7). This impacts the short-lived connections and page loading time. In addition to the handshake improvement, QSOCKS also enhances the security by making QSOCKS AUTH and QSOCKS REQUEST incognito using 0-RTT keys.

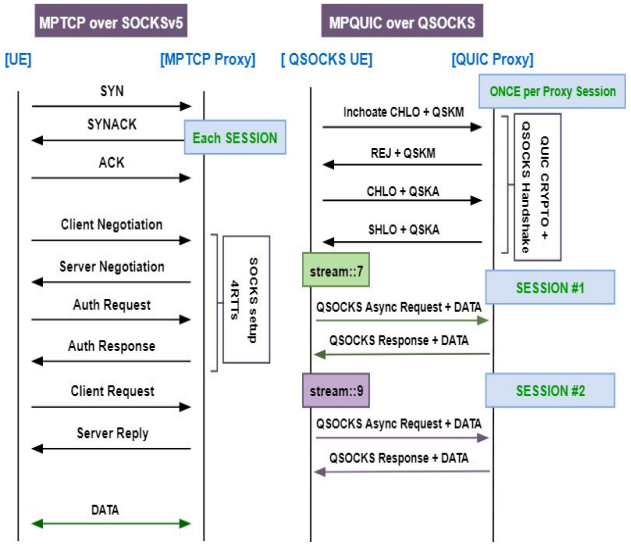


FIGURE 7. The overall flow of QSOCKS and its comparison with TCP SOCKSv5.

IV. PERFORMANCE EVALUATION

This section presents the performance of our proposal. QSOCKS is evaluated through live experiments and mininet based simulation results.

Live Air Experimental Results and Analysis: To prove the deployability and improved performance in real-world environments, we run our QSOCKS solution implemented on the Samsung Galaxy S9.

A. WEBPAGE DOWNLOAD

To show the efficiency of our QSOCKS solution, we compare the Page Loading Time (PLT) recorded with QUIC Booster (MPQUIC over QSOCKS) and the prevalent GigaLTE (MPTCP over SOCKSv5) [19]. We measured the overall PLT, consisting of the Transport handshake, the Session (QSOCKS/SOCKSv5) handshake, and the HTTP requests for all webpage dependencies including all essential files (i.e., JavaScript, CSS, and images). Each data pointed in the graphs represents ten trials with the QUIC booster solution and the average has been plotted.

HTTP Connections: Our first scenario is the study of HTTP webpage downloads in various round-trip-time conditions. We hosted a webpage with multiple objects (object size varying from 10 kB to 100 kB), at different regions and we tested with S9 Smartphones running QUIC Booster and GigaLTE. Fig. 8 confirms the performance improvement of our QSOCKS solution over existing legacy solutions. QSOCKS improves the PLT by 1813 ms, 2375 ms, 3869 ms, and 4404 ms for Delhi, Korea, Virginia, and Brazil, respectively. The 0-RTT session establishment approach in QSOCKS helps in fetching the HTTP wepages 18% faster than SOCKSv5.

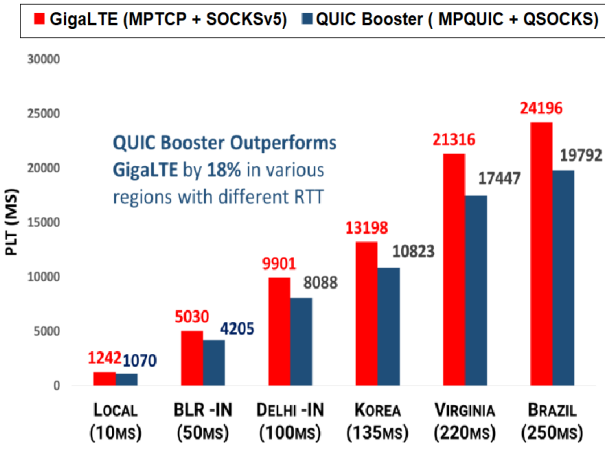


FIGURE 8. PLT Comparison of Plain HTTP Pages between QUIC Booster and GigaLTE.

