Workgroup: QUIC
Internet-Draft:
draft-seemann-quic-nat-traversal-00
Published: 10 July 2023
Intended Status: Standards Track
Expires: 11 January 2024
Authors: M. Seemann
Protocol Labs

Using QUIC to traverse NATs

Abstract

QUIC ([RFC9000]) is well-suited to various NAT traversal techniques. As it operates over UDP, and because the QUIC header was designed to be demultipexed from other protocols, STUN ([RFC5389]) can be used on the same UDP socket. This allows for using ICE ([RFC8445]) with QUIC. Furthermore, QUIC's path validation mechanism can be used to test the viability of an address candidate pair, allowing the immediate use of a new path.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the QUIC Working Group mailing list (quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/.

Source for this draft and an issue tracker can be found at https://github.com/marten-seemann/draft-seemann-quic-nat-traversal.

Status of This Memo

Table of Contents

1. Introduction

This document defines three distinct modes for traversing NATs using QUIC:

1. Using ICE with an external signaling channel to select a pair of UDP addresses. Once candidate nomination is completed, a new QUIC connection between the two endpoints can be established.

2. Using a (proxied) QUIC connection as the signaling channel for ICE. Once ICE has nominated a candidate pair (i.e., selected a usable path), the proxied connection is migrated using QUIC's connection migration.

3. Using a (proxied) QUIC connection as the signaling channel for ICE. Instead of using ICE's connectivity checks, QUIC's path validation logic is used to determine possible paths.

The first mode doesn't require any changes to existing QUIC and ICE stacks. The only requirement is the ability to send non-QUIC (STUN) packets on the UDP socket that a QUIC server is listening on. However, it necessitates running a separate signaling channel for the communication between the two ICE agents.

The second mode requires a minor modification of the QUIC stacks involved, as they now need to be capable of exchanging ICE messages on top of the proxied QUIC connection. This is achieved by defining an ICE frame to carry these messages. This mode makes it possible to start exchanging application data (via QUIC streams) on the proxied connection. The migration event is transparent to the application.

The third mode necessitates changes to both the QUIC and ICE stacks. The ICE delegates responsibility for performing connectivity checks to the QUIC stack. The QUIC stack utilizes QUIC's path validation logic to perform the connectivity check. In addition to the path validation mechanism described in [RFC9000], the QUIC server needs the capability to initiate path validation, which, as per [RFC9000], is solely executed by the client. Compared to the second mode, this mode elimitates the need to effectively probe a path twice (once at the ICE and once at the QUIC layer), leading to a faster connection migration.

2. Modes

2.1. Using an External Signaling Channel

When an external signaling channel is used, the QUIC connection is established after the two ICE agents have agreed on a candidate pair. This mode doesn't require any modification to existing QUIC stacks, particularly, it does not necessitate the negotiation of the ICE extension defined in this document.

Once ICE has completed, the client immediately initiates a normal QUIC handshake using the server's address from the nominated address pair. The ICE connectivity checks should have created the necessary NAT bindings for the client's first flight to reach the server, and for the server's first flight to reach the client.

2.2. Using a QUIC Connection as a Signaling Channel, using ICE for Connectivity Checks

A (proxied) QUIC connection (e.g. using CONNECT-UDP ([RFC9298])) can be used as the signaling channel required by the ICE protocol (see section 1 of [RFC8445]). ICE messages are sent on this QUIC connection using the ICE frame defined in this document. This mode requires the ICE extension defined in this document to be negotiated (Section 3). Implementions **MAY** use Trickle ICE ([RFC8838]) to speed up the exchange of address candidates.

Once ICE has successfully nominated a candidate pair, this path can be used as a direct connection between the two endpoints. The client **SHOULD** initiate a QUIC connection migration (section 9 of [RFC9000]) in a timely manner. The ICE connectivity check should have created all the NAT bindings needed for the QUIC path validation to complete successfully, however, these NAT bindings are usually only valid for a limited amount of time.

2.3. Using a QUIC Connection as a Signaling Channel, using QUIC for Connectivity Checks

Similar to mode 2, in this mode a (proxied) QUIC connection is used as the ICE signaling channel. Instead of performing the connectivy checks (section 7 of [RFC8445]) themselves, the ICE stacks delegates them to the QUIC stack.

The QUIC client **MUST** ensure that it ends up as the controlling agent (see section 2.3 of [RFC8445]). This can be achieved by sending the maximum allowed value for the tiebreaker value (see section 7.3.1. of [RFC8445]).

When the ICE stack requests to perform a connectivity check for an address candidate pair, each QUIC endpoint probes the path by sending a probing packet containing a PATH_CHALLENGE frames, as described in section 8.2 of [RFC9000]. Note that this differs slightly from [RFC9000], where only the client sends a probing packet. To create the required NAT bindings, it's necessary for both endpoints to send packets.

Upon the completion of path validation, the QUIC stack passes the result (successful or failed) back to the ICE stack. The ICE stack then nominates an address pair. The client **SHOULD** then migrate the QUIC connection to this path in a timely manner.

3. Negotiating Extension Use

Endpoints advertise their support of the extension needed for mode 2 and 3 by sending the ice (0x3d7e9f0bca12fea6) transport parameter (section 7.4 of [RFC9000]) with an empty value. An implementation that understands this transport parameter **MUST** treat the receipt of a non-empty value as a connection error of type TRANSPORT_PARAMETER_ERROR.

In order to the use of this extension in 0-RTT packets, the client **MUST** remember the value of this transport parameter. If 0-RTT data is accepted by the server, the server **MUST** not disable this extension on the resumed connection.

4. ICE Frame

```
ICE Frame {
    Type (i) = 0x1ce,
    Length (i),
    Data (...),
}
```

The ICE frame contains the following fields:

Length: A variable-length integer encoding the length of the following Data field.

Data: The ICE message.

If the Length is larger than the remaining payload of the QUIC packet, the receiver **MUST** close the connection with a connection error of type FRAME_ENCODING_ERROR.

ICE frames are ack-eliciting. When lost, they **MUST NOT** be retransmitted, as the ICE layer is handling retransmission of messages.

5. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

6. Security Considerations

TODO Security

7. IANA Considerations

This document has no IANA actions.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/rfc/rfc5389>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/rfc/rfc8445>.

[RFC8838] Ivov, E., Uberti, J., and P. Saint-Andre, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", RFC 8838, DOI 10.17487/RFC8838, January 2021, <https://www.rfc-editor.org/rfc/rfc8838>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9298] Schinazi, D., "Proxying UDP in HTTP", RFC 9298, DOI 10.17487/RFC9298, August 2022, <https://www.rfc-editor.org/rfc/rfc9298>.

Acknowledgments

TODO acknowledge.

Author's Address

Marten Seemann
Protocol Labs

Email: martenseemann@gmail.com