Multipath QUIC – Directions of the Improvements

Michał Morawski(✉) and Michał Karbowańczyk

Lodz University of Technology, Lodz, Poland
{michal.morawski,michal.karbowanczyk}@p.lodz.pl

Abstract. The multipath transmission becomes the recognized alternative for traditional Quality of Service architectures. Recently, the multipath version of TCP protocol and its modern replacement – QUIC – has been proposed. The paper presents the dynamic properties of the data transfer between physical systems, engaging the multipath version of QUIC protocol (MPQUIC) which inherits the properties of its predecessors. The advantages and weaknesses of the transmission are emphasized and compared to the singlepath QUIC. While QUIC is designed to convey HTTP traffic, in the paper, general-purpose networking is investigated. Based on the measurements, the use recommendations are given together with the directions of improvements.

Keywords: Multipath QUIC · Congestion control · Head-of-line blocking

1 Introduction

The traditional way for dealing with network-related data transfer performance flaws is addressed to routers or switches configuration. The intermediate devices tamper packet orders and force drops according to rules defined in ISA [1] and DSA [2] architectures. Such an approach is mature and practical, but it has two serious drawbacks: it neglects the last/first mile link problems and is applicable in proprietary networks only. When using the public Internet, privileging some traffic over another is prohibited by law (network neutrality [3]). Today, the most vulnerable links are located at first/last mile (LTE, DSL, Wi-Fi, etc.). They are shared, prone to noise, congestion, and interferences. Thus, their throughput and delays fluctuate significantly. Furthermore, the network neutrality principle causes the traditional Quality of Service approaches to be hardly applicable for smartphones, IoT nodes, or even wirelessly connected laptops.

Instead of dealing with network and link layer solutions, recently, the transport protocols gain attraction. They are implemented at endpoints, so the modifications may be introduced irrespectively to service providers. Despite their continuous development [4], transport protocols are often considered ossified. Almost all the Internet transmissions are controlled by forty-year-old TCP and UDP protocols because the ubiquitous network devices (e.g., switches, routers, firewalls, load balancers) deal only with these Internet workhorses. Other protocols are restricted to specific environments. Recently, this ossification has been relaxed, and quantum leaps have emerged:

W. Xiang et al. (Eds.): BROADNETS 2021, LNICST 413, pp. 193–207, 2022.
https://doi.org/10.1007/978-3-030-93479-8_13

- *Multipath transmission*. Nowadays, when a new device is purchased, it is usually equipped with multiple network interfaces. Other ones may be easily added. Nevertheless, without enormous efforts, the user cannot engage them simultaneously, esp. when connecting to the same peer. The default interface is assigned statically, and even if the active path stalls or breaks, the client persistently tries to use the malfunctioned links, despite the availability of other options. If multiple communication paths are available, the simultaneous degradation of all possible ones is unlikely. Hence, when one path degenerates, the data transfer may be seamlessly shifted to other ones. While dispatching packets using different interfaces is straightforward, establishing transport sessions using old protocols is impossible. They identify a peer using its logical address, protocol, and port number. These entities are assigned to logical endpoints, so data originated from another interface of the same peer are rejected. Apart from peer identification, data transport needs to handle the data coherency and pace the transfer by adjusting it to fluctuations of available bandwidth. The recently developed multipath version of TCP (MPTCP) [5–7] allows for simultaneous transfer of data through different interfaces and paths and grants the establishing and tear down communication channel within a single logical connection.
- *Sending data in parallel*. TCP protocol has been designed for stream transfers, i.e., it correctly handles long, bulk data, e.g., a file transmission. Recently, many applications exchange relatively short but numerous, logically unrelated data – e.g., series of HTTP asynchronous requests and responses. If such programs employ classical TCP, they need either open as many separate TCP connections as necessary or multiplex logical streams within the single TCP flow. The first solution seems straightforward, but it requires many resources (memory and CPU) from both peers, especially if the transmission is encrypted, which is expected today. It requires additional code to permit identification and synchronization with the remaining transfers, as well. As explained in Sect. 3, multiplexing logical channels within a single TCP connection suffers from mutual blocking of parallel streams. This problem is incredibly intense for HTTP traffic. Therefore, Google has elaborated protocol Quick UDP Internet Connection – QUIC [8]. It discards the obsolete elements of TCP, but the algorithms governing the transfer are transplanted and adjusted. While developed with HTTP in mind, QUIC may be deployed in many other applications in general networking, IoT, or industrial data exchange.

Both MPTCP and QUIC may be perceived as improved in different directions versions of TCP. Indeed, they share with TCP most of the algorithms that govern the data flow. Consequently, the idea of merging MPTCP and QUIC has popped up. The Multipath QUIC (MPQUIC) has been developed [9] by the same team as MPTCP. It should inherit all the advantages of QUIC and MPTCP. In this paper, MPQUIC governed transmission is verified experimentally using public networks and real devices. It extends the initial evaluations [11–13] by investigating the dynamics and mutual interactions of particular streams. The strengths and weaknesses of the current implementation of MPQUIC are identified, and the research direction is proposed.

