Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 11, 2018

C. Huitema
Private Octopus Inc.

January 7, 2018

QUIC Multipath Requirements
draft-huitema-quic-mpath-req-01

Abstract

This document describes the requirement and plausible architecture of QUIC multipath extensions. While the first version of QUIC is not scheduled to include multipath extensions, there are risks that decisions made in this first version might preclude some options that we may later find attractive. An early review of multipath extension requirements and issues should minimize that risk.

Table of Contents

1. Introduction

QUIC is a multiplexed and secure transport protocol that runs on top of UDP. QUIC aims to provide a flexible set of features that allow it to be a general-purpose transport for multiple applications. The QUIC working group is chartered to deliver a first version of the protocol in 2018. The current version is defined in a set of drafts, describing the base transport [I-D.ietf-quic-transport], congestion and recovery [I-D.ietf-quic-recovery], use of TLS [I-D.ietf-quic-tls], and mapping of HTTP over QUIC [I-D.ietf-quic-http].

According to its charter, the QUIC Working Group will first focus on delivering a first version of the protocol in 2018. This first version will focus on the basic scenario: client connecting to server over a single path, for a single application, HTTP. Multipath extensions will only be addressed later, once this first version is ready. Focusing on a simple version and delivering it first is obviously a pretty reasonable way to manage a complex project like the development of QUIC. However, there is a risk that decisions made during the development of the first version might preclude some options that we may later find attractive when developing the multipath extension.

It is almost impossible to discuss Multipath QUIC in an IETF context without making references to the Multipath extensions for TCP (MTCP), specified in [RFC6824]. The specification explains how to provide the same type of service to applications as TCP (i.e., reliable bytestream), while using multiple TCP flows across potentially disjoint paths. The details of the MTCP design are very specific to TCP, but many high level principles and experiences can inform the design of multipath QUIC.

The purpose of this document is to explore the multipath requirements and plausible solutions, in order to inform the development of QUIC V1 and hopefully facilitate a smooth introduction of multipath extensions after V1 is standardized.

2. QUIC Multipath Scenarios

The classic multipath scenarios involve splitting a connection's traffic over several paths, aiming for either better reliability by using several redundant paths, or better performance by balancing the traffic load over several path. For example, we see load balancing between equal cost multipaths using MTCP documented in [RFC8041]. Redundancy and load balancing define broad scenarios which will certainly be applicable to QUIC, but a couple of more specific scenarios have been discussed during the early development of QUIC. These scenarios include NAT re-binding, client migration, and anycast servers.

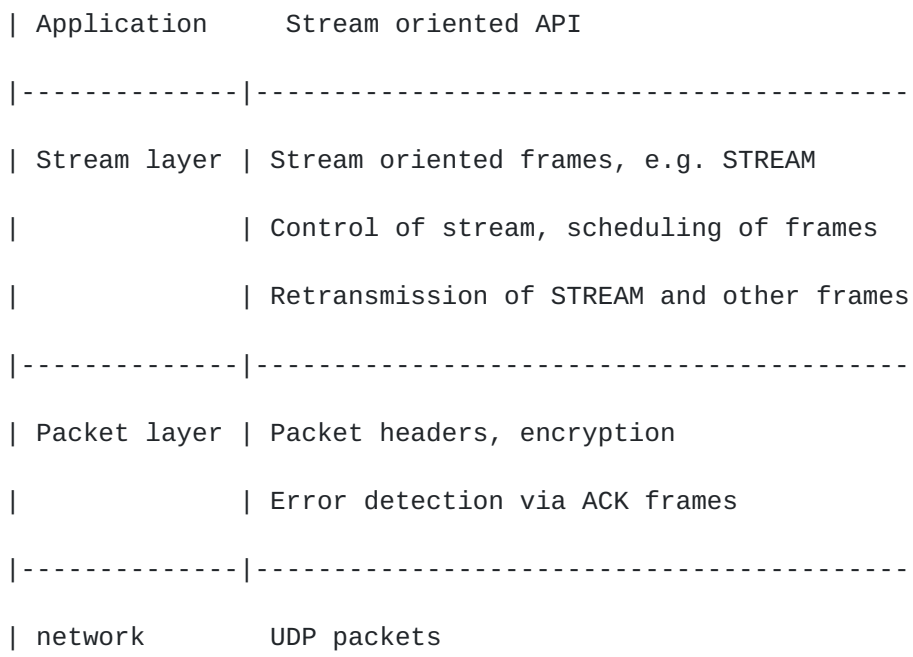
The classical example of client migration is the transition from one type of connectivity, e.g. Wi-Fi, to another, e.g. LTE. Client migration is often handled through a "make before break" strategy, in which a new "path" is established and tested before switching to it and decommisioning the old path. Such strategies involve maintaining the old and the new path active during the testing time, and thus require a simple form of multipath management. This is one of the scenarios studied in the context of MTCP, and documented in [RFC8041].

