Beyond QUIC v1 – A First Look at Recent Transport Layer IETF Standardization Efforts

Mike Kosek, Tanya Shreedhar, and Vaibhav Bajpai

***Abstract*—The transport layer is ossified. With most of the research and deployment efforts in the past decade focussing on the Transmission Control Protocol (TCP) and its extensions, the QUIC standardization by the Internet Engineering Task Force (IETF) is to be finalized in early 2021. In addition to addressing the most urgent issues of TCP, QUIC ensures its future extendibility and is destined to drastically change the transport protocol landscape. In this work, we present a first look at emerging protocols and their IETF standardization efforts *beyond* QUIC *v1*. While multiple proposed extensions improve on QUIC itself, *Multiplexed Application Substrate over QUIC Encryption (MASQUE)* as well as *WebTransport* present different approaches to address long-standing problems, and their interplay extends on QUIC's take to address transport layer ossification challenges.**

***Index Terms*—Transport protocols, IETF, QUIC, MASQUE, WebTransport**

I. Introduction

The transport layer, which is responsible for the end-to-end connectivity between peers, is ossified [1]. While most of the transport protocol related research in the past decade focussed on the extension of TCP as the predominant transport protocol in the Internet, multiple studies showed that this quest is cumbersome, leading to slow adoption of innovative improvements like Explicit Congestion Notification (ECN) (RFC 3168) or Multipath TCP (MPTCP) (RFC 6824), or even no adoption at all. Where previous efforts to deploy novel transport protocols like Stream Control Transmission Protocol (SCTP) (RFC 4960) also did not lead to a wide adoption due to deployment obstacles for non-TCP/UDP-based protocols, using UDP as a substrate protocol promises Internet-scale deployability. Recently, QUIC as a UDP-based protocol set out to solve TCP's issues while ensuring its future extendibility [2]. While QUIC is destined to drastically change the transport protocol landscape, TCP is still the most used protocol, and its importance will not diminish in the near future. Acting as a fallback protocol, services will be offered with both TCP *and* QUIC for an extensive amount of time, and more specialized use-cases like Border Gateway Protocol (BGP) (RFC 4271) might continue to use TCP indefinitely. To ensure continuous improvements, the IETF TCP Maintenance and Minor Extensions (TCPM) Working Group (WG) discusses TCP and MPTCP modifications, and directs the standardization process for proposed specifications. Within TCPM, RFC 8803 as well as RFC 8961 were recently standardized. The *0-RTT TCP Convert Protocol* (RFC 8803) aims at improving the deployment of TCP extensions, where *Requirements for Time-Based Loss Detection* (RFC 8961) discusses best practices to parameterize loss detection algorithms. *The RACK-TLP loss detection algorithm for TCP* [3] was submitted to the Internet Engineering Steering Group (IESG) for publication in December 2020, leveraging Recent Acknowledgements for fast recovery and improving on tail loss by explicitly triggering ACK feedback through Tail Loss Probes. While these additions are essential improvements to TCP, they may not overcome TCP's inherent issues.

With the standardization of QUIC by the IETF to be finalized in early 2021, QUIC's first version *v1* (see §II) addresses the most urgent issues of TCP such as multiplexing, Head-of-line blocking (HOL blocking), mandatory encryption, as well as reduced connection establishment time with 0-RTT support while focussing on the web use-case, i.e., delivery of web content to browsers. Extending on *v1*, the WG actively discusses future extensions, which we will detail in §III. These extensions introduce improvements to version negotiation as well as connection IDs, add multipath capabilities, enable unreliable delivery within QUIC as well as HTTP/3, further extend the future useability of the QUIC protocol, and add performance improvements by negotiating acknowledgement handling.

QUIC's mandatory encryption does provide challenges for specialized use-cases where end-to-end

connectivity is not possible (e.g., censorship), not feasible (e.g., satellite links), or not wanted (e.g., privacy concerns). The IETF MASQUE WG was chartered to address these challenges.

MASQUE (see §IV) proposes the use of QUIC as a substrate protocol, allowing arbitrary data to be tunneled over QUIC. While this addresses TCP proxy use-cases, it also introduces an alternative layering of Virtual Private Networks (VPNs), where nested reliability can be avoided by leveraging QUIC datagram frames.