The rest of the article is organized as follows. In Sect. 2, the general issues of multipath communication are presented together with a short remainder of the

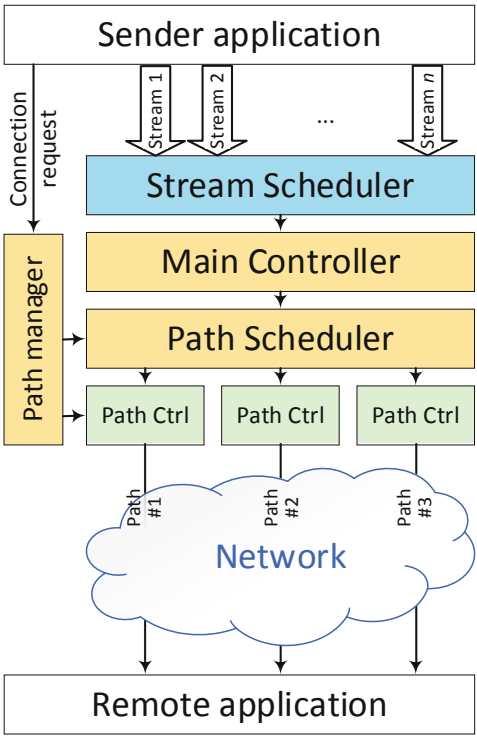

Fig. 1. Architecture of MPQUIC protocol. Elements taken from TCP are marked in green, from MPTCP – in yellow, from QUIC – in blue. (Color figure online)

state-of-the-art. In Sect. 3, the Head-of-Line blocking is explained as the most critical phenomenon impacting the multipath and multistream transmission. It allows introducing measures of transport efficiency. Next, in Sect. 4, the design of the physical setup is motivated and presented. Then, in Sect. 5, the results of the analysis of particular protocol modules are assessed. Finally, the observations are summarized in Sect. 6 and concluded in Sect. 7.

2 Multipath Transmission

Multipath transmission is the desirable feature recognized as an inexpensive way to increase transmission throughput. It is easily applicable at the link layer but not in the higher layers. Therefore, multiple attempts to design corresponding transport protocols have been taken up. The comprehensive survey concerning this topic can be found at [14]. Unfortunately, most of these solutions are not widely accepted due to challenges with their practical implementation. The common consensus for designing new network protocols requires providing the highest possible efficiency (high throughput, short delays) at the lowest possible cost (memory usage, CPU cycles used). Additionally, transparency to other layer protocols is desirable. It enforces compatibility

with the installed network devices – switches, routers, firewalls, load balancers. In many cases, compatibility with existing applications is necessary as well.

In general, the multipath transport protocols suffer from the increase of the protocol delay (i.e., the delay experienced by applications) and uneven path loads [15]. This drawback is the result of the path blocking explained in the next section. Another deficiency concerns transmissions of short streams [16]. While the protocol stack algorithms make decisions in real-time without a clue concerning future transfer properties and requirements, they must be robust enough to efficiently handle any data transfer patterns.

The first widely accepted multipath transport protocol was SCTP [17]. Unfortunately, due to limited support from ubiquitous security devices, it is challenging to use SCTP for general networking. Additionally, appropriate applications need to be adjusted for this protocol. Therefore, SCTP protocol is currently widespread in telecommunication networks for SIGTRAN (SS7 over IP) signaling. Nevertheless, SCTP is still developed, and this research is transplanted to other solutions.

So far, the only multipath transport protocol which has received higher acceptance is MPTCP. Its design has met the enthusiastic response from researchers and is continuously improved, tuned, and complemented. Currently, in terms of general networking, its only competitor is the recently designed MPQUIC [9] protocol – an extended version of QUIC initially developed for HTTP traffic, but evaluating towards general-purpose applications [10]. Both multipath protocols are similar, but they differ in essential details, impacting their efficiency, as investigated in the paper. Both MPTCP and MPQUIC are fully transparent for network devices, although the default configuration of some firewalls blocks MPTCP. MPTCP is implemented in a kernel space, so it is transparent for the applications as well. Unfortunately, it is available only for Linux (partially, also for IOS and Android). MPQUIC must be included as a library by an application that requires more developers' efforts but breaks the dependency on an operating system.

The MPQUIC architecture (Fig. 1) reveals its similarity to previously developed protocols – QUIC, MPTCP, and legacy TCP. When an application requests a new connection, the *Path Manager* module is triggered. As in MPTCP, the module is responsible for the addition and removal of paths during the connection lifetime. Its activity is unrelated to data transfer. While for a single connection, the activity MPQUIC *Path Scheduler* is the same as in the MPTCP counterpart, then for the subsequent connections to the same peer, the behavior of the protocols is different. MPTCP repeats the process, but MPQUIC internally handles a new logical data stream. Instead of creating the new connection, MPQUIC employs *Stream Scheduler* initially developed for QUIC. By default, it runs under a round-robin policy, but the actual actions may be adjusted to particular needs [18–20]. Next, user data are handled by *Main Controller,* which shapes general flow properties. By default, it takes responsibility for transmission fairness, i.e., it restricts favoring multipath streams over single path ones. MPTCP uses LIA algorithm [6], and MPQUIC adjusted version of OLIA [21] – protocol elaborated initially for MPTCP. Next, data are split over currently available paths by *Path Scheduler*. By default, both multipath protocols use the path, which has the lowest delay and can acquire new data. However, the MPQUIC *Path Scheduler* is more flexible and may insert the frames belonging to different application streams into the same packet.

Many other strategies are developed, e.g., [22–25]. Nevertheless, *Path* and *Stream Schedulers* are unrelated and do not cooperate with each other.

