

A Policy-Based Path Selection Mechanism in QUIC Multipath Extension

Masahiro Kozuka
Graduate School of Humanities and
Social Sciences
Okayama University
Okayama, Japan

Yasuo Okabe
Academic Center for Computing and
Media Studies
Kyoto University
Kyoto, Japan
<https://orcid.org/0000-0003-0825-2256>

Abstract— The new transport protocol QUIC supports connection migration and support for multipath extension is also in under standardization. However, currently discussed QUIC multipath extension does not include a mechanism for each application to select which of the possible paths to use. Even when an application is using multiple paths for a connection, it is not possible for the application to use a different path for each stream, taking into account the characteristics of each path and the requirements of each stream. In this paper, we have considered the need for path selection in multipath-aware transport protocols with mobile devices such as smartphones that have asymmetric upstream connections, Wi-Fi and mobile connections, and based on this, we have we propose a mechanism that determines which paths to include in QUIC connections based on policy for each process, and a mechanism that groups paths included in QUIC connections based on policy and specifies which group to use for each stream or flow of datagrams in QUIC data transmission for applications using QUIC. By incorporating the mechanism into our IP VPN over QUIC system, applications that do not support QUIC can benefit from resiliency and efficiency of QUIC multipath extension and policy-based path selection. to assign voice connections and video connections or screen sharing to different paths. Its DSCP support also allows common real-time applications such as Zoom and WebEX to assign voice connections and video connections or screen sharing to different paths depending on the characteristics of each path like metered or unmetered.

Keywords—*QUIC, multipath, path selection, IP VPN over QUIC, prioritization, DSCP.*

I. INTRODUCTION

Major contemporary mobile devices, such as smartphones, are equipped with two types of network connections to the Internet: LTE/5G mobile data communications and Wi-Fi. While Wi-Fi access in general has relatively broad bandwidth and low latency, it is unstable and susceptible to device movement and ambient noise, so it is expected that these two methods will be used simultaneously and in accordance with the characteristics of the application. However, at present, even if a mobile device is in a multi-homing state having connection via mobile data communication network and connection via Wi-Fi, the selection of route is determined by the OS of the device, and applications running on it cannot choose or prioritize one over the other.

In many cases, the Wi-Fi connection is configured to take priority, and hence if the Wi-Fi becomes unstable, communication will be interrupted until it is completely disconnected; When walking through an area with not a few sparsely populated Wi-Fi APs, frequent switching can cause a

significant reduction in QoE for applications such as teleconferencing. In iOS, Siri uses MPTCP (multipath extension of TCP) [1] to switch dynamically to ensure stability, but this is extremely exceptional. Therefore, when a user wants stable communication for an application, they have no choice but to turn off Wi-Fi communication of the device.

Recently, a new transport protocol, QUIC, proposed under the leadership of Google, was standardized [2]. Unlike ordinary TCP, QUIC allows connection migration by switching the connection path to another address when a device has multiple IP addresses. Furthermore, the IETF QUIC WG has started to study multipath extensions similar to TCP, and several Internet Draft has been proposed and discussed in the WG. Currently, HTTP/TCP is used for the majority of mobile device communications, and the shift to HTTP3/QUIC in the future is expected to lead to the widespread use of QUIC multipath extensions. However, the current discussion on QUIC multipath extension does not include a mechanism for each application to select which of the possible paths to use. Even when an application is using multiple paths for a connection, it is not possible for the application to use a different path for each stream, taking into account the bandwidth, delay, and other characteristics of each path and the requirements of each stream.

In this study, we propose a mechanism that determines which paths to include in QUIC connections based on policy for each process, and a mechanism that groups paths included in QUIC connections based on policy and specifies which group to use for each stream or flow of datagrams in QUIC data transmission for applications using QUIC. For example, if an application does not want to use paths on metered Internet connections at all, the application is launched with the path filtering policy “do not include metered Internet connections” in the path filter mechanism. This prevents paths on metered Internet connections from being included in the QUIC connection. If, for example, an application such as a web conferencing system wants only voice data to be sent and received using the paths on metered Internet connection, the application is started the path selection mechanism with a policy so that all available paths are used for sending and receiving voice data, while no path on metered Internet connections is used for sending and receiving other data.

