# The Packet Number Space Debate in Multipath QUIC

Quentin De Coninck*
UCLouvain, Belgium
quentin.deconinck@uclouvain.be

## ABSTRACT

With a standardization process that attracted much interest, QUIC can been seen as the next general-purpose transport protocol. Still, it does not provide true multipath support yet, missing some use cases that Multipath TCP addresses. To fill that gap, the IETF recently adopted a Multipath proposal merging several proposed designs. While it focuses on its core components, there still remains one major design issue: the amount of packet number spaces that should be used. This paper provides experimental results with two different Multipath QUIC implementations based on NS3 simulations to understand the impact of using one packet number space per path or a single packet number space for the whole connection. Our results show that using one packet number space per path makes Multipath QUIC more resilient to the receiver's heuristics to acknowledge packets and detect duplicates.

## CCS CONCEPTS

• **Networks** → **Transport protocols**; **Network protocol design**; **Network performance analysis**;

## KEYWORDS

Multipath QUIC, Packet Number Space, Protocol Design

## 1 INTRODUCTION

QUIC is a recently standardized protocol [19] aiming at providing the services of TLS/TCP (built-in encryption, reliable data transfer,...) with stream multiplexing atop UDP. Initially designed for HTTP/3 [2], QUIC sets itself up as the next general purpose transport protocol for the Internet and many extensions, such as the support for unreliable data transfer [35], have been proposed. Thanks to its flexibility, QUIC can serve many use cases [1, 17, 22] and enables rapid experiments with, e.g., congestion control algorithms [21, 33].

While QUIC supports probing new networks and switching to a different network, it only provides single-path data transmission. In particular, endhosts cannot simultaneously use different network paths to, e.g., aggregate their bandwidths or perform smooth network handover. Still, there is demand for such real multipath support for various use cases [29]. Multipath TCP [37] and CMT-SCTP [20] can now address them, such as mobility support for delay-sensitive applications [9], network handovers in high-speed trains [24] and hybrid access networks [3].

However, both Multipath TCP and CMT-SCTP faced deployment issues on the Internet [4, 13]. QUIC mitigates such network interference by design thanks to its built-in encryption, raising interest to introduce multipath support. The first attempts [8, 43] were mostly built on the design experience of Multipath TCP. However, these were based on an old version of QUIC [23] which differs from the standardized one. Further proposals were published by various authors [7, 10, 16, 27, 45], but these concurrent drafts actually slowed down reaching the consensus on one approach and some were considered too complex.

To advance the multipath work, all the previous drafts' authors worked together to make a common proposal [25] focusing on the core components that would suit their use cases. This multipath draft got adopted at IETF112 [28]. However, there remains a core design issue requiring consensus: how Multipath QUIC should number packets over paths (one shared number vs. per-path numbers).

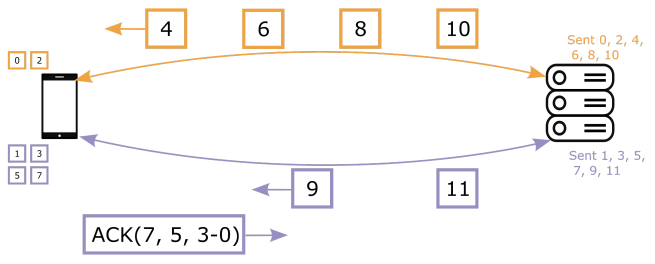
This paper aims at providing insights to the network research community in order to understand the implications of this design choice, not only for Multipath QUIC, but also for any multipath transport protocol. Indeed, Multipath TCP has several path's TCP sequence numbers with a global data one, while CMT-SCTP has a sequence number per data stream. Our evaluation reveals that while both designs may work for QUIC, using a single packet number space makes Multipath performance less predictable than using multiple ones because of the receiver's implementation.

The remaining of this paper is organized as follows. We start in Section 2 by describing the core components of Multipath QUIC and explaining the advantages and drawbacks of each packet number space design. Then, in Section 3, we evaluate these designs by considering two different implementations (`picoquic` [15] and `PQUIC` [11]) under a broad range of network scenarios using the NS3 simulator [39]. Finally, we discuss our results in Section 4.