HTTPS Connections: As of April 2018, 33.2% of Alexa top 1, 000, 000 websites use HTTPS as default [31] protocol. Moreover, 57.1% of the Internet's (137, 971) most popular websites have a secure implementation of HTTPS [32], and 70% of loaded pages (measured by Firefox Telemetry) use HTTPS [33]. Hence, we evaluate our proposal on the top HTTPS websites with different properties as shown in the Table 1.

TABLE 1. HTTPS Webpages Properties.

Website	# of Objects	Size	RTT (in ms)
Google	21	286kB	80
YouTube	102	3.2 MB	63
Facebook	65	1.1MB	65
Top 100	96	2MB	98
Top 1000	112	1.9MB	128

Unlike HTTP, HTTPS additionally uses TLS handshake for establishing a secured connection. We notice that the performance of QUIC Booster is far better than GigaLTE. Fig. 9 shows that QSOCKS reduces the PLT by up-to 42% as compared to SOCKSv5. On the other hand, the PLT in SOCKSv5 is up-to 60% slower than the traditional single path direct TCP connection.

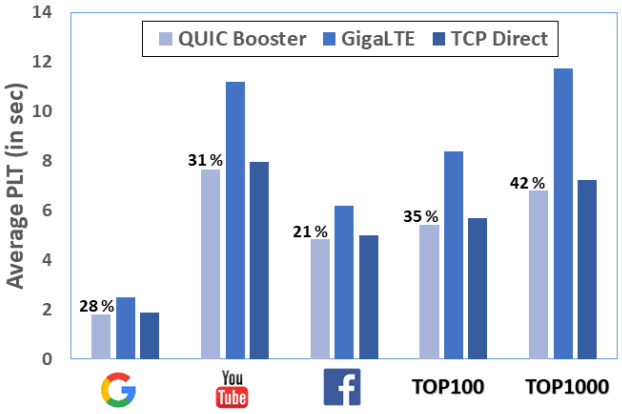


FIGURE 9. PLT Comparison of top websites using QUIC Booster, GigaLTE and direct TCP.

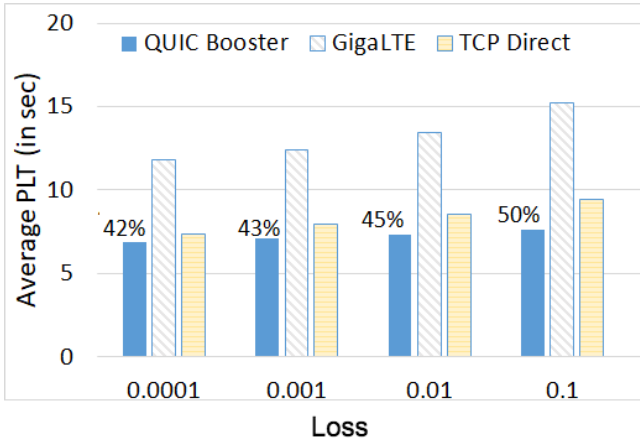


FIGURE 10. PLT Comparison in different loss conditions using QUIC Booster, GigaLTE and direct TCP.

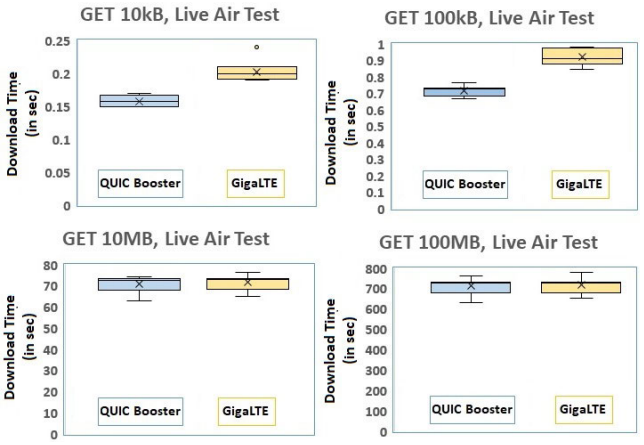


FIGURE 11. File Download in Homogeneous Environment.