After splitting a data stream, packets are transferred to the peer using *Path Controllers*. MPQUIC employs controllers similar to TCP [4]. Contrary to most TCP congestion control algorithms, it senses link bandwidth not based on drops as in TCP but based on an increase of packet delay. If it grows too much, congestion control leaves the slow-start phase. The linear-grow phase is pretty the same as in legacy TCP. As a result, MPQUIC experiences significantly fewer drops than (MP)TCP, especially in the initial phase.

For MPTCP, to provide compatibility with firewalls, the path controllers must follow the rules of singlepath TCP. The necessity of maintaining TCP-related flow control induces the separate TCP-like logic both at each *Path Controller* and at the *Main Controller*. MPQUIC is not obliged to obey such rules. From the point of view of firewalls, QUIC originated data are plain UDP datagrams. It allows for retransmission or even acknowledgment using a path different from the initially employed. The empowered version of SACK [26] is used for the acknowledgments, thus retrieving delayed or lost data do not impact dispatching other data.

All the controllers which are part of the protocols act independently. However, they interact with each other, with the network, and applications in a complex way, esp., if multiple parallel logical streams are transferred.

3 HoL Blocking Phenomenon

When transferring data from one peer to another, one assumes that the data belonging to different streams or conveyed using different paths do not interfere. Unfortunately, this reasoning is invalid. The data interact themselves through Head-of-Line blocking (HoL) phenomenon. There are two kinds of HoL blocking – path-related (PHoL, see Sect. 3.1) and stream-related (SHoL, see Sect. 3.2). Both of them increase protocol delay, especially when buffer-bloat (i.e., delay increase induced by filling buffers in interconnecting devices [27]) is present. Note, the impact of HoL depends on the activity of *Schedulers*, the *Path Controllers*, *Main Controller*, the applications, and the network state (Fig. 1).

3.1 Path-Related HoL Blocking

Let us consider packets conveying data from the same logical stream (Fig. 2). For some reason, the throughput on path 3 encounters throttling. Even though the data are transferred correctly using paths 1 (packets $x + 3$, $x + 4$) and 2, (packets $x + 5$, $x + 6$), the stream cannot be assembled coherently until segment $x + 2$ arrives. If it is, packets $x + 2$ up to $x + 6$ are instantaneously available to the user. However, if the receiver buffer is too short, the substreams conveyed using paths 1 and 2 must be suspended. A good quantitative measure of path-related HoL is the time of how long the given frame with data has been held in sender buffer waiting for the acknowledgment. This delay should be as low as the shortest SRTT (Smoothed Round Trip Time) read from path properties. The module responsible for the reduction of PHoL blocking is *Path Scheduler*.

3.2 Stream-Related HoL Blocking

Let us consider data conveying between the same peers, but logically different – e.g., file transfer together with interactive audio data and user keystrokes (Fig. 2). Bulk file transfer (most of the transferred data volume) has no special requirements. Audio data needs a constant bit rate, with the upper limit of delay, and keystrokes data are low

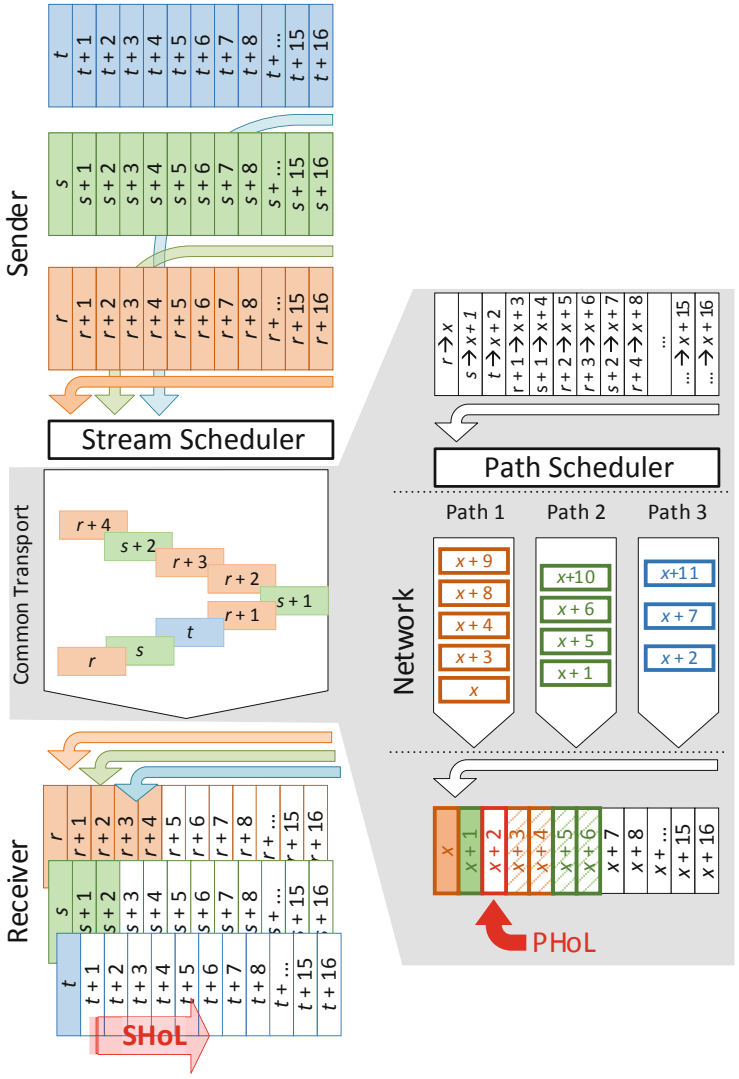

Fig. 2. Head of Line (HoL) blocking. *r*, *s*, *t* – number of frames for different logical streams. Stream-related HoL (SHoL) blocking: urgent stream *t* is blocked by streams *r, s*. Path-related HoL (PHoL) blocking: Single stream splits over three paths with different capabilities. *x* – transport offset. *x* is a mixture of streams *r*, *s*, *t*.