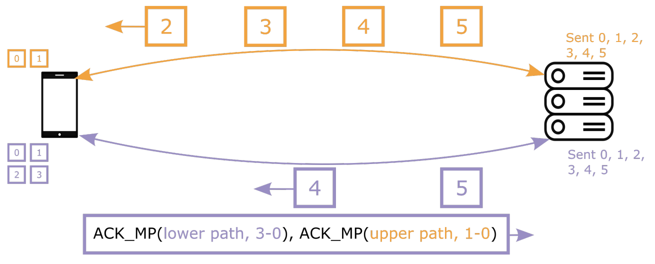
## 2 BRINGING MULTIPATH TO QUIC

Similarly to SCTP [41] having chunks, QUIC relies on *frames* as its core protocol units. For instance, the `STREAM` frame carries the application data along with an absolute offset that does not wrap-around, unlike TCP's sequence number and Multipath TCP's Data Sequence Number. QUIC conveys these frames inside fully encrypted packets. To set up such encryption, a QUIC connection relies on three different packet contexts: initial, handshake and application data. Each of these contexts has its dedicated cryptographic material and its own *packet number space* in which packets can be handled and acknowledged. Each packet number space is isolated from others. For instance, an application data packet can only be acknowledged by another application data packet. Upon QUIC handshake completion, all the packets use the application data context. QUIC packets numbers, included in the packet header, start at 0 and are monotonically increasing. A packet number cannot be reused in a given packet number space and a receiver must drop such duplicates. When a packet is lost, its frames can be retransmitted but in a packet with a greater packet number. While this prevents retransmission ambiguity as in, e.g., TCP, these exclusive packet numbers are also used to compute unique Authenticated Encryption with Associated Data (AEAD) nonces. Unlike (Multipath) TCP and (CMT-)SCTP that are based on data sequence numbers, QUIC acknowledges packet

*FNRS Post-doctoral Researcher.



**(a) Single packet number space case.**



**(b) Multiple packet number spaces case.**

**Figure 1: An example of a 12-packets data transfer where the server follows a round-robin scheduling strategy to send packets. Here, the client acknowledges received packets with a single packet sent on the lower path.**

numbers using the ACK frame. Compared to TCP where the number of selective acknowledgments [31] is often limited to 2-3 per packet, an ACK frame is constrained only by the size of the QUIC packet, limiting the amount of acknowledged packet number ranges to several hundreds entries. In addition, an end-host includes in ACK frames the delay between the reception of the largest packet number seen and the time the ACK frame was sent, allowing accurate RTT estimations by its peer. Implementations can then implement precise packet number- and time-based loss recovery.

A few fields of the QUIC packet header remain in clear-text. Among them, the *Destination Connection ID* enables the endhosts to map packets to QUIC connections. The Connection IDs of a QUIC session might change over time. The (encrypted) NEW_CONNECTION_ID frame enables endhosts to exchange additional Connection IDs, each being associated with a sequence number. This makes it hard for a third-party observer to correlate full connection activity to a single clear-text identifier. Relying on Connection IDs makes QUIC unbound to the 4-tuple ($\mathrm{IP}_{\mathrm{src}}$, $\mathrm{IP}_{\mathrm{dst}}$, $\mathrm{port}_{\mathrm{src}}$, $\mathrm{port}_{\mathrm{dst}}$). This *connection migration* support is one of the key features of QUIC. For instance, a connection initiated by a smartphone can move from a Wi-Fi network to a cellular one due to, e.g., user mobility. For that, a client willing to change the network path needs first to check that the endpoint is still reachable using the new 4-tuple. A specific process, called *path validation*, checks the viability of the new 4-tuple without affecting the ongoing transfer, i.e., neither STREAM nor ACK frames are sent on the network path under validation. To do so, both endhosts need to have an unused Destination Connection ID provided by their peer through NEW_CONNECTION_ID frames. If the path validation succeeds, the new 4-tuple is marked as validated. As soon as the client starts sending data (e.g. STREAM or ACK frames) over this new validated path, the server stops using the previous path and migrates the connection over the new 4-tuple.

While QUIC bundles such connection migration feature, it does not enable hosts to simultaneously use several network paths to send data. The adopted proposal [25] strives at providing multipath usage with as few changes as possible to the QUIC specification [19]. It focuses on the core building blocks: negotiating multipath, initiating new paths and numbering packets.

*Multipath Negotiation.* Endhosts negotiate the multipath extension during the connection handshake using a QUIC transport parameter. If both the client and the server advertise common support for a multipath design, then the connection uses the multipath extension upon handshake completion.

*Initiating New Paths.* Multipath QUIC builds on the path validation process to initiate new paths. The main addition to the single-path QUIC is that with the multipath extension, several validated paths can be simultaneously used to send data packets. Note that since an endhost must use distinct Destination Connection IDs per path, its peer controls the maximum number of paths that can be opened by choosing when to send NEW_CONNECTION_ID frames.

*Numbering Packets.* In single-path QUIC, all post-handshake packets use the application data packet number space. The current multipath proposal enables implementations to experiment with either single packet number space or multiple packet number spaces. We first describe the single packet number space (SPNS) design. Then, we discuss the multiple packet number spaces (MPNS) one.

### 2.1 Single Packet Number Space (SPNS)

This design lets endhosts spread packets over different paths while keeping the regular QUIC application data packet number space. This means that a packet with number $N$ can be sent over path $A$ while a packet with number $N + 1$ can be transmitted on path $B$. To illustrate this situation, Figure 1a represents a server sending 12 packets to a client in a round-robin fashion. Here, the server sends all even packet numbers on the upper path and all the odd ones on the lower path. The packets are acknowledged with the regular ACK frame. The receiver cannot know in advance on which path it will see a given packet number. When paths do not exhibit the same performance, the receiver is likely to observe out-of-order packet numbers. In the example, the lower path is faster than the upper one. The client received two packets on the upper path (0, 2) and four on the lower path (1, 3, 5, 7). It acknowledges their reception by a sending an ACK frame having three ranges: 7, 5 and from 3 to 0, although there is no reordering within each individual path. Here, the ACK frame acknowledges packets from both paths.