While QUIC and MASQUE set out to change our transport protocol usage, the web security model limits browser-based web applications to directly access transport protocol features. Protocols like WebSocket (RFC 6455) and WebRTC (RFC 7478) [4] were indispensable in rejuvenating static request-response-based web content and benefited from years of deployment of their substrate protocols. However, they also inherited their fundamental disadvantages. The WebTransport WG (see §V) addresses these shortcomings by utilizing QUIC as a substrate protocol, exposing its features to browser-based web applications while considering fallback mechanisms to traditional TCP-based connections.

In this article, we present a first look at these most recent transport layer IETF standardization efforts *beyond* QUIC *v1*. While our work does not cover advances in congestion control schemes such as Bottleneck Bandwidth and Round-trip propagation time (BBR) or related standardization work by the Internet Congestion Control Research Group (ICCRG), we refer the inclined reader to [5] [6]. The remainder of this article is structured as follows: §II briefly introduces QUIC, where §III details future extensions beyond *v1*. §IV presents recent developments in the usage of QUIC as a substrate protocol within MASQUE, where §V details the advances of providing novel transport protocol features within the web security model pursued by the WebTransport WG. Finally, §VI details the interplay of the presented IETF standardization efforts, followed by the conclusion in §VII.

II. QUIC v1

QUIC was launched by Google in 2012 [7] with the goal to provide secure and reliable low latency end-to-end transport. Google added support for Chrome in 2013, and by 2017, all Chrome

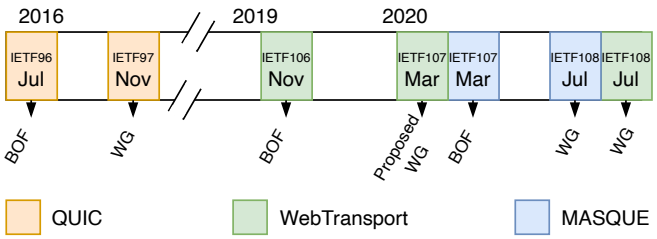

Fig. 1. *Timeline of recent IETF transport layer standardization efforts. The QUIC working group was established at IETF '97 in 2016, followed by the establishment of MASQUE as well as WebTransport at IETF '108 in 2020.*

and Android YouTube app users were using QUIC. QUIC provides stream multiplexing without the drawbacks of TCP's HOL blocking. The initial design idea of QUIC was provided by SPDY, which was later standardized as HTTP/2 (RFC 7540), enabling the multiplexing of streams using the same TCP connection. The IETF chartered the QUIC working group in 2016 (see Figure 1) to provide a standards-track specification for a UDP-based, stream-multiplexing, encrypted transport protocol based on Google's pre-standardization implementation and deployment experiences. QUIC's WG charter holds several goals and milestones relating to the core transport functionality, security, the mapping between different HTTP protocols, the extension of core protocol facilities, and the applicability and manageability of the implications of the protocol. QUIC mitigates the HOL blocking issue by leveraging stream multiplexing in the transport layer, improves on connection establishment times by sending a cryptographic handshake as part of the transport handshake, and provides 1-RTT handshakes for first-time connections as well as 0-RTT handshakes for subsequent connections using TLS 1.3 (RFC 8446).

III. QUIC Extensions

While the working group initially focussed on the web use-case, many QUIC extensions have recently been proposed. In the following, we will discuss the proposals listed in Table I.

An Unreliable Datagram Extension to QUIC: QUIC transmits a reliable stream of application data where reliability is achieved on a per-stream basis. The proposed extension enhances QUIC with the support for unreliable datagrams, aiding in use-cases where reliability is undesired (e.g., real-time communication). With its reduced handshake latency,

WG	Document	Type	Reference
QUIC	WG Charter	Charter	/wg/quic/about/
	An Unreliable Datagram Extension to QUIC	WG I-D	draft-ietf-quic-datagram
	QUIC-LB: Generating Routable QUIC Connection IDs	WG I-D	draft-ietf-quic-load-balancers
	Compatible Version Negotiation for QUIC	WG I-D	draft-ietf-quic-version-negotiation
	3GPP Access Traffic Steering Switching and Splitting	Ind I-D	draft-bonaventure-quic-atsss-overview
	Multipath Extensions for QUIC	Ind I-D	draft-deconinck-quic-multipath
	Multipath Extension for QUIC	Ind I-D	draft-liu-multipath-quic
	Greasing the QUIC Bit	Ind I-D	draft-quic-bit-grease
	Sender Control of Acknowledgement Delays in QUIC	Ind I-D	draft-quic-delayed-ack