volume but urgent. Predominantly, only bulk data are present. If a user presses the key, the corresponding data are enqueued. They must wait until the previously sent low priority data are successfully delivered because the data order is fixed at the corresponding application activity.

Here, the quantitative measure of SHoL blocking phenomenon is the difference between the throughput of streams. For properly working transport, this value should be close to zero. The module responsible for the reduction of SHoL blocking is *Stream Scheduler*.

3.3 Protocol Efficiency

Due to the interaction of application logic, the network load, congestion control algorithms, schedulers activity (thus SHoL and PHoL blocking), the available paths are not fully used. The quantitative, popular measure of the protocol efficiency proposed in [28] is

$$\varphi = \begin{cases} \frac{G - \max(c_i)}{\sum_{i=1}^{n} c_i - \max(c_i)}, & \text{if } G \geq \max(c_i), \\ \frac{G - \max(c_i)}{\max(c_i)}, & \text{otherwise,} \end{cases} \tag{1}$$

where G is the throughput of multipath transmission, and c_i is the throughput of path i. $\varphi \in [-1, +1]$. If the measure reaches +1, then all the paths are fully engaged. If zero, then the multipath transmission has the same throughput as the fastest path. If $\varphi = -1$, the transmission stalls. Thus φ should never drop below zero, which indicates negative gain from multipath transport. Unfortunately, φ cannot be easily evaluated in a real environment in real-time. The throughput can be assessed *a posteriori*, as c_i = volume of data/transmission time. Thus c_i can be obtained only statistically. The paper is addressed to the assessment of dynamic properties of the transmission. Therefore (1) is redefined in the following way:

$$\varphi(k) = \begin{cases} \frac{G(k) - \max(c_i(k))}{\sum_{i=1}^{n} c_i(k) - \max(c_i(k))}, & \text{if } G(k) \geq \max(c_i(k)), \\ \frac{G(k) - \max(c_i(k))}{\max(c_i(k))}, & \text{otherwise,} \end{cases} \tag{2}$$

where k is the time instant of the subsequent frame departures. While evaluating (2), c_i (k) is approximated by c_i $(k) \approx$ inflight data(k)/SRTT(k) and averaged from multiple runs of the single path transfer. The term "inflight data" refers to the amount of data already sent but not acknowledged yet. The evolution of the multipath efficiency - $\varphi(k)$ – allows for identifying weaknesses of the control algorithms.

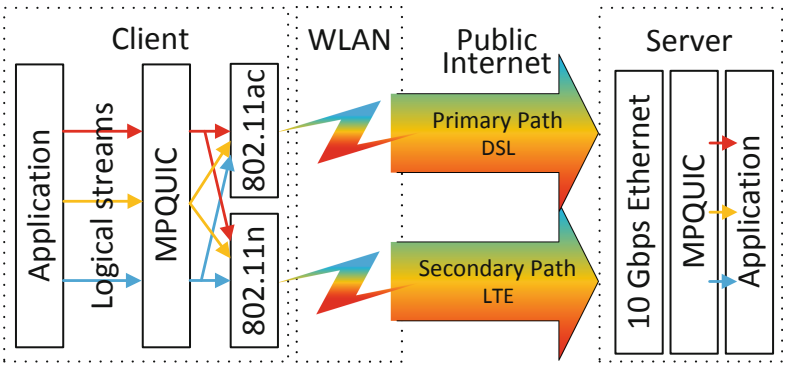

Fig. 3. Test setup. A multistream application employs MPQUIC protocol to send data using different paths. Different streams are marked with colors. The paths are established using different WLANs, and different providers of public Internet. The high performance infrastructure at last mile is shared among streams and paths.

4 Setup Design

In order to assess the dynamic properties of MPQUIC and compare them with QUIC, the test setup illustrated in Fig. 3 has been prepared. To make sure the application and operating system do not influence the transmission significantly, as a client, both inexpensive Raspberry Pi and Windows 10 based machines have been employed. The peer is run on a high-end server under Ubuntu Linux. Both client devices are equipped with two wireless communication interfaces. Next, data sent through the primary path encounter DSL as a likely bottleneck, and those sent through the secondary one – LTE. The segments sent through the DSL connection reach the destination in 8 hops, and those conveyed by LTE – in 11 hops. Initial RTTs during wee hours are 35–40 ms and 75–100 ms, respectively. The streams are generated using the dedicated application designed using DeConnick [9] code.

While the experiments involve real devices and public networks, it is impossible to receive precisely the same results in subsequent runs. Instead, the experiments have to be conducted $\sim$30 times. For the presented assessments, the manually selected, in authors' opinion – most typical and frequently observed experiments are selected.