Many services are deployed using anycast addresses, which allow a client to get automatically connected to the closest available server. However, the connectivity between client and server can be affected by changes in routing conditions. For better reliability, it may be desirable to transition the connection to use an "unicast" address specific to that server instance. The transition from anycast to unicast requires a connection migration similar to the client migration described in the previous paragraph. That, too, requires a simple form of multipath management.

An even simpler form of migration happens when a client is located behind a NAT. The NAT will sometimes changed the mapping between the private address of the client and the public address visible to the server. From the server point of view, this may have some commonality with the "client migration" scenario described above, since the client appears to move to a new address. There are however some marked differences, such as the absence of the overlap period seen in the make before break scenario, or the fact that the client is generally unaware of the change in NAT mappings. For simplification, we will not consider NAT rebinding as a multipath scenario.

3. Basic Architecture And Discussion Points

QUIC is effectively a layered protocol, in which a stream of packets carries traffic multiplexed as a number of streams.



Most experts will recognize that the partition between the stream layer and the packet layer provides a natural opportunity for implementing multipath. For example, when describing a pre-standard experiment with QUIC multipath [I-D.deconinck-multipath-quic], De Coninck and Bonaventure state that "Since nearly all QUIC frames are independent of packets, no change is required (for supporting multipath)". This implies a broad agreement for an architecture where stream frames, at the stream layer, can be exchanged over several paths.

We can thus establish a set of consensual principles. A connection could be composed of many paths, each defined by a specific pair of client and server IP addresses and UDP ports. The set of paths associated to a connection could vary over time. Nodes could use various strategies to send stream oriented frames over the available paths. Losses of packets will be detected at the packet layer, and the corresponding stream-oriented frames may be retransmitted over any of the available paths. But to go from these easily drawn principles to a more precise architecture, we need to consider a set of requirements.

4. QUIC Multipath Requirements

4.1. Path Initiation

In a multipath scenario, one of the peers starts a new path after becoming aware that different addresses are available. In some scenarios, clients can start new paths because they gain additional connectivity, e.g. on Wi-Fi in addition to LTE. In these cases, the server will learn the availability of the new path by observing the IP addresses and ports on which the new packet is received.

In other scenarios, servers may ask clients to initiate new paths, for example in order to transition away from an anycast address. This requires communicating the additional addresses to the other peer. This may require defining an "ADD_ADDRESS" frame, as suggested in [I-D.deconinck-multipath-quic], if we want the additonal addresses to be passed withing QUIC. An alternative might be to let applications exchange addresses as part of their application protocol, and to use an API to "plumb" these addresses in the QUIC context.

Paths are normally initiated by having one peer send an encrypted packet to the other peer. The encrypted packet carries a connection identifiers that was previously associated with the connection, typically through a "NEW_CONNECTION_ID" frame. Privacy issues related to the use of connection identifiers are discussed in Section 4.2. In addition to privacy issues, we need to also consider the security issues, and specifically the risk of enabling denial of service attacks, as discussed in Section 4.3.

4.2. Privacy

When traffic is carried over multiple paths, it become observable at many points, and this has privacy implications. For example, if packets belonging to a given connection carry some unique identifiers, observers could use these identifiers to track client

migrations through several paths, and thus potentially expose the successive locations of a particular user.