This multipath-triggered out-of-order packet number reception might raise several concerns related to the receiver's implementation. Keeping track of acknowledged packets is required to avoid accepting duplicate packets, but this requires maintaining state at receiver's side. Endhosts typically want to control the size of this state to avoid specific reception patterns to increase it too much. Many implementations keep track of an *ACK horizon*, i.e., incoming packets having a number lower than this horizon are considered as duplicate and thus discarded. Some implementations [12] stop tracking numbers they sent in an ACK frame that got acknowledged. Others [32] limit the number of ranges they track, considering packets older than the oldest range as duplicate. Some [6, 36] implement a fixed-size (128 packet long) ACK horizon relative to the largest packet number seen and track duplicates with a bitmask, similar to the anti-replay mechanisms of several encrypted protocols [34, 38, 46]. picoquic integrates a delay-based ACK horizon keeping ranges for 1 second when multipath is enabled. These heuristics are effective in single-path scenarios to stop tracking (probably) lost packets. However, in multipath settings each of these heuristics might end up declaring a packet coming from a slow path as duplicate (and thus drop it) if another much faster path is used. In such cases, it would make it impossible to benefit from multipath. In addition, since the ACK frame only includes the delay relative to the largest packet number received, precise RTT estimates on slow paths could be rare. Finally, the packet number-based loss recovery algorithm must be adapted to be relative to a path instead of the absolute numbers over the whole connection. Otherwise, reception of new ACK ranges by the sender may trigger many spurious retransmissions.

## 2.2 Multiple Packet Number Spaces (MPNS)

With this design, each path has an associated packet number space. As depicted in Figure 1b, subsequent packets over a given path use consecutive packet numbers. Because packet numbers are now path-dependent, an augmented version of the ACK frame, called the ACK_MP frame, includes the numerical identifier of the Connection ID to which the acknowledged packet numbers refer to. The path identification is thus made using Connection IDs. Compared to the SPNS design, ranges in ACK_MP frames are not affected by multipath out-of-order delivery. This should make the transfer's performance less dependent of the receiver's strategy. Still, it remains possible to acknowledge packets from one path on another one by sending the associated ACK_MP frame, as depicted in Figure 1b. Furthermore, the delay included in ACK_MP frames relates to the largest packet number seen on the given path, making precise latency estimates possible even on slow paths, although the computed latency here depends on the path chosen by the receiver.

Unlike the SPNS design, the MPNS one requires the addition of new mechanisms. In addition to the new ACK_MP frame, it requires adapting the computation of the AEAD nonce by including the path identifier to avoid reusing a same nonce on different paths. Furthermore, this design is not compatible with some specific QUIC features such as zero-length Connection IDs.

## 3 THE IMPACT OF MULTIPATH REORDERING

As discussed in the previous section, the theoretical impact of packets received in a different order than the one they were sent differs depending on the multipath design used, especially when paths have very different characteristics. To experimentally assess this out-of-order delivery impact on Multipath QUIC designs, we explore a large set of network scenarios within the NS3 environment [39] using the direct code execution framework [42]. Compared to emulation, this setup enables fully reproducible and stable results with very low variability while still using actual implementations.

We focus on two different open-source implementations of Multipath QUIC. picoquic [15] supports both SPNS and MPNS designs following the adopted proposal [25]. Regardless of the design used, it integrates the same per-path loss recovery and packet scheduler selecting the first path whose congestion window is open. PQUIC [11] has a multipath plugin implementing an earlier proposal [7] relying on multiple packet number spaces. This is a fork of an earlier picoquic version. Like picoquic, it integrates a per-path loss recovery and a round-robin based packet scheduler. Note that both implementations limit to 33 the maximum number of ACK ranges (AR) contained in a single ACK(_MP) frame. We start our experiments with picoquic and PQUIC versions that prioritize writing the latest ranges in ACK(_MP) frames.

We explore many network situations following an experimental design approach using the WSP algorithm [40] enabling us to broadly cover the parameter space with 95 runs. All our experiments consist in a 50 MB download initiated by a GET request over a single stream. Relying on such a large transfer and a large initial receive buffer of 2 MB enables us to limit the impact of the sender's packet scheduler, as the transfer will be ACK-clocked. The client initiates the usage of all the available paths upon handshake completion. We define the transfer time as the delay between the first QUIC packet sent by the client and the last QUIC packet received by the client closing the connection. In addition, we generate QLOG files [30] at client and server sides to get a full view of the connection and the internal state of the implementations.

We first experiment with 2-path network scenarios where both paths share the same characteristics. Then, we explore 2-path situations where paths exhibit heterogeneous performances in terms of bandwidth and delay. Finally, we extend our heterogeneous scenarios to 3-path networks.