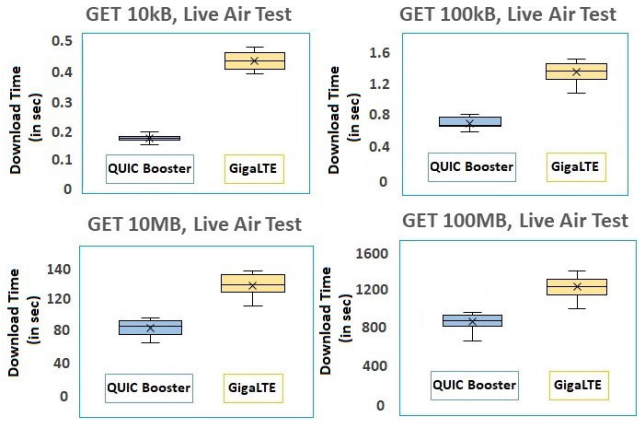


FIGURE 12. File Download in Heterogeneous Environment.

To evaluate QUIC Booster against legacy protocols in lossy network conditions, we performed Page load tests of the top 1000 sites on various loss conditions. To capture the real world wireless loss conditions, we simulated packet loss ranging from 0.0001% to 0.1% with TC netem tool. From the results in Fig. 10, we can clearly infer that the existing protocols like GigaLTE and TCP under-perform with respect to PLT as the packet loss increases. However QUIC Booster copes well with lossy conditions, taking advantage of QUIC's fast loss recovery and pacing in congested networks.

The improvement in PLT using QUIC Booster is mainly because of a) 0-RTT proxification using QSOCKS; b) Leveraging QUIC property to couple TLS negotiation with connection establishment. Whereas, GigaLTE undergoes multiple round-trip-time for SOCKSv5 and HTTPS negotiations. As the top websites are yet to adapt the QUIC protocol, our future work presupposes the evaluation of QSOCKS with legacy QUIC as well.

B. FILE DOWNLOAD

To study the commercial prospect of QSOCKS solution in real-world environment, we evaluate the download times for different file sizes and network characteristics. Each data pointed in the box plot represents 100 trials with the GigaLTE and our proposed QUIC Booster solution.

QUIC Booster and GigaLTE Operates Similar in Homogeneous Environment: We use Wi-Fi and LTE with similar characteristics. The average latency in both the path is around 20 ms. We download multiple files of 10 kB, 100 kB, 10 MB and 100 MB with the help of S9 Smartphones running QUIC Booster and GigaLTE. Fig. 11 shows the comparison graph of different file size in homogeneous environment. For small file size (10 kB and 100 kB), QSOCKS outperforms the SOCKSv5 solution, whereas for larger file size (10 MB and 100 MB) we observe similar performance in terms of download time. The SOCKSv5 handshake impacts the small file downloads in GigaLTE, whereas QUIC Booster completes the download faster with the help of QSOCKS 0-RTT handshakes.

MPQUIC Performs Better in Heterogeneous Environment: For the heterogeneous scenario (latency of 20 ms for Wi-Fi and 50 ms for LTE), we observe that QUIC Booster outperforms the GigaLTE solution irrespective of the file sizes. The MPQUIC Packet Scheduler, with the help of RTT stats and leveraging the QUIC's adaptability to varying network characteristics, copes up better to the heterogeneous conditions

than MPTCP. As depicted in Fig. 12, the path utilization in MPQUIC is prevalent in heterogeneous conditions and hence, QUIC Booster is comparatively more suitable in real time scenario.