We have designed and implemented the proposed mechanisms. Rust is used as the development language, Cloudflare's quiche is used as the base QUIC implementation, and Windows 11 and Linux are supported as platforms. The policy can specify the available (or unavailable) network interfaces, IP address ranges, and, in the Windows version, whether or not to use a metered Internet connection.

These research results were obtained from the commissioned research by National Institute of Information and Communications Technology (NICT), JAPAN.

Unfortunately, there are currently few applications that use QUIC other than web browsers. Therefore, in this study, we incorporated the proposed path selection mechanisms into a VPN over QUIC system which we have developed, that can create a virtual tunnel using QUIC between a client and a server, so that applications that do not support QUIC can easily perform policy-based path selection. In the VPN over QUIC system, IP packets are transmitted as QUIC datagram frames. The policy of which path a QUIC datagram frame takes can be specified according to the value of DSCP (DiffServ Code Point) in the header of the corresponding IP packet. This allows voice communication, which is marked with a high priority in applications that support DSCP marking such as Zoom and WebEX, to take precedence in path selection over video and screen sharing.

The rest of this paper is organized as follows: Section II describes QUIC and QUIC datagram extension; Section III discusses multiplexing and prioritization in QUIC; In Section IV we argue on issues in multihoming with multiple addresses and present the design of our path selection mechanism; Section V describes the design of our IP VPN over QUIC system; Section VI provides a summary and discusses a future issue.

II. QUIC DATAGRAM EXTENSION

A. The QUIC Transport Protocol

QUIC is designed as a transport protocol implemented on UDP, replacing not only HTTP/2 but also native TCP, with TLS 1.3 security features. It is designed to speed up connection establishment, has a mechanism equivalent to TCP Fast Open and TLS Snapstart, achieves 0-RTT, and improves signaling for packet loss detection. Flow control, retransmission control, and congestion control have also been improved by introducing monotonically increasing packet numbers to replace TCP sequence numbers, which eliminates retransmission ambiguity, supporting up to 256 ACK blocks, and pacing to prevent packet loss. In addition, the introduction of connection ID reduces reconnection loss, especially for mobile clients. Security features, such as resistance to packet amplification attacks, have also been enhanced.

One of the key features of QUIC is migration. If a client's IP address or port number changes during communication, specifically if the network interface used is switched from Wi-Fi to a mobile network (and vice versa), or if an intermediate NAT assigns a different port number, the connection is not maintained by TCP and the application must create a new connection. The TCP multipath extension [1] can handle the former case by allowing multiple network interfaces to be used simultaneously, but it cannot handle the latter NAT case. QUIC uses connection IDs as connection identifiers, independent of IP addresses and port numbers, to maintain connectivity in both cases through migration.

Furthermore, the IETF QUIC WG is considering a QUIC multipath extension [3] that would enhance resilience and load balancing by using multiple interfaces simultaneously.

B. QUIC Datagram Extension

QUIC provides a multiplexed connection for transmitting reliable streams of application data. QUIC uses various frame types to transmit data within packets, and each frame type defines whether the data it contains will be retransmitted. Streams of reliable application data are sent using STREAM frames.

In addition to that, an extension to QUIC have been standardized to support the transmission of unreliable application data [4]. This provides secure datagram transfer with the added benefit of sharing the cryptographic and authentication context used for reliable streams.

The RFC says “Unreliable QUIC datagrams can also be used to implement an IP packet tunnel over QUIC, such as for a Virtual Private Network (VPN). Internet-layer tunneling protocols generally require a reliable and authenticated handshake followed by unreliable secure transmission of IP packets. This can, for example, require a TLS connection for the control data and DTLS for tunneling IP packets. A single QUIC connection could support both parts with the use of unreliable datagrams in addition to reliable streams.” However, to the authors' knowledge, no VPN over QUIC implementations have been found, except at PoC level ones.

III. MULTIPLEXING AND PRIORITIZATION IN QUIC

A. Stream Prioritization in QUIC

In QUIC, streams in a connection are multiplexed and transmission of packets is supposed to behave based on priority information received from the application (typically HTTP/3). The Extensible Prioritization Scheme for HTTP [5] provides an extensible priority mechanism and specifies urgency, which takes integer values from 0 to 7, and incremental, which is 0 or 1, as its basis, in place of the HTTP/2 stream priorities specified in dependencies and weights defined in RFC 7540 [6]. However, the specific behavior of QUIC with respect to priority in scheduling packet delivery is implementation-dependent and not specified in RFCs.