**Table 1: Parameter space for the 95 homogeneous runs.**

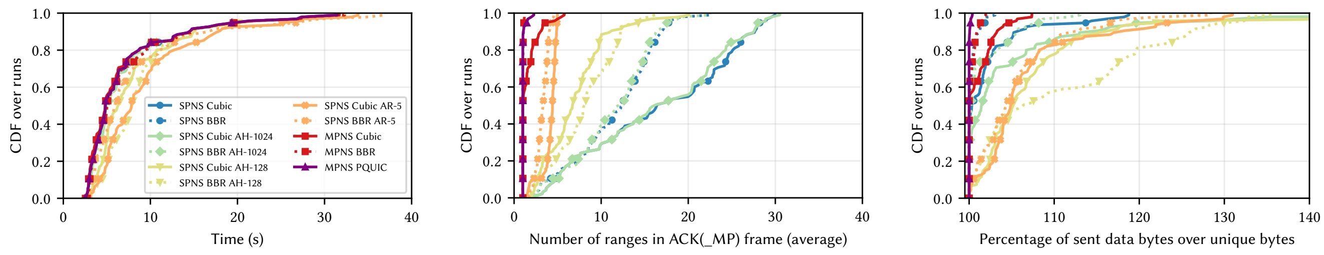
| Factor | Min | Max |
|---|---|---|
| Bandwidth [Mbps] | 2.5 | 100 |
| RTT [ms] | 5 | 100 |

## 3.1 Homogeneous 2-Path Experiments

When all the network paths provide the same performance, i.e., same delay and bandwidth, there should not be much packet reordering at receiver's side. Therefore, there should not be much performance difference between packet number space designs. To assess this intuition, we consider the parameter space depicted in Table 1. In this paper, unless explicitly mentioned otherwise, we only consider loss-less networks and the buffer size of the bottleneck router is always set to 1.5 times the bandwidth-delay product. Note that actual packet losses may still occur due to bottleneck router's buffer overflow.

Figure 2a shows the completion times for the 50 MB transfer. As picoquic supports both packet number space designs and two different congestion control schemes (Cubic and BBR), we evaluate each combination. PQUIC only supports MPNS design with Cubic. The transfer times are mostly dependent of the paths' bandwidth that differs between runs. Still, we do not observe much performance difference between SPNS and MPNS, which is expected.

However, having homogeneous paths does not prevent the receiver from observing reordering. We extract the ranges advertised in the ACK (for SPNS) and ACK_MP frames (for MPNS) sent by the



**(a) Transfer time.** **(b) Number of ranges advertised by the client.** **(c) Data retransmitted by the server.**

**Figure 2: Homogeneous experiments. The legend is common to all the figures.**

client. Figure 2b shows the arithmetic average size of the ACK(_MP) ranges over runs. With multiple packet number spaces, ACK_MP frames often show a single range, as no reordering occurs within a path. Still, a few packet losses due to buffer overflow might occur, leading to a few MPNS runs where the average range size is greater than 1. With SPNS, ACK frames can carry many ranges, even if there is no buffer loss. Indeed, it can happen that the server pushes slightly more on a path than the other, hence adding queuing delay on that path, causing an initial multipath reordering. Afterwards, the reception of ACK frames with several ranges lets the server send a burst of packets on the less-loaded path. picoquic integrates pacing at the sender to limit this effect.

Yet, the high average range size of SPNS raises concerns when the receiver implements stricter heuristics like limiting the number of ranges it wants to maintain [32] or keeping a fixed-size ACK horizon [6, 36]. If there is reordering involving many packets, the client may never acknowledge a given packet, even if it was received. To implement the limited ACK range (AR) situation, we limit the maximum number of ranges that an ACK(_MP) frame can contain to 5. This value might be considered for low-power devices such as in IoT. We also implemented the fixed ACK horizon (AH) strategy with two different values: 128 (some implementations' default [6, 36]) and 1024. Figure 2a shows that such strategies hinders the performance of the transfer. To explain this performance drop, we consider the STREAM frames the server sends. Based on the data offset and length fields, we compute the number of retransmitted data bytes and make it relative to the transfer size (50 MB). Figure 2c indicates that there are indeed more data retransmission with SPNS, and especially when the number of ranges that ACK frames can convey is limited or when the fixed ACK horizon makes the receiver drops low packet numbers. Even in homogeneous networks, such heuristics make the receiver unable to acknowledge some packets, leading to spurious retransmissions, decreased sending rate and lower overall performance. Note that the choice of the receiver's heuristic also affects the performance of the congestion control scheme.

## 3.2 Heterogeneous 2-Path Experiments

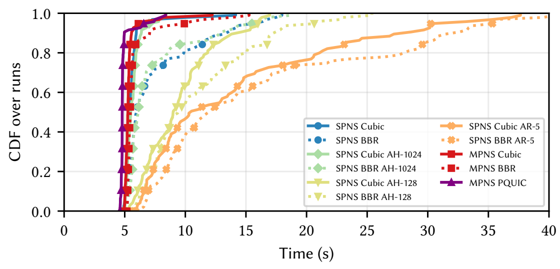
The previous experiments considered an "idealistic" case where all network paths share the same characteristics. This situation rarely happens in practice with, e.g., Wi-Fi/LTE or terrestrial/satellite [44]. To investigate situations where paths have different properties, we consider the parameter space shown in Table 2. We split the bandwidth and RTT between paths such as their sum is always the same. For instance, if bandwidth balance is 0.9 and RTT balance 0.1, then the first path has 90 Mbps 20 ms RTT and the second path 10 Mbps 180 ms RTT. As the theoretical bandwidth remains the same across all runs, this enables easier comparison in terms of transfer time.