5 Results

The MPQUIC stack includes modules that interfere with each other. The experiments are planned to investigate the impact of particular parts of the protocol on the overall transmission properties. Hence, three scenarios are selected.

5.1 Path Controllers

When the only single communication path is allowed, then *Path Scheduler* is idle, together with the *Main Controller*. Additionally, if a single stream is conveyed, the *Stream Scheduler* is idle, as well. Thus, only *Path Controllers* impact the transfer. In fact, such transmission is similar to an ordinary TCP one. The comparison of QUIC (Cubic) and MPQUIC (OLIA) is illustrated in Fig. 4. QUIC is more aggressive than MPQUIC. It implies higher path delays but a greater throughput, as well. QUIC is vulnerable to series of drops starting ~ 8 s (primary path) and ~ 10 s (secondary path). It significantly increases the protocol delay – 5, 6 times more than in the case of less aggressive MPQUIC. However, if the application is not time-sensitive, then the impact of the drops on throughput is acceptable.

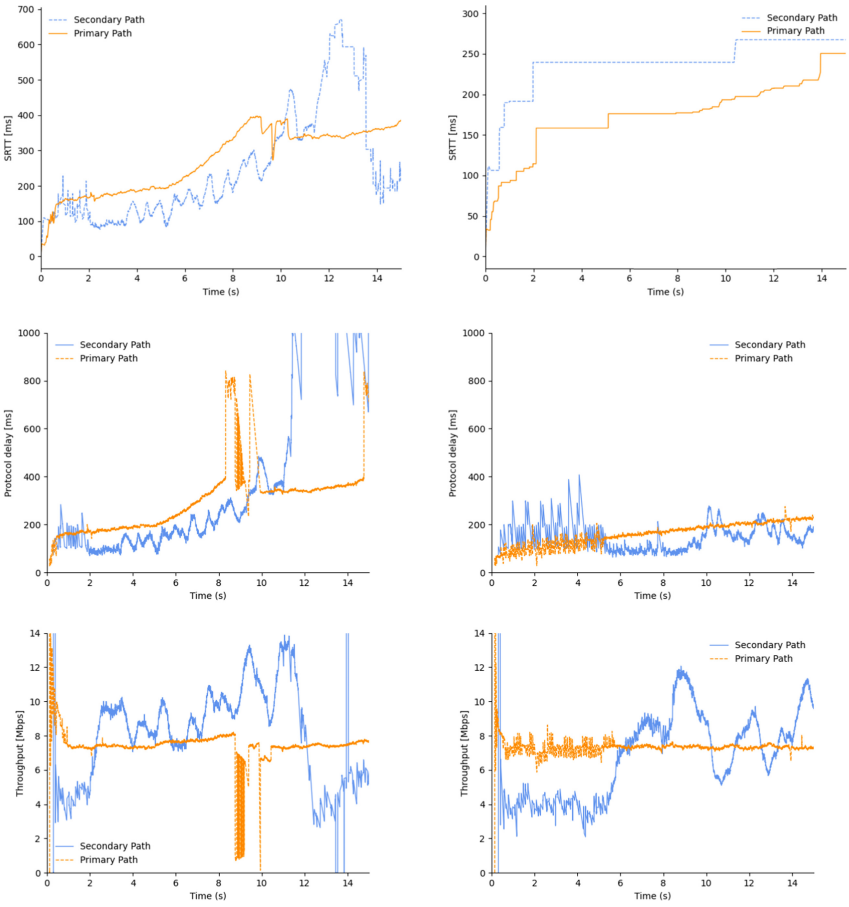

Fig. 4. Basic properties of the paths. Left: singlepath QUIC, Right: multipath QUIC where one path is active. Upper line: path perspective view (different vertical scale), middle line: protocol delay perspective view, bottom line: application perspective view.

Nevertheless, MPQUIC overperforms QUIC in terms of infrequent drops and short delays. Still, buffer bloat is present. It is exceptionally well visible on the primary path, where the constant throughput ($\sim$7.5 Mbps) accompanies the delay increase (35–220 ms). It is desirable to avoid this phenomenon, as in recently developed TCP congestion control algorithms (BBR [29], Wave [30]).

5.2 Master Controller and Path Scheduler

When more separate paths are available, then *Master Controller* (protocol fairness) and *Path Scheduler* (splitting process) impact the data transfer (Fig. 1). In the experiments, two uncoupled (without common bottleneck) paths are employed. Still, only one stream is conveyed, so *Stream Scheduler* does not influence the data transfer.

Figure 5 presents the typical evolution of the delay and throughput for the transmission. These measurements should be compared to those presented in Fig. 4. Contrary to MPTCP, MPQUIC sustains near-to-constant protocol delay. It proves that *Path Scheduler* works efficiently. *Master Controller* promptly boosts transfer. For the assessment of the dynamic of the multipath effectiveness $\varphi(k)$ (2), the average throughput for each path is evaluated and compared to the transfer speed obtained for the multipath transfer – Fig. 5, bottom, right chart. Usually, $\varphi(k)$ initially grows fast ($\varphi(k) \approx +1$), and after a few seconds, it slowly decreases. The direct reason is a sequence of drops. After such an incident, the efficiency remains at the level $\varphi(k) \approx +0.5$. Such a pattern is observed frequently. Sometimes, $\varphi(k)$ temporarily drops below zero, especially at rush hours.