B. Scheduling in Transports That Support Multipath

When only one path is used at a time, including the case of migration in QUIC, scheduling only needs to consider the order in which packets are sent out, and an appropriate priority-based algorithm such as Priority Queue, Weighted Round Robin, Class-Based Queue, etc. can be used. In contrast, for QUIC multipath extension that use multiple paths simultaneously, each packet must also be considered in terms of which path it will be sent out to, in addition to the scheduling of packets.

Coninck et al. incorporated an implementation of MPTCP's default scheduler, lowest round-trip time first (LowRTT), into the QUIC multipath extension and evaluated it [7]. Considering that LowRTT is not efficient for heterogeneous multipaths such as combination of mobile connections and Wi-Fi, Lim et al. proposed a scheduler called Earliest Completion First and evaluated in MPTCP [8]. Rabitsch et al. proposed a scheduler called Stream-Aware Earliest Completion First (SA-ECF), which takes into account the priority of each stream in asymmetric multipath, and incorporated it into QUIC for evaluation [9]. Shi et al. obtained even better performance than SA-ECF by using schedulers specialized for per-stream priorities called FStream [10] and PStream [11]. A comprehensive survey on the state-of-the-art in multipath transmission protocols identifies how they can be leveraged in access traffic steering, switching and splitting, and suggests how they can help meet 5G requirements [12].

Stream-aware path selection must also accommodate changes in path characteristics due to changes in connection status over time. In his doctoral dissertation [13], Coninck

analyzed the multipath TCP mode in iOS and described as follows. In iOS, applications can explicitly request the use of Multipath TCP using the iOS API. Apple provides three modes of operation for Multipath TCP with different purposes: *handover*, *aggregate*, and *interactive*. Of these, the interactive mode attempts to make the connection with the lowest latency and is recommended for connections with small amounts of data that are sensitive to latency. In interactive mode, the Wi-Fi network is preferred over the cellular network, marking the latter as a backup sub-flow. This ensures that peers will not use cellular networks unless they experience connectivity issues with paths on Wi-Fi, such as retransmission timeouts (RTOs).

IV. POLICY BASED PATH SELECTION

A. Issues in Multihoming with Multiple Address

Currently, even if a mobile device is in a multihoming situation, where it is connected to both a mobile network and a Wi-Fi network, the device's OS decides which one to use for the connection, and applications on it cannot choose either. Normally, the Wi-Fi connection is set to have priority (Fig. 1 (a)), so if a mobile device is communicating only with a mobile network and then newly connects to a Wi-Fi access point, the already established TCP and UDP connections will be interrupted, even though the mobile network will remain connected. Some real-time applications that specifically address this problem, such as Zoom, will detect a communication breakdown and quickly reestablish the connection to continue communication, but even then a few seconds of interruption is inevitable.

One point to consider when distinguishing between mobile and Wi-Fi connections is that mobile data communications are usually metered billing. Thus users prefer to use Wi-Fi connections, which are not metered, if at all possible. In Windows and Android OS, not only mobile connections but Wi-Fi connections have settings that allow to identify whether the connection is metered or not (Fig. 1 (b)), and metered billing connections are not used for background communications such as updates of OS or applications. The Facebook application on smartphones can be configured to change the behavior of video playback when connected to Wi-Fi and when connected to a mobile device (Fig. 2). However, the application's response is only how it behaves with respect to the connection which is selected by the OS, and the application cannot select the connection. A teleconferencing application such as Zoom could be better if it can use uninterrupted mobile communication for audio stream even with metered charges because voice is the most important in interaction, while uses Wi-Fi for video streaming and screen sharing since they are not essential.

QUIC is designed with the assumption that the client side has multiple IP addresses, and if the address used for communication becomes unavailable, the client can switch to another address while maintaining the connection through migration. The QUIC multipath extension further uses multiple paths associated with multiple addresses at the same time. However, even with these features, it is not possible for applications to distinguish between paths associated with a mobile connection and a Wi-Fi connection based on the performance characteristics and metered billing as described above.

B. Design of Path Selection Mechanism

Considering the issues discussed in Section IV-A, we aim to solve these by providing the following path selection mechanism.