**Table 2: Parameter space for the 95 heterogeneous 2-path network scenarios.**

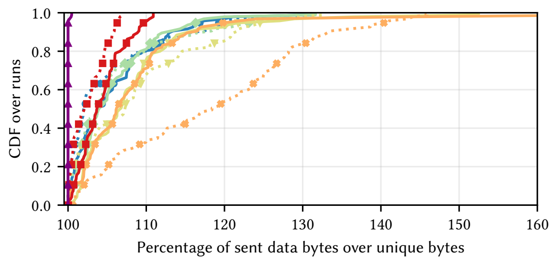
| Factor | Value |
|---|---|
| Total Bandwidth [Mbps] | 100 |
| Total RTT [ms] | 200 |
| Bandwidth Balance | [0.1; 0.9] |
| RTT Balance | [0.1; 0.9] |

When using its default settings and considering Cubic, picoquic with SPNS keeps performance close to picoquic with MPNS and PQUIC, as depicted in Figure 3a. However, as in homogeneous experiments, picoquic with SPNS triggers ACK frames advertising many ranges, sometimes hitting the receiver's ACK range limit. If the receiver uses stricter heuristics with lower ACK range limits or with a fixed ACK horizon, the transfer performance degrades because of spurious loss detection and increased data retransmission (Figure 3b). In particular, with SPNS picoquic using Cubic whose receiver does not advertise more than 5 ranges in ACK frames, there is a run where there are more than 70 % of the data that get retransmitted, some piece of data being retransmitted up to 8 times. Furthermore, the performances of picoquic transfers using BBR are worse than the ones using Cubic.

We contacted the picoquic's implementer to report these results. He was aware of the two aforementioned issues. First, for the BBR performance, it was related to the ACK frame scheduling by the data receiver. Under some circumstances, a specific range was always sent on the slow path. Then, a later range arrived first on the fast path, affecting the path's RTT estimate and loss detection algorithm based on RACK [5]. The implemented fix consists in duplicating ACK(_MP) frames on both paths. Second, regarding limited ACK ranges, the implementation's behavior was to acknowledge the most recent ranges first. In case of large packet reordering, it may happen that packets were either lately or never acknowledged, leading to spurious data retransmissions. To address that, picoquic now includes an heuristic placing the oldest ranges first, knowing the maximum number of ranges it wants to advertise.

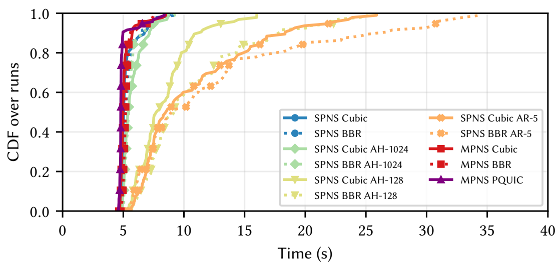


**(a) Transfer time.**

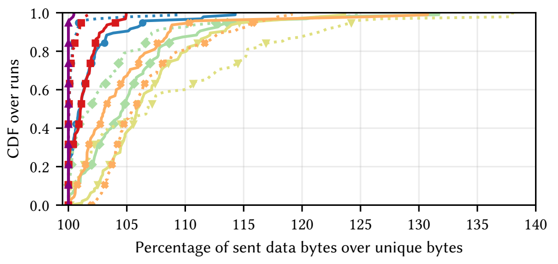


**(b) Data retransmitted by the server.**

**Figure 3: Heterogeneous 2-path experiments with original `picoquic`. The legend is common to all the figures.**



**(a) Transfer time.**



**(b) Data retransmitted by the server.**

**Figure 4: Heterogeneous 2-path experiments with fixed `picoquic`. The legend is common to all the figures.**

In the remaining experiments, we consider a version of `picoquic` integrating these fixes. We rerun the previous 2-path heterogeneous experiments and present the results in Figure 4. To better highlight the impacts of those fixes, Figure 5 shows the ratio of the transfer time between the original `picoquic` and the fixed one on the same run. On the one hand, the performance of `picoquic` using BBR considerably improved. Indeed, BBR is very sensitive to the path's latency, and duplicating `ACK` frames on both paths makes these estimates more stable. This ACK duplication also benefits to Cubic runs and the MPNS variant. On the other hand, changing the range selection strategy provided mixed results. While acknowledging lowest ranges first brings benefits in some network scenarios, it provides worse results in others. Depending on the receiver's heuristic, it can trigger sub-optimal decisions at sender's side and increases the amount of retransmitted data.

Interestingly, some runs of `picoquic` with MPNS design using Cubic show data retransmissions and `ACK_MP` frames with several ACK ranges while `PQUIC` does not (like `picoquic` BBR). Two elements might explain this result. First, the implementations of Cubic are slightly different, and `picoquic`'s one uses the estimated path's RTT to determine whether it should exit the slow-start phase or not. This might make the `picoquic`'s Cubic slightly more aggressive than `PQUIC`'s one. Second, as depicted in Figure 6, the `PQUIC` receiver acknowledges data much more aggressively than the `picoquic` one. Indeed, the `PQUIC` receiver sends an `ACK_MP` frame for both paths as soon as two new packets arrive for any given path. Furthermore, it seems that recent `picoquic` tends to batch the reception of new packets much more than `PQUIC`, hence decreasing the number of generated packets by the client. Still for `picoquic`, while its initial limit is also set to 2, it uses the `ACK_FREQUENCY`