The generally agreed privacy requirement is to prevent "linkability" between multiple paths. In QUIC, linkability could derive from observing components in the packet headers, such as connection identifiers or sequence numbers, an possibly also clear text parameters of the initial packet and the TLS "Client Hello" message that it carries.

In order to prevent linkability, we should adopt a set of clear guidelines, such as never reusing the same connection identifier over several paths, or breaking potential correlation betwen the sequence numbers used on multiple paths. We should also require that all messages used to set up additional paths be encrypted, in order to avoid leaking clear text identifiers. (The experiment described in [I-D.deconinck-multipath-quic] does not meet this privacy requirements.)

Note that these requirements are necessary, but may not be sufficient. Connections can potentially be correlated by observing packet timings and other characteristics.

4.3. Security

Path creation mechanisms can potentially be abused to enable denial of service against third parties. A rogue client could for example send path creation packets in which the source address is spoofed, and points to a third party. If the server accepts the path, it could then direct a stream of packets to that third party, contributing to a denial of service attack.

Similar attacks could be mounted by "man-in-the-middle" or "man-on-the-side" attackers, would could capture a legitimate client packet and replay it with a different source address.

These attacks are discussed in the transport draft [I-D.ietf-quic-transport] in the context of connection migration. The proposed solution is to have an explicit handshake in which each peer demonstrates that it can send and receive packets on the proposed path.

4.4. Non Detectability

Lots of efforts are being made in the design of QUIC to prevent "ossification", which can happen if middleboxes somehow start parsing QUIC packets and dropping them if some parameter does not have an expected value. Ideally, we would like to encrypt the entire packet,

including its header, to prevent such mistreatments, but some fields like connection ID or sequence numbers still have to be exposed. If a multipath extension exposed new packet header fields, it would run the risk of failing in the presence of already ossified middleboxes, or conversely of creating an expectation in middleboxes that would increase the risk of ossification.

To prevent such risks, the multipath extensions should use pretty much the same packet header formats as QUIC V1. (The experiment described in [I-D.deconinck-multipath-quic] was not attempting to meet this requirement.)

4.5. Error Detection

In QUIC, errors are detected by observing holes in the received acknowledgements. A packet will be declared lost if it is not acknowledged by the receiver, while later transmitted packets are. We could design multipath extensions to operate error detection either separately on each path, or globally on the entire set of paths.

Global processing require that the sequence numbers be coordinated between the different paths, e.g. sending packets 1, 2, 3 on Path 1, then packets 4, 5, 6 on Path 2, etc. In order to meet the requirements asserted in Section 4.2, the packet sequence numbers will have to be encrypted, as explained in Section 4.9.

If a global sequence number is used, implementations can manage a single packet retransmission queue for packets sent to all paths. The ACK format currently defined in QUIC can be retained. However, we may expect that different paths have different latency and different queuing delays, which will manifest itself as a large amount of out of sequence deliveries and ACK frames carrying many ranges. This may make error detection less efficient, but implementations can regain efficiencies by keeping track of which packet was sent on what path, and then doing smarter treatment. For example, instead of marking a packet lost after a higher sequenced packet was acknowledged on any path, implementations could do that only if the higher sequenced packet was acknowledged on the same path.

Independent processing assumes that there will be independent packet sequence numbers on each space. For example, there may be packets 1, 2, 3 on Path 1, and a different set of packets 1, 2, 3 on Path 2. Each sequence number would of course start at a different random offset to break correlation. Independent numbering solves the privacy issues caused by packet number linkability, but causes potential problems with AEAD encryption, as explained in Section 4.8.

If independent sequence numbers are used for each path, implementations will have to maintain separate retransmission queues for each path. ACK frames will need to be specific to a given path, either because they are sent on that path, or because they carry an identifier of the path (the solution adopted in [I-D.deconinck-multipath-quic]). The "identifier" variant allows acknowledgements for one path to be sent on another path, which may prove very useful when dealing with some error conditions.

4.6. Congestion Control

We assume that different "QUIC paths" will be routed over different network paths. Each path will be composed of links of different latency and capacity, and will go through routers experiencing different levels of load. There are potential exceptions, such as one path using IPv6 addresses and another using IPv4 addresses when these addresses in fact correspond to the same network interfaces and will be routed similarly -- as pointed in [I-D.deconinck-multipath-quic]. But even with this IPv4/IPv6 example, chances are that some network elements will perform differently, for example by performing Network Address Translation for IPv4.