TABLE I
OVERVIEW OF QUIC IETF DOCUMENTS. **TYPE** DIFFERENTIATES DOCUMENT TYPE: WG CHARTERS ARE DENOTED AS **CHARTER**, ADOPTED INTERNET-DRAFTS AS **WG I-D**, AND INDIVIDUAL DRAFTS AS **IND I-D**

unreliable delivery via QUIC improves on existing solutions such as DTLS (RFC 6347). Moreover, its multi-streaming feature can be leveraged to multiplex reliable and unreliable streams within one connection, thereby providing partial reliabilty, and use pluggable congestion control where required. Another use-case for unreliable delivery are VPNs, requiring generic IP packet tunnelling as provided by MASQUE (see §IV).

QUIC-LB: Generating routable QUIC connection IDs: QUIC maintains a connection ID (CID) per connection, which allows migration during network changes, and provides unlinkability features across connection migration. If servers do not provide additional CIDs, they might choose linkable CIDs from servers behind load balancers. In this situation, the client either terminates the connection during the migration or remains linkable, violating QUIC's design goal. QUIC-LB specifies standards for encoding routing information given a small set of configuration parameters. Using QUIC-LB, load-balancers communicate the algorithm parameters to generate routable CIDs rather than generating individual CIDs to servers.

Compatible version negotiation for QUIC: Currently, the QUIC server indicates if a client offered version is not accepted, but does not provide information to select a mutually supported version. The proposed version negotiation mechanism allows a client and a server to leverage the similarities between different versions and establish a mutually supported/compatible version without the overhead of extra round trips.

Multipath Extension for QUIC: The QUIC Multipath extension preserves the single-path QUIC design features while simultaneously using multiple paths for a single connection. The introduction and preliminary work on multipath QUIC was presented in [8]. Recently, the 3rd Generation Partnership Project (3GPP) started a discussion to enable Access Traffic Steering, Switching, and Splitting (ATSSS) service for QUIC on multiple paths, where IETF standardization and specifications can be beneficial to attain the ATSSS design goals. Table I lists the Informational 3GPP ATSSS Overview document, as well as multiple individual multipath QUIC drafts which are actively discussed within the WG. As of now, no consensus has been reached on the adoption of multipath QUIC in general, or a specific proposal in particular. While all proposals differ in their way to handle multipath QUIC requirements like linkability between flows, they are commonly in-line with multipath extensions of other transport protocols such as Concurrent Multipath Transfer SCTP (CMT-SCTP) [9] or MPTCP (RFC 8684) [10]. These include features like bandwidth aggregation, seamless handovers, and improved user Quality-of-Experience related to the increasing number of multi-homed devices. As an example, connection migration can be leveraged on ordinary QUIC connections to move a single QUIC flow from one IP address to another, resulting in a *hard handover*. Like MPTCP, multipath QUIC improves on this, allowing devices to seamlessly switch from one interface to another, thus providing resilience to connection failures. Extending on these common multipath features, the primary motivation behind multipath QUIC however lies in the aggregation of all available network resources to send data through a single connection [11]. While this is useful for e.g., large transfers, it also benefits dual-stacked hosts, automatically selecting the best available path in case the quality of the IPv4 and IPv6 paths differ [12]. A descriptive example of multipath QUIC is presented in §VI.

Greasing the QUIC Bit: Intermediaries and end-

points use the *QUIC Bit* to distinguish QUIC from other protocols. A fixed value is currently sent in the QUIC Bit of every packet, thus allowing end-points and intermediaries to depend on a fixed value. By leveraging the concept of *GREASE* (Generate Random Extensions And Sustain Extensibility), the *grease_quic_bit* transport parameter ensures the future usage of the QUIC Bit by indicating that an end-point is willing to receive QUIC packets regardless of this bit's value.

Sender Control of Acknowledgement Delays in QUIC: A receiver acknowledges the reception of data from the sender. Delaying these acknowledgements reduces the CPU utilization at both sender and receiver and potentially improves throughput. However, these benefits are traded off by negatively impacting congestion control and loss recovery. The *Sender Control of Acknowledgement Delays in QUIC* extension allows the end-points to advertise the *min_ack_delay* transport parameter, which defines the minimum amount of time an ACK can be delayed.