1) *Path filtering*: Path filtering is a mechanism that determines, based on specified policy, whether the path using an IP address of a local network should be included in a QUIC connection, and provides the functionality described below when an application obtains the available IP addresses of a local network via the library. At library initialization, a pre-specified policy for the filter function is set. Specific policies allow the following combinations to be specified:

- Available (or unavailable) network interfaces,
- Available (or unavailable) IP address ranges,
- Whether or not to use a metered connection network (if supported by the target OS).

Applications are notified if there is a change in available IP addresses after library initialization. If the policy at the time of initialization is set to "do not include the path of the metered connection network," for example, the application will not be notified even if a change in an IP address belonging to the metered connection network is detected. When notifying, information on whether the target IP address belongs to the metered billing connection network or not shall be added.

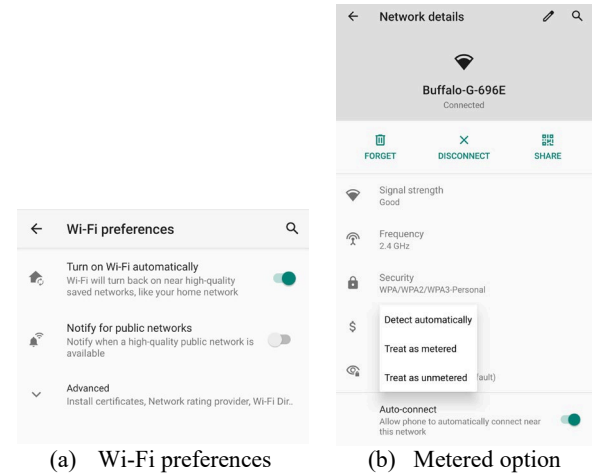


Fig. 1 Android Wi-Fi preferences and metered option.

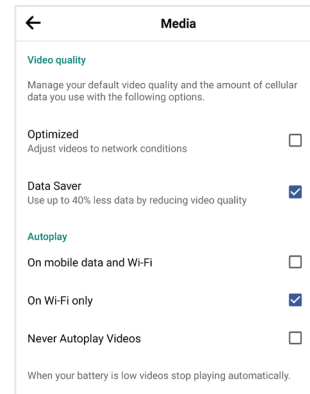


Fig. 2 Video autoplay setting in the Facebook application.

2) *Path grouping*: When a client-side application calls the API of the QUIC implementation, it is able to specify which path group the path associated with each IP address in the QUIC connection belongs to. Multiple path groups can be specified to which a path belongs. Regardless of whether or not a path group is specified, all paths are included in path group 0. When a client-side QUIC stack detects a change in a path group, it shares the path group information with the server-side QUIC stack using a Path Group Frame, which contains a list of path group IDs and the IDs of the paths to which they belong.

3) *Path selection*: The API is modified to allow both client and server side to specify the path group to be used for transmission when creating a stream or sending a datagram. If not specified, it is assumed that path group 0 is specified. In principle, the transmission priorities of streams (Stream Prioritization (RFC9000 2.3) and datagrams (HTTP Datagram Prioritization) shall be kept for each path when sending packets. For example, if Stream 0 is configured to use path group 1 with priority 0, and Stream 4 is configured to use path group 0 with priority 1, then the transmission of Stream 0 is attempted first. However, if there is no room in the congestion window for paths that belongs to path group 1, then Stream 4 is sent if it is possible to send Stream 4 on a path that does not belong to path group 1.

We have designed and incorporated the above functionality into *quiche*, Cloudflare's QUIC implementation¹. The implementation is written in Rust, the implementation language of *quiche*. The mechanism that allows users to specify which of multiple addresses to use for paths in a connection using per-process rules incorporates the same kind of proposal in SCTP [14].

V. DESIGN OF IP VPN OVER QUIC SYSTEM

A. Why IP VPN over QUIC is Needed

By using QUIC instead of TCP as the transport protocol, it is possible to benefit from features of QUIC including migration and future multipath extensions, such as continued communication even if the IP address changes. Since QUIC is a transport layer protocol, applications must have direct QUIC support on both the server and the client to use QUIC. However, many existing applications use TCP as their transport protocol, and to use QUIC, each of them needs to work separately to support QUIC.