Since different paths experience different network conditions, it follows that congestion control should be executed separately for each path. We should note that the experimental congestion control for MTCP [RFC6356] tries to meet three separate goals:

- Goal 1 (Improve Throughput) A multipath flow should perform at least as well as a single path flow would on the best of the paths available to it.

- Goal 2 (Do no harm) A multipath flow should not take up more capacity from any of the resources shared by its different paths than if it were a single flow using only one of these paths. This guarantees it will not unduly harm other flows.

- Goal 3 (Balance congestion) A multipath flow should move as much traffic as possible off its most congested paths, subject to meeting the first two goals.

Having separate instances of congestion control on each path will meet the second goal, and will also meet the first goal if combined with proper scheduling of frames on different paths. However, purely independent control of each path is not sufficient to meet the third goal, which is why "coupled congestion control" is introduced in [RFC6356]. This is largely a research issue. It is likely that initial deployments of multipath QUIC will start with a coupled

congestion control similar to [RFC6356], before exploring alternatives.

Congestion control algorithms require measuring acknowledgements, losses and delays on each specific path. This happens naturally if independent sequence numbers are used for each path. It will also happen if the solution uses a global sequence number space, as long as the congestion control algorithm is aware of the path associated with the acknowledgement or loss of individual packet, or with delay measurements.

4.7. Flow Control

Congestion control is a voluntary algorithm performed independently by each peer. It does not protect against misbehaving peers that would send too much data and overwhelm the receiver. In the first version of QUIC, this protection is ensured by connection level flow control, using MAX DATA frames, and stream level flow control, using MAX STREAM DATA frames.

At some theoretical level, one may worry that the connection level MAX DATA frame may not provide sufficient protection against excessive use of individual paths. For example, in a migration scenario, a low capacity path may inherit large credits from MAX DATA frames received on a high capacity path. If this turned out to be a real problem, peers would need to manage an independent flow control limit on each path, perhaps using a multipath specific MAX PATH DATA frame.

On the other hand, we can observe that MTCP uses the same flow control window for each separate path. This was done after considering using different windows for different paths, as explained in [NSDI12]. If the same window is used for all paths, we have the guarantee that should one path fail, all packets sent on that path can be retransmitted on any of the other paths, without having to wait for flow control updates.

Based on the MTCP experience, we should assume that in the first versions of multipath QUIC, flow control will remain a global "stream level" function.

4.8. Packet Encryption

QUIC packets are protected by AEAD encryption. The security of the AEAD algorithm depends on never using the same nonce for a given key. If the same nonce was used twice, the encryption algorithm would repeat the same "stream cipher" twice, and simple XOR could provide cues on the packet's content. In QUIC V1, the nonce uniqueness

requirement is met by mixing the packet sequence number with an otherwise preset IV. Since each packet carries a different sequence number, the nonce never repeats.

If different paths use uncoordinated sequence numbers, there is a risk that at some points the numbers on one path repeat numbers previously used on another path. If all paths used the same encryption key, this would lead to AEAD failure. To avoid that failure, different paths will have to use either different keys or otherwise ensure the uniqueness of nonces.

Different encryption keys per path could be obtained in various ways, such as mixing the path specific connection ID with a master secret, or considering that different paths use different iterations of the secret expansion algorithm. However, it can be argued that this management of multiple keys significantly increases the complexity of the protocol, especially if we consider the management of key rotation.

Instead of just mixing the packet sequence number with the preset AEAD IV, implementations could create nonces by mixing the IV with both the path specific connection ID and the packet sequence number. This would provide for nonce uniqueness across multiple paths, without imposing significant management burden. It may require changing the API to the AEAD encryption in some stacks, but the change is trivial.

4.9. Sequence Number Encryption

Privacy issues arise if the same sequence number space is used across multiple paths, as adversaries can observe the numbers and use them to link otherwise independent paths. This privacy issue is mitigated if different paths use different sequence number spaces, but it can also be mitigated if the packet sequence numbers are encrypted.