While these proposals improve on QUIC, there usage requires both communication partners to mutually support an extension. As deployment experience with TCP has shown, this can lead to slow adoption, or even no adoption at all. *Pluginzing QUIC* [13] enables QUIC end-points to dynamically exchange protocol extensions on a per-connection basis, therefore requiring only one communication partner to feature an extension.

IV. MASQUE

Driven by the shortcomings of proxying mechanisms like native HTTP Proxies (unencrypted, HTTP/TCP), Socket Secure (SOCKS) (unencrypted signaling, TCP and UDP), HTTP CONNECT (encryption optional, TCP), or transparent TCP Proxies (must be on-path, mandatory to use, TCP), the IETF MASQUE WG (see Figure 1) was formed in early 2020. MASQUE is chartered to develop mechanisms that will allow arbitrary connections to be tunneled within a single HTTP/3 connection using explicit client-initiated signaling. Besides the existing request/response model and authentication mechanisms of HTTP, which can be leveraged for service and parameter negotiation, QUIC's unified congestion controller will greatly improve on the uncoupled flows handled by traditional proxies, and allow multiple client-initiated reliable and unreliable connections to be transported within a single HTTP/3 connection. To address censorship use-cases, the tunneled data will be indistinguishable to arbitrary encrypted HTTP connections on the wire, preventing hints which possibly expose the nature of the connection to adversaries. Moreover, to address instances where UDP and/or HTTP/3 is actively blocked on the client-proxy leg of the connection, the MASQUE WG will consider HTTPS/TCP as a fallback.

Initially proposed within the QUIC WG, *Using QUIC Datagrams with HTTP/3* (see Table II) was recently moved to and adopted by the MASQUE WG as a WG Internet draft. While the unreliable datagram extension of QUIC (see §III) provides a mechanism to send reliable and unreliable data simultaneously leveraging the security and congestion-control properties of QUIC, it is unable to de-multiplex application contexts. *Using QUIC Datagrams with HTTP/3* adds flow identifiers for HTTP/3 applications at the start of the frame payload. This *Datagram-Flow-Ids* represent bidirectional flows in a single QUIC connection and allow multiplexing and de-multiplexing of the application data. This concept is leveraged within MASQUE as well WebTransport (see §V).

As a primary focus for the WG, *CONNECT-UDP* (see Table II) proposes a UDP-based counterpart to the TCP-only *HTTP CONNECT* method. While it would be possible to reuse *HTTP CONNECT* for UDP, existing implementations would fallback to TCP on the proxy-server leg of the connection, which should be avoided. However, *CONNECT-UDP* will be supported on HTTP/1.1, 2, and 3, and therefore provides a TCP fallback mechanism on the client-proxy leg of the connection as detailed earlier. Using the *CONNECT-UDP* header, the client instructs the proxy to open a UDP connection to a provided URI. For HTTP/3, QUIC datagram frames are leveraged, providing a proxied unreliable connection between client and server. This enables connections to multiple servers to be transported within the same client-proxy HTTP/3 connection, which are multiplexed and de-multiplexed using *Datagram-Flow-Ids* While the chaining of multiple proxies is supported, a proxy receiving *CONNECT-UDP* can either connect to the indicated target or to an upstream proxy. To use UDP on an end-to-end path, all involved proxies have to support HTTP/3

WG	Document	Type	Reference
MASQUE	WG Charter	Charter	/wg/masque/about/
	Using QUIC Datagrams with HTTP/3	WG I-D	draft-ietf-masque-h3-datagram
	The CONNECT-UDP HTTP Method	WG I-D	draft-ietf-masque-connect-udp
	Requirements for a MASQUE Protocol to Proxy IP Traffic	WG I-D	draft-ietf-masque-ip-proxy-reqs
	The CONNECT-IP method for proxying IP traffic	Ind I-D	draft-kuehlewind-masque-connect-ip
	QUIC-Aware Proxying Using CONNECT-UDP	Ind I-D	draft-pauly-masque-quic-proxy
	Discovery Mechanisms for QUIC-based Proxy Services	Ind I-D	draft-kuehlewind-masque-proxy-discovery
	Transport Considerations for IP and UDP Proxying	Ind I-D	draft-westerlund-masque-transport-issues
WebTransport	WG Charter	Charter	/wg/webtrans/about/
	The WebTransport Protocol Framework	WG I-D	draft-ietf-webtrans-overview
	WebTransport using HTTP/2	Ind I-D	draft-kinnear-webtransport-http2
	WebTransport over HTTP/3	Ind I-D	draft-vvv-webtransport-http3
	WebTransport over QUIC	Ind I-D	draft-vvv-webtransport-quic