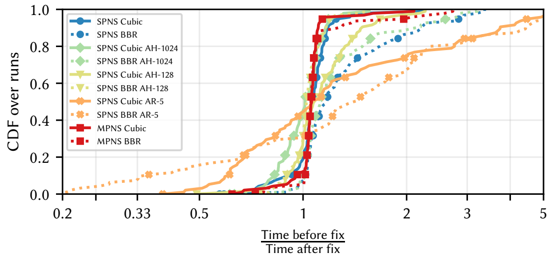


**Figure 5: A ratio greater than 1 means that fixed `picoquic` is faster than original `picoquic`. Note the logarithmic x scale.**

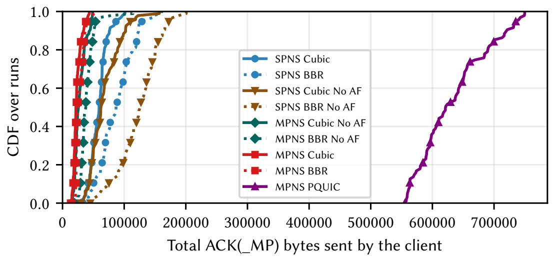


**Figure 6: Heterogeneous 2-path experiments, total number of acknowledgment frames' bytes. No AF = No ACK Frequency.**

frame [18] by default and requests its peer to allow delaying ACK frame sending by at most 25 ms instead of the default 1 ms. It can

**Table 3: Parameter space for the 95 heterogeneous 3-path network scenarios.**

| Factor | Value |
|---|---|
| Total Bandwidth [Mbps] | 100 |
| Total RTT [ms] | 300 |
| Bandwidth Path Weight | [0.1; 0.9] |
| RTT Path Weight | [0.1; 0.9] |

then wait a few milliseconds between subsequent packets containing ACK(_MP) frames. Disabling the ACK_FREQUENCY negotiation to keep the maximum ACK delay to 1 ms slightly increases the transfer performance by a few hundreds milliseconds (figure not shown due to space limit). Also note that while the minimum size of an ACK_MP frame is larger than a regular ACK one, picoquic with MPNS manages to send less acknowledgment-related bytes than picoquic with SPNS. Smaller ranges explain this result.

## 3.3 Heterogeneous 3-Path Experiments

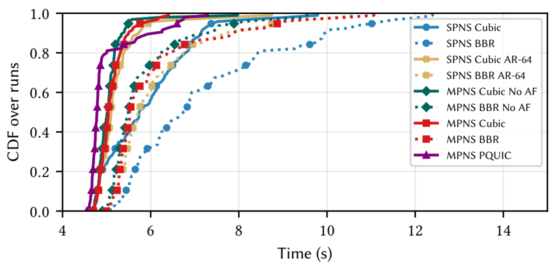
When considering multipath, one usually starts with two network paths. Still, there are cases where more than two network paths are simultaneously available, such as dual-stacked IPv4/IPv6 Wi-Fi and LTE. To cover them, we consider 3-path scenarios covering the parameter space depicted in Table 3. Unlike in the previous subsection that explored a 2-dimensional space, each path has a weight for the budget bandwidth and RTT, leading to 6 varying parameters (note that the sum of the weights might differ from 1). As an example with the RTT, if the first path has a weight of 0.5, the second path 0.25 and the third path 0.75, the RTT of each path will respectively be 100 ms, 50 ms and 150 ms.

In the remaining experiments, we keep PQUIC and the fixed version of picoquic. Figure 7 shows that when three heterogeneous paths are simultaneously used, 33 ranges inside ACK frames are often not sufficient with SPNS to provide a full view of the received packets to the sender. Indeed, in nearly all our runs, at least 30% of all the ACK frames sent by the client hit the implementation limit. This lack of information triggers more spurious loss detection events and retransmissions (Figure 7b), leading to lower performance compared to multiple packet number spaces (Figure 7a).

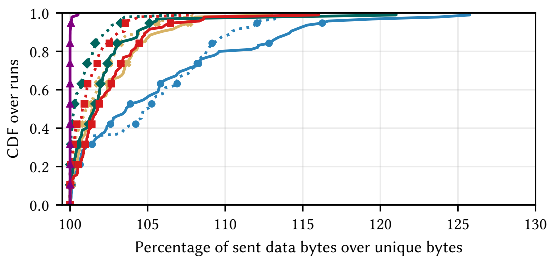
In such networks, it is still possible to make the SPNS design work as well as the MPNS one by increasing or removing the maximum number of ACK ranges that an implementation is willing to support (here shown for 64 ranges). However, when using SPNS, this makes multipath more dependent on the receiver's implementation than with MPNS. Note that increasing the acknowledgment rate of picoquic (by disabling ACK_FREQUENCY support to keep a max ACK delay of 1 ms instead of 25 ms) enables it to get results closer to the PQUIC's one while keeping the same trend between the different designs.