Packet sequence number encryption is not commonly practiced, but it is easy to imagine. For example, implementation could first encrypt the packet's payload using AEAD, and then derive a unique masking number by computing the keyed hash or the encryption of the last 16 bytes in the encrypted payload, and then XOR the sequence number field in the packet header with that masking number.

On the other hand, on-path diagnostic tools currently observe the packet numbers to detect transmission errors and thus potential network level issues. Encrypting the packet numbers would impede these tools.

4.10. Key Phases

It is generally considered unsafe to encrypt "too much" data with a single encryption key. QUIC supports a key update mechanism. Short packet headers include a KEY_PHASE bit that "allows for refreshes of keying material by either peer".

If the same encryption key is used for all paths, the key updates will happen at the same time for all different paths. An adversary might be able to observe the simultaneous changes in the KEY_PHASE bit and link otherwise uncorrelated paths.

This KEY_PHASE bit linkability is a privacy issue, but whether it is important is debatable. For client or server migration scenarios, the issue can be easily mitigated by not engaging in key updates during a migration, but for load balancing scenarios this simple mitigation will not suffice. Other mitigations could include delaying the key transition by random values on different paths, or performing double transitions on one path while the other paths remain silent.

The KEY_PHASE bit linkability does not occur if different paths use independently derived key materials, which is one of the option suggested in Section 4.8.

4.11. Key and sequence numbers per path

Our initial analysis pointed to two possible designs, a global sequence packet sequence number common to all paths, or separate packet sequence numbers for each path. After reviewing the consequences, it is clear that the simplest design is to use different keys and separate sequence numbers for each path. This is consistent with the use of a separate CONNECTION-ID for each path. It does not require encrypting the sequence number, and thus does not impede develoment of network quality monitoring tools. It can also provide an alternative to the KEY_PHASE mechanism: if too much data was sent on a path and new keying material is needed, just open a new path.

5. Next Steps

The first version of QUIC is progressing steadily, and engaging on multipath discussion might detract from that progress. However, it seems that parity with GQUIC requires handling at least the client migration scenario, in addition to NAT rebinding. As soon as we start working on that scenario, we will have to tackle a number of multipath related issues.

The early design of connection migration mechanisms privilege the simplest multipath options, and in particular the use of a single sequence number space for the old and new streams. This is fine, but we should probably start considering several of the issues listed in Section 4. In particular:

- the standard should mandate the use of different connection ID for different paths, as explained in Section 4.2
- implementations should consider the use of "path tags" for packets in the retransmission queue, as explained in Section 4.5.
- implementations should manage separate congestion control contexts for each path, as explained in Section 4.6.
- the standard should consider simple ways to associate different encryption contexts with each path and each connection-ID.

6. Acknowledgements

Many of the issues in this draft were already debated in the QUIC working group. Thanks to Yoshifumi Nishida and Martin Thomson for their constructive comments. Thanks also to Olivier Bonaventure and Quentin De Coninck for detailed feedback on the first version of this draft, and for many references to the experience gained with MTCP.

7. Informative References

[I-D.deconinck-multipath-quic]
Coninck, Q. and O. Bonaventure, "Multipath Extension for QUIC", draft-deconinck-multipath-quic-00 (work in progress), October 2017.

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol (HTTP) over QUIC", draft-ietf-quic-http-08 (work in progress), December 2017.

[I-D.ietf-quic-recovery]
Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-08 (work in progress), December 2017.

[I-D.ietf-quic-tls]
Thomson, M. and S. Turner, "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-08 (work in progress), December 2017.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-08 (work in progress), December 2017.

[NSDI12] Raicius, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and M. Handley, "How hard can it be? designing and implementing a deployable multipath TCP", NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, April 2012, <https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf>.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <https://www.rfc-editor.org/info/rfc6356>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <https://www.rfc-editor.org/info/rfc6824>.

[RFC8041] Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", RFC 8041, DOI 10.17487/RFC8041, January 2017, <https://www.rfc-editor.org/info/rfc8041>.

Author's Address

Christian Huitema
Private Octopus Inc.
Friday Harbor WA 98250
U.S.A

Email: huitema@huitema.net