TABLE II
OVERVIEW OF MASQUE AND WEBTRANSPORT IETF DOCUMENTS. **TYPE** DIFFERENTIATES DOCUMENT TYPE: WG CHARTERS ARE DENOTED AS **CHARTER**, ADOPTED INTERNET-DRAFTS AS **WG I-D**, AND INDIVIDUAL DRAFTS AS **IND I-D**

leveraging QUIC datagram frames. Following successful negotiation, all intermediaries will switch to tunnel mode and restrict to forwarding packets until the connection is closed.

Besides *CONNECT-UDP*, the requirements for generic *IP Proxying* (see Table II) addressing traditional VPN use-cases are actively discussed, and were recently adopted as a WG Internet draft. Favoring HTTP/3 using QUIC datagram frames to prevent nested reliability, a fallback to HTTP/2 is also supported, leveraging both protocols multiplexing capabilities to run multiple IP proxied connections over the same HTTP connection. For this purpose, an IPv4 or IPv6 session has to be established between the end-points, including support for IP address assignment requests, route negotiation, and client and server identification as well as authentication. Where *IP Proxying* lays out the requirements for proxying IP packets, *CONNECT-IP* (see Table II) proposes a specific method to enable IP proxying using HTTP/3 connections, thus partially covering the outlaid requirements. A descriptive use-case of *IP Proxying* is presented in §VI.

To proxy arbitrary QUIC connections, *QUIC-Aware Proxying Using CONNECT-UDP* (see Table II) addresses the specifics of tunneling QUIC over QUIC for long header packets, e.g., the encapsulation and encryption overhead of nested QUIC connections, as well as the forwarding of short header QUIC packets on established connections by leveraging connection IDs.

Exceeding the presented efforts, supplemental topics are discussed within the WG which are also shown in Table II. *Discovery Mechanisms for QUIC-based Proxy Services* discusses mechanisms to enable clients to be able to discover non-transparent MASQUE proxies, while *Transport Considerations for IP and UDP Proxying in MASQUE* addresses challenges to preserve end-to-end properties of the proxied flows.

V. WEBTRANSPORT

The web security model shapes the Internet landscape. While abstracting transport protocol features to application layer protocols and exposing them to web developers, browser-based web applications became truly interactive and highly dynamic, radically replacing static request-response based content.

The TCP streams exposed by the WebSocket protocol (RFC 6455) provide bidirectional ordered delivery and suffer from HOL blocking as well as mandatory reliability, making it a poor fit for real-time communication or latency-sensitive applications. This is improved by bootstrapping WebSocket onto HTTP/2 (RFC 8441), which multiplexes arbitrary streams in a single HTTP/2 connection, hence eliminating HTTP HOL blocking, but still suffering from TCP HOL blocking. Layering WebSocket onto HTTP/3 would solve this issue. However, existing disadvantages persist, requiring additional round-trips for the WebSocket protocol handshakes for every stream, limiting connection initiation to clients only, and lacking support for unreliable transport.

WebRTC (RFC 7478) [4] data channels improve on this while leveraging SCTP (RFC 4960), providing ordered and unordered delivery, partial reliability, and eliminating HOL blocking. As SCTP faced deployment challenges (see §I), SCTP WebRTC data channels use UDP as a substrate, a pattern also embraced by QUIC (see §II).

The IETF WebTransport WG (see Figure 1) was formed to provide a mapping of HTTP and QUIC-based protocols to a web interface API developed by the World Wide Web Consortium (W3C) [14] honoring the web security model. The utilized protocols (referred to as *transports*) mandate uni- and bi-directional streams, datagram support, and encryption. Moreover, optional properties are defined, which rely on features of specific protocols. These include stream independence to prevent HOL blocking, partial reliability to prevent retransmissions of datagrams, pooling support to share a unified congestion controller, connection migration to keep connections alive if the path changes, and bandwidth prediction to aid use-cases like video streaming or real-time gaming.