V. CONCLUSION

This paper proposed a robust QSOCKS, which enhances the capabilities of QUIC. We presented a novel proxy-based design for QUIC (QSOCKS), bundled with QUIC/Multipath QUIC. QSOCKS ensures 0RTT/1RTT connection time, enhances the Page Loading Time (PLT) and short-lived connections. Moreover, QSOCKS improves the handshake mechanism and also enhances the transmission security as compared to existing SOCKSv5 protocol. Our live air experiments and extensive simulations show that, proposed QSOCKS outperforms the existing mobile proxy solutions in the market such as GigaLTE. Our future work presupposes to propose QSOCKS in 3GPP, to enhance the operation of Access Traffic Steering, Switch and Splitting (ATSSS-TR 23.793 Rel 13) support in 5G Systems.

REFERENCES

[1] P. Jonsson, S. Carson, G. Blennerud, J. K. Shim, B. Arendse, A. Husseini, P. Lindberg, and K. Ôhman. (Nov. 2019). *Ericsson Mobility Report*. [Online]. Available: https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf

[2] A. Handa, *System Engineering for IMS Networks*. London, U.K.: Newnes, 2009.

[3] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, and J. Bailey, "The QUIC transport protocol: Design and Internet-scale deployment," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 183–196.

[4] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 6824, 2013.

[5] R. Stewart, *Stream Control Transmission Protocol*, document RFC 4960, 2007.

[6] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, and R. Steinmetz, "Multipath QUIC: A deployable multipath transport protocol," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[7] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-based secure and reliable transport for HTTP/2 draft-tsvwg-quic-protocol-02," Google, IETF Netw. Work. Group Versions: 00 01 02, Tech. Rep., 2017.

[8] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol.*, Nov. 2017, pp. 160–166.

[9] I. Livadariu, A. Elmokashfi, and A. Dhamdhere, "Characterizing IPv6 control and data plane stability," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[10] I. Livadariu, S. Ferlin, Ö. Alay, T. Dreibholz, A. Dhamdhere, and A. Elmokashfi, "Leveraging the IPv4/IPv6 identity duality by using multi-path transport," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr./May 2015, pp. 312–317.

[11] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, pp. 399–412.

[12] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of MultiPath TCP performance over wireless networks," in *Proc. Conf. Internet Meas. Conf. (IMC)*, 2013, pp. 455–468.

[13] O. Bonaventure and S. Seo, "Multipath TCP deployments," *IETF J.*, vol. 12, no. 2, pp. 24–27, 2016.

[14] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 5, pp. 47–52, Sep. 2008.

[15] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/WiFi handover with multipath TCP," in *Proc. ACM SIGCOMM Workshop Cellular Netw., Oper., Challenges, Future Design (CellNet)*, 2012, pp. 31–36.

[16] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, *SOCKS Protocol Version 5*, document RFC 1928, 1996.

[17] Q. D. Coninck, M. Baerts, B. Hesmans, and O. Bonaventure, "A first analysis of multipath TCP on smartphones," in *Proc. 17th Int. Passive Act. Meas. Conf.* Heraklion, Greece: Springer, 2016, pp. 57–69.

[18] (2018). *Socks and Multipath TCP*. [Online]. Available: http://blog.multipath-tcp.org/blog/html/2018/12/21/socks_and_multipath_tcp.html

[19] *Samsung, KT Announce 5G GiGA LTE, Available for S6 and S6 Edge Today*. [Online]. Available: https://www.sammobile.com/2015/06/16/samsung-kt-announce-5g-giga-lte-available-for-s6-and-s6-edge-today

[20] J. Heuschkel, A. Frömmgen, J. Crowcroft, and M. Mühlhäuser, "Virtual-Stack: Adaptive multipath support through protocol stack virtualization," in *Proc. Int. Netw. Conf. (INC)*, 2016, pp. 73–78.

[21] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proc. IEEE INFOCOM-35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[22] D. Lukaszewski and G. Xie, "Multipath transport for virtual private networks," in *Proc. USENIX Workshop Cyber Secur. Experimentation Test (CSET)*, 2017, pp. 1–8.

[23] M. Scharf and T. Banniza, "MCTCP: A multipath transport shim layer," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2011, pp. 1–5.