Since QUIC implementations are generally designed to operate in userland, not in kernel, it is very difficult to support QUIC by modifying libraries linked to applications, etc., and modifications must be made at the source code level for each existing application. Therefore, modifications must be made at the source code level for each existing application.

Furthermore, since QUIC can be used in userland separately from the kernel, many implementations of QUIC exist, and the API for using QUIC from application programs is not standardized and differs greatly from one QUIC implementation to another. Unlike SCTP [15], there is no API defined that is compatible with the socket API. Therefore, the

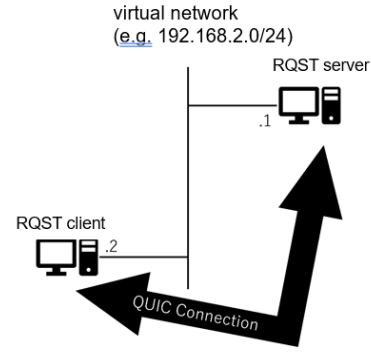


Fig. 3 Overview of IP VPN over QUIC system

hurdle to use QUIC from applications that do not currently support QUIC is high.

A few Web browsers such as Google Chrome and Mozilla Firefox support HTTP/3, and QUIC is automatically used if the Web server supports HTTP/3. Or if the application supports connection via an HTTP proxy, it is possible to set up a local HTTP/3-compliant proxy server to convert normal HTTP/1.1 or HTTP/2 requests to HTTP/3 and use QUIC to communicate with an HTTP/3-compliant server. However, there is no easy way to use QUIC in applications that use protocols other than HTTP, especially UDP datagrams.

In this research, the QUIC datagram extension [4] is used as a VPN to realize datagram delivery with the advantages of QUIC like TLS1.3 authentication and encryption, congestion avoidance without retransmission, speed-up functions and migration.

B. Overview of the IP VPN over QUIC system

We have developed a VPN over QUIC system that can create a virtual tunnel using QUIC between clients and servers, and named it RQST. This system allows communication between client and server using QUIC, even if the client application does not support QUIC. If the server is installed on a high-quality network, communication can continue even if the IP address of the client changes due to mobility or other reasons.

The structure of the VPN over QUIC system is shown in Fig. 3. A QUIC connection is established between the client and server, and packets are relayed between the virtual I/F of the client and the virtual I/F of the server. That is, (1) packets entering the client's virtual I/F are output from the server's virtual I/F through the above QUIC connection, and (2) packets entering the server's virtual I/F are output from the client's virtual I/F through the above QUIC connection.

The IP VPN over QUIC system² was implemented in the Rust language, using Cloudflare QUIC implementation *quiche* that supports the QUIC multipath extension, allowing simultaneous use of multiple paths.

C. QUIC Connection

The QUIC connection between client and server is established by the client sending a QUIC initial packet to the server to initiate the TLS1.3 handshake. The default value of the UDP port on the server side is 3456, but it can be changed through a configuration file. The server authenticates the

¹ <https://github.com/cityroam/quiche/tree/beyond5g-20230317>

² <https://github.com/cityroam/rqst>

client using the TLS1.3 client authentication or password method.

IP packets input to one virtual interface are transmitted through the QUIC connection using datagram frames and output from the opposite virtual interface. Datagram frames are not retransmitted, and hence retransmissions are not duplicated even for IP packets that convey TCP segments.

D. Virtual Interface

Establishing a QUIC connection from the RQST client to the RQST server creates a virtual network between the client and server. Packets flowing in the virtual network are encapsulated in QUIC datagram frames and delivered to the client server through the QUIC connection between the client and server. The IP address of each virtual I/F is set by the user in advance. Since the virtual network is an L2 network that operates as a virtual Ethernet, a DHCP server or similar device can be run on the server to automatically assign IP addresses to the virtual I/Fs of the clients. The server can also act as a router and perform NAT, or bridge the server's virtual I/F with the I/F connected to the router, so that clients can communicate through the server-side network.

E. Incorporate Path Selection Mechanism into the System

We have made the path selection functionality described in Section V.B with the IP VPN over QUIC system. Incorporates. It allow users to specify path filtering like “not to include paths in the metered connection network.” The path grouping function allows users to create path groups that “do not include paths in the metered connection network.”