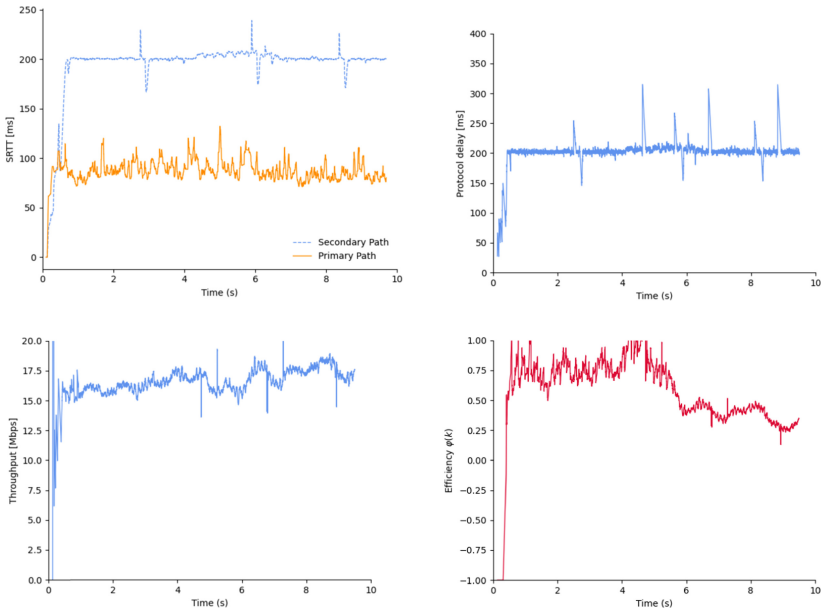

Fig. 5. Typical measurements of a single stream transmitted over different paths.

5.3 Stream Scheduler

The entire stack is evaluated in three subscenarios. In all of them, four streams are transmitted. Every new one starts two seconds after the previous one. From the point of view of previously started streams, the freshly added one is a disturbance. In the first subscenario, all the streams are transmitted continuously, with the best possible throughput. In the second one, stream 1 is a sequence of bursts (10 kB, every 100 ms). It illustrates a live media transmission, for example, voice or a game player activity. In the third subscenario, stream "1" is additionally privileged, i.e., the data from stream "1" are expedited as soon as they are available. The results of the measurements are presented in Fig. 6, each subscenario in subsequent rows.

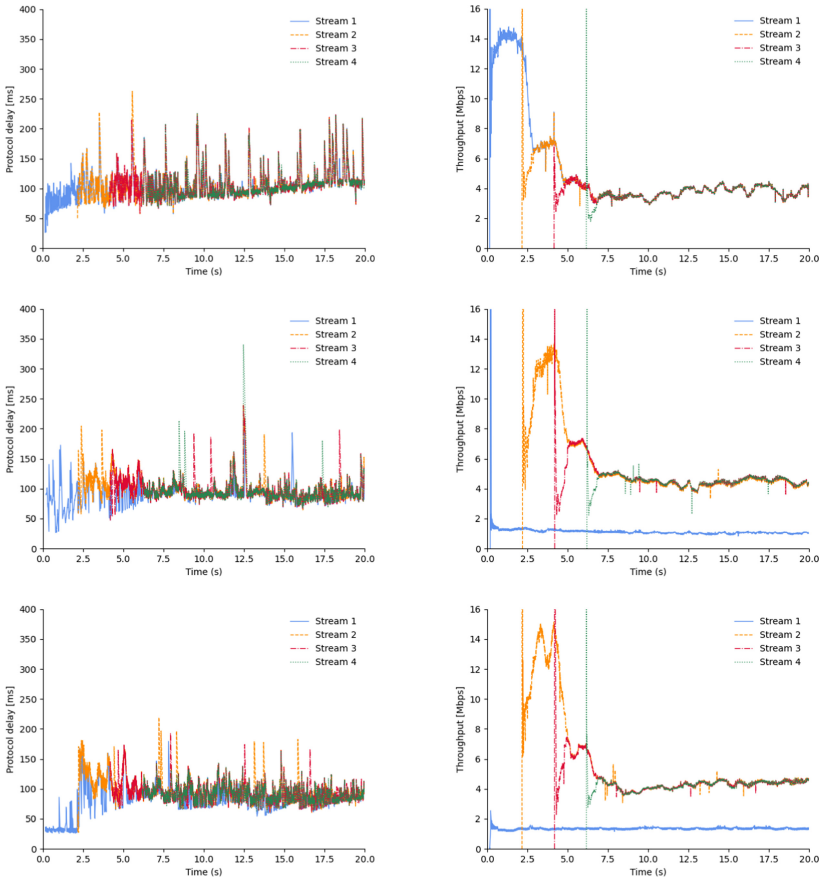

Fig. 6. Four streams transmitted using two paths scenario. Upper row: continuus transmission, middle row: short bursts for stream 1, bottom row: short bursts for priviledged stream 1.

In all cases, the disturbance introduced by a new stream transmission is quickly suppressed. However, the protocol does not obey the rules of priority correctly. Stream 1, despite is delayed like other streams, irrespectively from its priority, as an effect of buffer bloat vulnerability. If buffer bloat is present, *Stream Scheduler* has limited impact on transfer.