## 4 DISCUSSION AND NEXT STEPS

In this paper, we evaluated the impact of the amount of packet number spaces on a Multipath QUIC bulk transfer. While this scenario does not cover all the possible usages of multipath, it assesses its bandwidth aggregation feature. While both designs can address



**(a) Transfer time.**



**(b) Data retransmitted by the server.**

**Figure 7: Heterogeneous 3-path experiments with fixed picoquic. The legend is common to all figures.**

this need, our experiments pointed out that when using the single packet number space design, the performance of Multipath QUIC is more dependent on the receiver's heuristics to acknowledge packets and detect duplicates.

Huitema [14] mentioned that the final decision of using one or several packet number spaces will be a trade-off between implementation complexity and design flexibility. However, the multipath transfer performance remains a critical factor. The behavior of a multipath connection should, by design, only depend on the data sender's decisions and the network conditions. Use cases such as 3GPP's ATSSS [1] will require further sender control such as path prioritization or specific scheduling using different networks. If the multipath feature relies on a single packet number space, it lets the receiver add variability by design, especially when the path's heterogeneity is high — see, e.g., the impact of a fixed-length ACK horizon on BBR. Such uncertainty, out of sender's control, might make some use cases harder to reach, if not impossible. Although the support of per-path packet number spaces requires new mechanisms (ACK_MP frame, nonce computation adaptation,...), we argue that these remain simple to introduce into path-aware implementations. Given that it prevents many performance issues by design, the latest draft [26] consider per-path packet number spaces as the default design, keeping the one packet number space solution as an optional feature only to handle zero-length Connection IDs.

## ACKNOWLEDGMENTS

The author thanks Maxime Piraux for his support for the NS3 tooling.

## A ARTIFACTS

We provide two kinds of artifacts. The first one relates to the simulation results and Jupyter notebook used to generate the graphs of this paper (including not shown ones). These are available using the following URL: https://doi.org/10.5281/zenodo.6323195.

The second one consists in the NS3 setup in which we run `PQUIC` and the modified `picoquic` implementations. All the software, provided as git submodules, as well as instructions are available at https://github.com/qdeconinck/ccr2022-artifacts.

## REFERENCES

[1] 2021. Study on access traffic steering, switching and splitting support in the 5G system architecture; Phase 2 (Release 17). Standard v17.0.0. (March 2021).

[2] Mike Bishop. 2022. HTTP/3. RFC 9114. (June 2022). https://doi.org/10.17487/RFC9114

[3] Olivier Bonaventure and S Seo. 2016. Multipath TCP deployments. *IETF Journal* 12, 2 (2016), 24–27.

[4] Łukasz Budzisz, Johan Garcia, Anna Brunstrom, and Ramon Ferrus. 2012. A taxonomy and survey of SCTP research. *ACM Computing Surveys (CSUR)* 44, 4 (2012), 1–36.

[5] Yuchung Cheng, Neal Cardwell, Nandita Dukkipati, and Priyaranjan Jha. 2021. The RACK-TLP Loss Detection Algorithm for TCP. RFC 8985. (Feb. 2021). https://doi.org/10.17487/RFC8985

[6] Cloudflare. 2022. quiche: Savoury implementation of the QUIC transport protocol and HTTP/3. (2022). https://github.com/cloudflare/quiche

[7] Quentin De Coninck and Olivier Bonaventure. 2021. *Multipath Extensions for QUIC (MP-QUIC)*. Internet-Draft draft-deconinck-quic-multipath-07. IETF. https://datatracker.ietf.org/doc/html/draft-deconinck-quic-multipath-07

[8] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath quic: Design and evaluation. In *Proceedings of the 13th international conference on emerging networking experiments and technologies*. 160–166.

[9] Quentin De Coninck and Olivier Bonaventure. 2018. Tuning multipath TCP for interactive applications on smartphones. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 1–9.

[10] Quentin De Coninck and Olivier Bonaventure. 2021. Multiflow QUIC: A Generic Multipath Transport Protocol. *IEEE Communications Magazine* 59, 5 (2021), 108–113.

[11] Quentin De Coninck, François Michel, Maxime Piraux, Florentin Rochet, Thomas Given-Wilson, Axel Legay, Olivier Pereira, and Olivier Bonaventure. 2019. Pluginizing quic. In *Proceedings of the ACM Special Interest Group on Data Communication*. 59–74.

[12] Google. 2022. QUICHE. (2022). https://github.com/google/quiche

[13] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. 2011. Is it still possible to extend TCP?. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. 181–194.

[14] Christian Huitema. 2021. How many packet number spaces for QUIC Multipath? (2021). https://huitema.wordpress.com/2021/02/14/how-many-packet-number-spaces-for-quic-multipath/.

[15] Christian Huitema. 2021. picoquic. (2021). https://github.com/private-octopus/picoquic.

[16] Christian Huitema. 2021. *QUIC Multipath Negotiation Option*. Internet-Draft draft-huitema-quic-mpath-option-01. IETF. https://datatracker.ietf.org/doc/html/draft-huitema-quic-mpath-option-01

[17] Christian Huitema, Sara Dickinson, and Allison Mankin. 2022. DNS over Dedicated QUIC Connections. RFC 9250. (May 2022). https://doi.org/10.17487/RFC9250

[18] Jana Iyengar and Ian Swett. 2021. *QUIC Acknowledgement Frequency*. Internet-Draft draft-ietf-quic-ack-frequency-01. IETF. https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-01 Work in Progress.