While the core incentives of WebTransport have been discussed since 2018 as QUIC standardization progressed, the WG was chartered in March 2020, currently defining the requirements of WebTransport, and the requirements on the utilized *transports* *Http2Transport*, *Http3Transport*, as well as *QuicTransport* (see Table II). A descriptive example of WebTransport is presented in §VI.

Http2Transport allows WebTransport peers to multiplex arbitrary bidirectional streams over HTTP/2 connections, where either end-point can initiate a new stream. While WebTransport and regular HTTP data can be multiplexed simultaneously, intermediaries traversed must explicitly support WebTransport. Additionally, TCP HOL blocking remains an issue, and the mandated support for unidirectional streams and unreliable delivery are noticeably missing. While unidirectional streams can be forged by requiring bidirectional streams to only use one half of the connection, unreliability can not be provided as TCP forcibly retransmits HTTP/2. As datagrams are *not* expected to be reliably delivered, but they *might* if the *transport* is using a TCP-based protocol, the specification also covers this fallback case. Additionally, *Http2Transport* does feature pooling support, which ensures that a shared congestion controller between multiple *transports* sharing the same HTTP connection can be used.

Http3Transport does support all requirements covered by *Http2Transport*, and extends on it by also providing unidirectional streams, unreliable delivery leveraging QUIC datagram frames with HTTP/3 (see §IV), as well as stream independence which eliminates HOL blocking.

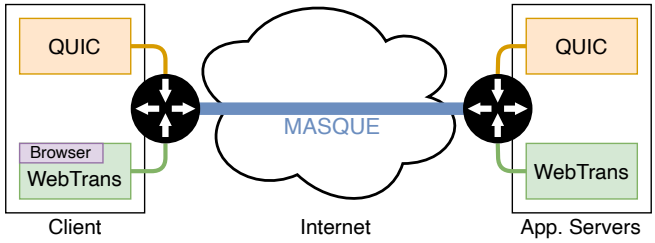

Fig. 2. *Remote office use-case: A client requests resources via QUIC and WebTransport from Application Servers, which are multiplexed over a MASQUE tunnel proxying arbitrary IP packets.*

Lastly, *QuicTransport* offers a minimal protocol on top of QUIC, where WebTransport concepts are directly mapped to the corresponding QUIC counterparts if applicable. The main design goal is a low overhead protocol, minimizing implementation effort and complexity for extending existing QUIC stacks with *QuicTransport* capabilities. *QuicTransport* satisfies all WebTransport design requirements except pooling support.

Besides the three presented proposals, a fourth option of *FallbackTransport* (no active document) is discussed within the WG. Aiming at a mapping to HTTP/1.1 and HTTP/2, multiplexed streams can be simulated on top of the WebSocket protocol, where the existing standardized WebSocket mappings to the HTTP protocols are utilized as-is.

While *QuicTransport* offers a solution with low overhead, low complexity, and minimal implementation effort, *Http3Transport* offers pooling support as well as HTTP features like status codes, headers, load balancing, and rerouting, possibly outweighing the increased complexity and interdependency. Acknowledging these advantages, an adoption call was recently issued for the *Http3Transport* proposal, aiming to focus the WGs ressources at WebTransport over HTTP/3 in the foreseeable future.

VI. Interplay

While all discussed protocols and extensions propose vastly different approaches, their interplay extends on QUIC's take to address transport layer ossification challenges. We present two different use-cases highlighting their combined benefits. Figure 2 showcases a remote office scenario. A client requests resources using plain QUIC as well as WebTransport via a browser. The client runs a VPN service leveraging a MASQUE tunnel to proxy arbitrary IP packets, connecting to a VPN gateway at the main office that de-multiplexes the tunnel and

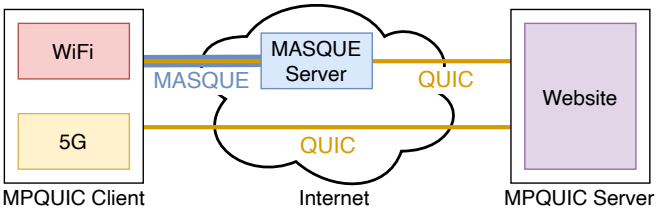

Fig. 3. *Mobile use-case: A client is connected to a server using multiple paths, where one QUIC connection is proxied via a MASQUE server.*

proxies the requests to their respective application servers. A mobile use-case is presented in Figure 3. A multipath QUIC client is connected to a multipath QUIC server using WiFi and 5G simultaneously, thus featuring multiple end-to-end paths. One connection is proxied using a MASQUE server, where the end-to-end QUIC connection is tunneled within the MASQUE QUIC connection. Benefitting from both the proxied MASQUE connection optimized for the access network as well as the multipath capabilities, the client's packet scheduler can dynamically select the optimal path and seamlessly re-route packets in case of path property changes or connection losses.