6 Result Assessment

The analysis of the protocol dynamics reveals problems that need to be solved:

- Buffer bloat vulnerability. The increase of path and protocol delay results in a protocol delay increase. If the incident lasts long enough, the retransmissions of dropped packets are necessary, which induces observed throughput diminishes (Fig. 7, top row). The buffer bloat phenomenon aggravates the stream recovery mechanisms. The reactive methods for constraining the protocol delay grow [31, 32] increases network load and energy dissipation and have limited effectiveness [33]. They should be replaced or accompanied by the tailored congestion control mechanisms that hold every path delay as low as possible.
- Decreasing efficiency. After a series of drops, the multipath transmission gain $\varphi(k)$ (2) decreases and does not restore. The problem is addressed to the *Master Controller* module.

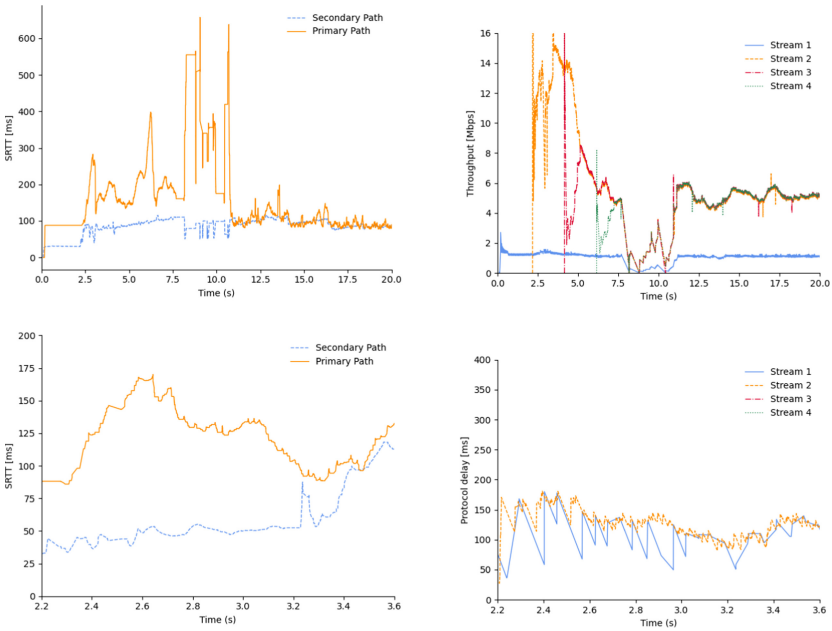

Fig. 7. Weaknesses of multipath transmission. Upper row: transmission throttling due to series of drops. Bottom row: oscillation of protocol delay due to path switching.

- *Stream Scheduler* and *Path Scheduler* are not cooperating with each other. For ordinary streams, the problem can be neglected. However, when MPQUIC is employed in real-time transmissions (e.g., unbuffered multimedia or interactive games), the delays should be constrained as high as possible. Even if *Stream Scheduler* privileges selected stream, *Path Scheduler* ignores this setting (Fig. 7, bottom row). Hence, stream delays oscillate among current path delay values. The efficient algorithm for assigning stream priority for *both* schedulers should be designed, not only for the *Stream Scheduler* [19].

7 Conclusions

The paper evaluates the dynamical properties of MPQUIC transmission in a natural network environment and using physical devices. MPQUIC protocol inherits the best advantages of its predecessors – multipath transmission, efficient congestion control, and streams decoupling. However, the protocol design is not closed, and the directions for further improvements are identified.

The necessity of implementing MPQUIC in an application is often considered a severe disadvantage compared to TCP or MPTCP. However, MPQUIC may be used in any operating system and device without restrictions introduced by firewalls, which sometimes tamper TCP options. Additionally, implementing new algorithms in the user space is more accessible than in the kernel space, as necessary for (MP)TCP. Therefore, the discovered weaknesses of MPQUIC should be solved promptly.

Acknowledgement. This work was supported in part by the National Science Centre, Poland, under Grant 2021/41/B/ST7/00108 "Robust control solutions for multi-channel networked flows".

References

1. Qadir, J., Ali, A., Yau, K.A., Sathiaseelan, A., Crowcroft, J.: Exploiting the power of multiplicity: a holistic survey of network-layer multipath. IEEE Commun. Surv. Tutor. **17** (4), 2176–2213 (2015). 4Q
2. Barreiros, M., Lundqvist, P.: QoS-Enabled Networks: Tools and Foundations. Wiley, Hoboken (2016)
3. Easley, R., Guo, H., Krämer, J.: From net neutrality to data neutrality. Inf. Syst. Res. **29**(2), 253–272 (2015)
4. Afanasyev, A., Tilley, N., Reiher, P., Kleinrock, L.: Host-to-host congestion control for TCP. IEEE Commun. Surv. Tutor. **12**(3), 304–342 (2010). 3Q
5. Ford, A., Raiciu, C., Handley, M., Bonaventure, O., Paasch, C.: TCP extensions for multipath operation with multiple addresses. RFC 8684 (2020)
6. Barré, S., Paasch, C., Bonaventure, O.: MultiPath TCP: From theory to practice. Technical report, Université Catholique de Louvain (2011)
7. Barré, S., Paasch, C.: MultiPath TCP – Linux kernel implementation. http://www.multipath-tcp.org
8. Langley, A., et al.: The QUIC transport protocol: design and internet-scale deployment. In: Proceedings of ACM SIGCOMM, New York, USA, pp. 183–196 (2017)