[19] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. (May 2021). https://doi.org/10.17487/RFC9000

[20] Janardhan R Iyengar, Paul D Amer, and Randall Stewart. 2006. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on networking* 14, 5 (2006), 951–964.

[21] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols. In *Proceedings of the 2017 Internet Measurement Conference*. 290–303.

[22] Mike Kosek, Tanya Shreedhar, and Vaibhav Bajpai. 2021. Beyond QUIC v1: A First Look at Recent Transport Layer IETF Standardization Efforts. *IEEE Communications Magazine* 59, 4 (2021), 24–29.

[23] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*. 183–196.

[24] Li Li, Ke Xu, Tong Li, Kai Zheng, Chunyi Peng, Dan Wang, Xiangxiang Wang, Meng Shen, and Rashid Mijumbi. 2018. A measurement study on multi-path TCP with multiple cellular carriers on high speed rails. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 161–175.

[25] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. 2022. *Multipath Extension for QUIC*. Internet-Draft draft-ietf-quic-multipath-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-multipath-00 Work in Progress.

[26] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. 2022. *Multipath Extension for QUIC*. Internet-Draft draft-ietf-quic-multipath-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/02/ Work in Progress.

[27] Yanmei Liu, Yunfei Ma, Christian Huitema, Qing An, and Zhenyu Li. 2021. *Multipath Extension for QUIC*. Internet-Draft draft-liu-multipath-quic-04. IETF. https://datatracker.ietf.org/doc/html/draft-liu-multipath-quic-04

[28] Robin Marx, Jonathan Hoyland, and Watson Ladd. 2021. IETF-112 QUIC WG Meeting Minutes. (2021). https://github.com/quicwg/wg-materials/blob/main/ietf112/minutes.md.

[29] Robin Marx and Eric Kinnear. 2020. QUIC 2020-10-22 Interim Meeting Minutes. (2020). https://github.com/quicwg/wg-materials/blob/main/interim-20-10/minutes.md.

[30] Robin Marx, Maxime Piraux, Peter Quax, and Wim Lamotte. 2020. Debugging QUIC and HTTP/3 with qlog and qvis. In *Proceedings of the Applied Networking Research Workshop*. 58–66.

[31] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. 1996. RFC2018: TCP selective acknowledgement options. (1996).

[32] Mozilla. 2022. Neqo, an Implementation of QUIC written in Rust. (2022). https://github.com/mozilla/neqo

[33] Akshay Narayan, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. 2018. Restructuring endpoint congestion control. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 30–43.

[34] Karl Norrman, David McGrew, Mats Naslund, Elisabetta Carrara, and Mark Baugher. 2004. The Secure Real-time Transport Protocol (SRTP). RFC 3711. (March 2004). https://doi.org/10.17487/RFC3711

[35] Tommy Pauly, Eric Kinnear, and David Schinazi. 2022. An Unreliable Datagram Extension to QUIC. RFC 9221. (March 2022). https://doi.org/10.17487/RFC9221

[36] quinn rs. [n. d.]. quinn: Async-friendly QUIC implementation in Rust. ([n. d.]). https://github.com/quinn-rs/quinn

[37] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How hard can it be? designing and implementing a deployable multipath TCP. In *9th USENIX symposium on networked systems design and implementation (NSDI 12)*. 399–412.

[38] Eric Rescorla and Nagendra Modadugu. 2012. Datagram Transport Layer Security Version 1.2. RFC 6347. (Jan. 2012). https://doi.org/10.17487/RFC6347

[39] George F Riley and Thomas R Henderson. 2010. The ns-3 network simulator. In *Modeling and tools for network simulation*. Springer, 15–34.

[40] J Santiago, M Claeys-Bruno, and M Sergent. 2012. Construction of space-filling designs using WSP algorithm for high dimensional spaces. *Chemometrics and Intelligent Laboratory Systems* 113 (2012), 26–31.

[41] Randall R. Stewart, Michael Tüxen, and Karen Nielsen. 2022. Stream Control Transmission Protocol. RFC 9260. (June 2022). https://doi.org/10.17487/RFC9260

[42] Hajime Tazaki, Frédéric Uarbani, Emilio Mancini, Mathieu Lacage, Daniel Camara, Thierry Turletti, and Walid Dabbous. 2013. Direct code execution: Revisiting library os architecture for reproducible network experiments. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. 217–228.

[43] Tobias Viernickel, Alexander Froemmgen, Amr Rizk, Boris Koldehofe, and Ralf Steinmetz. 2018. Multipath QUIC: A deployable multipath transport protocol. In *2018 IEEE International Conference on Communications (ICC)*. IEEE, 1–7.

[44] Bo Zhang, TS Eugene Ng, Animesh Nandi, Rudolf Riedi, Peter Druschel, and Guohui Wang. 2006. Measurement based analysis, modeling, and synthesis of the internet delay space. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. 85–98.

[45] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, et al. 2021. XLINK: QoE-driven multi-path QUIC transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 418–432.

[46] Tina Tsou (Ting ZOU) and Xiangyang Zhang. 2012. IPsec Anti-Replay Algorithm without Bit Shifting. RFC 6479. (Jan. 2012). https://doi.org/10.17487/RFC6479