[24] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, Oct. 2006.

[25] *The Chromium Projects*. [Online]. Available: https://www.chromium.org/quic

[26] G. Sinha, M. R. Kanagarathinam, S. R. Jayaseelan, and G. K. Choudhary, "CQUIC: Cross-layer QUIC for next generation mobile networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Seoul, South Korea, May 2020, pp. 1–8, doi: 10.1109/WCNC45663.2020.9120850.

[27] G. K. Choudhary, M. R. Kanagarathinam, H. Natarajan, K. Arunachalam, S. R. Jayaseelan, G. Sinha, and D. Das, "Novel MultiPipe QUIC protocols to enhance the wireless network performance," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Seoul, South Korea, May 2020, pp. 1–7, doi: 10.1109/WCNC45663.2020.9120821.

[28] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1651–1665, Oct. 2013.

[29] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[30] *SOCKS Protocol Version 6*. [Online]. Available: https://datatracker.ietf.org/meeting/99/materials/slides-99-mptcp-sessa-socks-protocol-version-6-00

[31] *HTTPS Usage Statistics on top 1M Websites*. Accessed: Oct. 20, 2018. [Online]. Available: https://StatOperator.com

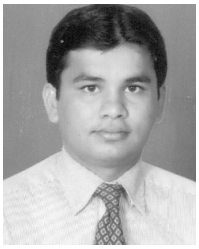
[32] *Qualys SSL Labs-SSL Pulse*. Accessed: Oct. 20, 2018. [Online]. Available: https://www.ssllabs.com

[33] *Let's Encrypt Stats*. Accessed: Oct. 20, 2018. [Online]. Available: https://LetsEncrypt.org

MADHAN RAJ KANAGARATHINAM (Senior Member, IEEE) received the B.E. degree in computer science and engineering from Anna University, Chennai, India, in 2012. He has seven years of working experience in design and development of TCP/IP protocols, multipath TCP, and UNIX flavored operating systems. He was an Engineer with Aricent Technology Private Ltd., India. He is currently a Chief Engineer with Samsung Research and Development India-Bengaluru. He is the author of ten articles. He holds more than 20 inventions. His current research interests include communication and networks, include pre-6G/beyond 5G, next generation mobile networks, software defined network architecture, 40 transport layer protocols, and cross layer optimization technique. He is a member of ACM.

SUKHDEEP SINGH (Member, IEEE) received the Ph.D. degree from Sungkyunkwan University, South Korea. He is currently a Chief Engineer with Samsung Research and Development India-Bengaluru. His work was published by the IEEE, Wiley, Taylor and Francis, and IET. His research interest includes 5G/6G wireless communications.

SUJITH RENGAN JAYASEELAN received the B.Tech. degree in computer science and engineering from the National Institute of Technology, Trichy. He is currently a Software Engineer with Samsung Research and Development India-Bengaluru. His current research interests include next generation mobile networks and transport protocols.



MUKESH KUMAR MAHESHWARI received the master's degree from the University of Leicester, U.K., and the Ph.D. degree from Sungkyunkwan University, South Korea. He is currently a Faculty Member with the Department of Electrical Engineering, Bahria University, Pakistan. His work was published by the IEEE, Wiley, Taylor and Francis, and IET. His research interests include 5G wireless communications and energy-efficient networks.

GUNJAN KUMAR CHOUDHARY has an overall 12 years of industry experience. He worked on Android framework and HTTP protocol. He is currently a Senior Chief Engineer with Samsung Research and Development India-Bengaluru. His work was published by the IEEE. His current research interests include QUIC protocol and connectionless oriented protocols.

GAURAV SINHA (Member, IEEE) graduated in computer engineering from the Thapar Institute of Engineering and Technology, India, in 2018. He joined Samsung Research and Development India-Bengaluru, where he is currently a Senior Software Engineer with the Mobile Communication Division. His research interests include next generation mobile networks, AI in wireless, cross-layer optimizations, applied machine learning, and natural language processing.