9. De Coninck, Q., Bonaventure, O.: Multipath QUIC: design and evaluation. In: Proceedings of 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2017), New York, NY, USA, pp. 160–166 (2017)
10. De Coninck, Q., Bonaventure, O.: Multiflow QUIC: a generic multipath transport protocol. IEEE Commun. Mag. **59**(5), 108–113 (2021)
11. Viernickel, T., Froemmgen, A., Rizk, A., Koldehofe, B., Steinmetz, R.: Multipath QUIC: a deployable multipath transport protocol. In: Proceedings of IEEE International Conference on Communications (ICC), Kansas City, MO, USA, pp. 1–7 (2018)
12. De Coninck, Q., Bonaventure, O.: MultipathTester: comparing MPTCP and MPQUIC in mobile environments. In: Proceedings of Network Traffic Measurement and Analysis Conference (TMA), Paris, France, pp. 221–226 (2019)
13. Vu, V.A., Walker, B.: On the latency of multipath-QUIC in real-time applications. In: 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 1–7 (2020)
14. Li, M., et al.: Multipath transmission for the internet: a survey. IEEE Commun. Surv. Tut **18** (4), 2887–2925 (2016). Q4
15. Yedugundla, K., et al.: Is multipath transport suitable for latency sensitive traffic? Comput. Netw. **105**, 1–21 (2016)
16. Tang, W., Fu, Y., Dong, P., Yang, W., Yang, B., Xiong, N.: A MPTCP scheduler combined with congestion control for short flow delivery in signal transmission. IEEE Access **7**, 116195–116206 (2019)
17. Stewart, R. (ed.): Stream control transmission protocol. RFC 4960 (2007)
18. Wang, J., Gao, Y., Xu, C.: A multipath QUIC scheduler for mobile HTTP/2. In: Proceedings of 3rd Asia-Pacific Workshop on Networking 2019 (APNet 2019), New York, NY, USA, pp. 43–49 (2019)
19. Shi, X., Wang, L., Zhang, F., Zhou, B., Liu, Z.: PStream: priority-based stream scheduling for heterogeneous paths in multipath-QUIC. In: Proceedings of 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, pp. 1–8 (2020)
20. Chiariotti, F., Deshpande, A.A., Giordani, M., Antonakoglou, K., Mahmoodi, T., Zanella, A.: QUIC-EST: a QUIC-enabled scheduling and transmission scheme to maximize VoI with correlated data flows. IEEE Comm. Mag. **59**(4), 30–36 (2021)
21. Khalili, R., Gast, N., Popovic, M., Le Boudec, J.-Y.: MPTCP is not Pareto-optimal: performance issues and a possible solution. IEEE/ACM Trans. Netw. **21**(5), 1651–1665 (2013)
22. Paasch, C., Ferlin, S., Alay, O., Bonaventure, O.: Experimental evaluation of multipath TCP schedulers. In: Proceedings of on ACM SIGCOMM CSWS, pp. 27–32, Chicago, USA (2014)
23. Morawski, M., Ignaciuk, P.: Energy-efficient scheduler for MPTCP data transfer with independent and coupled channels. Comp. Commun. **132**, 56–64 (2018)
24. Hurtig, P., Grinnemo, K., Brunstrom, A., Ferlin, S., Alay, Ö., Kuhn, N.: Low-latency scheduling in MPTCP. IEEE/ACM Trans. Netw. **27**(1), 302–315 (2019)
25. Ferlin, S., Alay, Ö., Mehani, O., Boreli, R.: BLEST: blocking estimation-based MPTCP scheduler for heterogeneous networks. In: Proceedings of IFIP Networking Conference Workshops, pp. 431–439, Vienna, Austria (2016)
26. Floyd, S., et al.: An extension to the selective acknowledgement (SACK) option for TCP. RFC 2883 (2000)
27. Gettys, J.: Bufferbloat: dark buffers in the internet. IEEE Internet Comput. **15**(3), 96 (2011)
28. Kimura, B.Y.L., Lima, D.C.S.F., Villas, L.A., Loureiro, A.A.F.: Interpath contention in multipath TCP disjoint paths. IEEE/ACM Trans. Netw. **27**(4), 1387–1400 (2019)

29. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: BBR: congestion-based congestion control. ACM Queue **14**(5), 20–53 (2016)
30. Abdelsalam, A., Luglio, M., Patriciello, N., Roseti, C., Zampognaro, F.: TCP wave over Linux: a disruptive alternative to the traditional TCP window approach. Comp. Netw. **184**, 1–14 (2021)
31. Ferlin, S., Kucera, S., Claussen, H., Alay, Ö.: MPTCP meets FEC: supporting latency-sensitive applications over heterogeneous networks. IEEE/ACM Trans. Netw. **26**(5), 2005–2018 (2018)
32. Michel, F., De Coninck, Q., Bonaventure, O.: QUIC-FEC: bringing the benefits of forward erasure correction to QUIC. In: Proceedings of IFIP Networking Conference, Warsaw, Poland, pp. 1–9 (2019)
33. Morawski, M., Ignaciuk, P.: A green multipath TCP framework for industrial internet of things applications. Comp. Netw **187**, 107831 (2021)