In addition to the 4 tuple (source/destination |IP address/port number) that identify the TCP or UDP flows, the value of the DiffServ Code Point (DSCP) field in the IP header can be used to specify which path group to use when encapsulating and forwarding IP packets into QUIC datagrams at the VPN tunnel interface.

The developed software is now released public as open-source software under the Apache license.³

VI. CONCLUDING REMARKS

In this paper, we have considered the need for path selection in multipath-aware transport protocols with mobile devices such as smartphones that have asymmetric upstream connections, Wi-Fi and mobile connections, and based on this, we have proposed, designed, and implemented a policy-based path selection mechanism in QUIC multipath extension. By incorporating the mechanism into our IP VPN over QUIC system, applications that do not support QUIC can benefit from resiliency and efficiency of QUIC multipath extension and policy-based path selection. to assign voice connections and video connections or screen sharing to different paths. Its DSCP support also allows common real-time applications such as Zoom and WebEX to assign voice connections and video connections or screen sharing to different paths depending on the characteristics of each path like metered or unmetered.

In the path selection mechanism, it is assumed that the user knows the characteristics of each path and sets the policy based on them. In practice, however, especially in Wi-Fi connections, the characteristics can change dynamically and drastically due to movement of the device and other factors.

The design and implementation of a mechanism to dynamically measure the bandwidth and delay of each path and switch paths accordingly is a future issue.

REFERENCES

- [1] A. Ford, C. Raiciu, M. Handley, O. Bonaventure and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 8684, March 2020. [Online]. Available: <https://www.rfc-editor.org/info/rfc8684>.
- [2] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: <https://www.rfc-editor.org/info/rfc9000>.
- [3] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema and M. Kühlewind, "Multipath Extension for QUIC," Internet Draft, draft-ietf-quic-multipath-04, March 2023. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/>.
- [4] T. Pauly, E. Kinear and D. Schinazi, "An Unreliable Datagram Extension to QUIC," RFC 9221, March 2022. [Online]. Available: <https://www.rfc-editor.org/info/rfc9221>.
- [5] K. Oku and L. Pardue, "Extensible Prioritization Scheme for HTTP," RFC 9218, June 2022. [Online]. Available: <https://www.rfc-editor.org/info/rfc9218>.
- [6] M. Belshe, R. Peon and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540, May 2015. [Online]. Available: <https://www.rfc-editor.org/info/rfc7540>.
- [7] Q. d. Coninck and O. Bonaventure, "Multipath QUIC: Design and Evaluation," in *13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '17)*, Incheon, Republic of Korea, 2017.
- [8] Y.-s. Lim, E. M. Nahum, D. Towsley and R. J. Gibbens, "ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths," in *13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '17)*, Incheon, Republic of Korea, 2017.
- [9] A. Rabitsch, P. Hurtig and A. Brunstrom, "A Stream-Aware Multipath QUIC Scheduler for Heterogeneous Paths," in *Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ'18)*, Heraklion/Crete, Greece, 2018.
- [10] X. Shi, L. Wang, F. Zhang and Z. Liu, "FStream: Flexible Stream Scheduling and Prioritizing in Multipath-QUIC," in *25th International Conference on Parallel and Distributed Systems (ICPADS)*, Tianjin, China, 2019.
- [11] X. Shi, L. Wang, F. Zhang, B. Zhou and Z. Liu, "PStream: Priority-Based Stream Scheduling for Heterogeneous Paths in Multipath-QUIC," in *29th International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, USA, 2020.
- [12] H. Wu, S. Ferlin, G. Caso, Ö. Alay and A. Brunstrom, "A Survey on Multipath Transport Protocols Towards 5G Access Traffic Steering, Switching and Splitting," *IEEE Access*, vol. 9, pp. 164417-164439, 2021.
- [13] Q. D. Coninck, "Flexible Multipath Transport Protocols," Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2020.
- [14] S. Maruyama, M. Kozuka, Y. Okabe and M. Nakamura, "Policy-Based IP Address Selection in SCTP Automatic Address Reconfiguration," in *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, Biopolis, Singapore, 2011.
- [15] R. Stewart, M. Tuexen, K. Poon, P. Lei and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)," RFC 6458, Dec. 2011. [Online]. Available: <https://www.rfc-editor.org/info/rfc6458>.

³ <https://github.com/cityroam/rqst/tree/beyond5g-20220317>