VII. Conclusion

The transport layer is evolving. With QUIC at the core of this renewal, its future versions will build on the foundation of QUIC v1 deployed on the Internet, thereby extending its reach to increasingly more application areas. While multiple extensions improve on QUIC itself, MASQUE shows promise to supersede traditional proxies and VPNs, and WebTransport will further enhance the rejuvenation of the Web, thus aiding the development of next-generation Web applications. While this article offered a first impression at the recent transport layer IETF standardization efforts *beyond* QUIC *v1*, the presented protocols and extensions propose different approaches to address long-standing problems, and their interplay extends on QUIC's take to address ossification challenges. Marking only the beginning, a new era of protocols is about to emerge.

Acknowledgments

We would like to thank Lars Eggert and Mirja Kühlewind for their feedback and guidance, as well as the editor and the reviewers for their valuable remarks.

References

[1] G. Papastergiou *et al.*, "De-Ossifying the Internet Transport Layer: A Survey and Future Perspectives," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, 2017. [Online]. Available: https://doi.org/10.1109/COMST.2016.2626780

[2] Y. Cui *et al.*, "Innovating Transport with QUIC: Design Approaches and Research Challenges," *IEEE Internet Computing*, vol. 21, no. 2, 2017. [Online]. Available: https://doi.org/10.1109/MIC.2017.44

[3] "The RACK-TLP loss detection algorithm for TCP," *(Date accessed: 03.01.2021)*. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tcpm-rack/

[4] C. Jennings *et al.*, "Real-time communications for the Web," *IEEE Communications Magazine*, vol. 51, no. 4, 2013. [Online]. Available: https://doi.org/10.1109/MCOM.2013.6495756

[5] M. Polese *et al.*, "A Survey on Recent Advances in Transport Layer Protocols," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3584–3608, 2019. [Online]. Available: https://doi.org/10.1109/COMST.2019.2932905

[6] T. Zhang and S. Mao, "Machine Learning for End-to-End Congestion Control," *IEEE Communications Magazine*, vol. 58, no. 6, 2020. [Online]. Available: https://doi.org/10.1109/MCOM.001.1900509

[7] A. Langley *et al.*, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *SIGCOMM*, 2017, pp. 183–196. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[8] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and Evaluation," in *CoNEXT*, 2017, pp. 160–166. [Online]. Available: https://doi.org/10.1145/3143361.3143370

[9] J. R. Iyengar *et al.*, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, 2006. [Online]. Available: https://doi.org/10.1109/TNET.2006.882843

[10] M. Li *et al.*, "Multipath Transmission for the Internet: A Survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2887–2925, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2016.2586112

[11] J. Qadir *et al.*, "Resource Pooling for Wireless Networks: Solutions for the Developing World," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 4, pp. 30–35, 2016. [Online]. Available: https://doi.org/10.1145/3027947.3027953

[12] V. Bajpai and J. Schönwälder, "A longitudinal view of dual-stacked websites—failures, latency and happy eyeballs," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 577–590, 2019. [Online]. Available: https://doi.org/10.1109/TNET.2019.2895165

[13] Q. De Coninck *et al.*, "Pluginizing QUIC," in *SIGCOMM*, 2019, pp. 59–74. [Online]. Available: https://doi.org/10.1145/3341302.3342078

[14] "PROPOSED WebTransport Working Group Charter," *(Date accessed: 03.01.2021)*. [Online]. Available: https://w3c.github.io/webtransport-charter/charter.html

Mike Kosek is a PhD Student at TUM, Germany. His current research focuses on Internet architectures in general, and transport protocol standardization, development, and deployment, in particular.

Tanya Shreedhar is a PhD student at IIIT-Delhi, India. Her research interests include next-generation networks and systems with a focus on transport layer protocols.

Vaibhav Bajpai is a senior researcher at TUM, Germany. He is interested in performance and management of next-generation networked systems.