           Peer-to-Peer Connections for the QUIC Transport Protocol
                       draft-perkins-quic-p2p-mux-00

Abstract

   The QUIC transport protocol is intended to be a general purpose
   transport, but is currently defined for client-server operation only.
   To be applicable to all use cases, it needs to develop support for
   peer-to-peer connection establishment. This memo describes how this
   can be done, in outline form. Future work is needed to determine if
   such peer-to-peer use of QUIC is desirable and, if so, to define a
   complete and workable standard for peer-to-peer QUIC connection
   establishment.

include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

   QUIC [I-D.ietf-quic-transport] is a multiplexed and secure general-
   purpose transport protocol. It is a connection-oriented protocol,
   where the end-points take the role of either client or server. The
   server passively listens for incoming connections; clients actively
   connect to servers. Once the connection has been established, QUIC
   is symmetric and allows either end-point to send and receive data on
   multiplexed streams within the connection.

   The client-server design of QUIC supports connection establishment
   when client and server are in the same addressing realm, or if the
   client is behind a network address/port translator (NAT). In this
   latter case, the outgoing connection request establishes state in the
   NAT, opening the port to allow the response from the server to reach
   the client. QUIC provides connection migration and path validation
   mechanisms that ensure connections can survive NAT rebinding events.
   The initial version of QUIC has no support, however, for establishing
   connections with a server that is behind a NAT. Specifically, QUIC
   does not provide any mechanism to probe connectivity and create the
   necessary NAT bindings to allow incoming connections to a server that
   is behind a NAT.

   The combination of the STUN [RFC5389] protocol and the Interactive
   Connectivity Establishment (ICE) framework [RFC8445] provides those
   mechanisms needed to establish connections in the presence of NATs,

supporting path probing and NAT binding discovery/creation. This
memo discusses how STUN and ICE can be used with QUIC to establish
connections when the server (and optionally the client) are behind
NATs.

2. Background

A NAT translates IP addresses and ports in IP and UDP packet headers,
separating the addressing realm used behind the NAT from the external
addressing realm. The addressing realm behind the NAT will often use
private IP addresses [RFC1918], and the external addressing realm is
often the public Internet, but this is not mandated. When a packet
is sent from an endpoint behind a NAT to an external endpoint, state
is created in the NAT allowing replies to be returned. This is known
as a NAT binding. If the NAT receives a packet on an external port
that is not subject to an active NAT binding, that packet is dropped.

Two QUIC endpoints wish to communicate. One or both of the endpoints
might be behind a NAT. If the QUIC client is behind a NAT, then the
outgoing initial packet sent from client to server to establish the
connection will create a NAT binding, allowing the response from the
QUIC server to pass the NAT and reach the client. The QUIC handshake
proceeds as normal, and the connection is established. However, if
the QUIC server is behind a NAT, then the initial packet sent by the
QUIC client will reach a port on the NAT for which there is no active
binding and will be dropped. The connection cannot be established in
this case.

ICE provides a framework for probing connectivity and creating NAT
bindings to allow connections to be established. It proceeds in four
phases:

o Gathering candidates: The endpoints discover the set of possible
   IP addresses and ports ("transport addresses") on which they can
   be reached, known as the set of "candidates". The candidate set
   includes transport addresses bound to directly attached network
   interfaces, translated transport addresses on the outside of a NAT
   ("server-reflexive addresses"), and transport addresses allocated
   via some form of relay ("relayed addresses". When gathering the
   candidates, STUN is used to discover server reflexive addresses,
   and the relay protocol (e.g., TURN) is used to determine relayed
   addresses.

o Exchanging candidates: The endpoints exchanges their candidate
   sets. Since the two endpoints cannot directly communicate, due to
   the presence of the NATs, this exchange has to be done indirectly
   via an external relay that is reachable by both endpoints. In the
   multimedia conferencing systems for which ICE was defined, this

exchange takes place over the signalling channel (e.g., SIP or
WebRTC). This signalling is heavyweight and multimedia specific,
and is not appropriate for QUIC, so an alternative mechanism will
need to be defined.

o Connectivity checks: Once the candidate sets have been exchanged,
   the endpoint systematically probe all pairs of local and remote
   candidates, in priority order, to determine which candidate pairs
   can be used to communicate. Each pair of candidates is probed, by
   sending a request and checking for a response, in both directions.
   This ensures that NAT bindings are established form each endpoint,
   so connectivity is found even if both endpoints are behind NATs.

o Connection establishment: Once the connectivity checks succeed for
   a suitable pair of candidates, the connection is established using
   those candidates in the normal manner.

The ICE framework was designed to support multimedia conferencing,
and defines the signalling to exchange candidates in a way that is
specific to those systems. The technique is generic however, and can
be adapted to support NAT traversal and connection establishment for
QUIC.

3. QUIC Connection Establishment in the Presence of NATs

3.1. Gathering Candidates

Candidates are gathered as in Section 5 of [RFC8445]. The Initiating
Agent is the QUIC client. The Responding Agent is the QUIC server.
RTP and RTCP are not used, and candidates are gathered for a single
QUIC component instead. A QUIC server that "knows" that it is not
behind a NAT MAY use the lite mechanism described in Section 5.2 of
[RFC8445].

Editor's note: clarifications are likely needed since the candidate
gathering in [RFC8445] is written assuming RTP and RTCP candidates,
but the approach looks suitable for QUIC without significant change.

3.2. Exchanging Candidates

Section 5.3 of [RFC8445] describes the information to be communicated
during the exchange of candidates. This can be used unchanged, but
the protocol used to exchange the candidates needs to be defined for
the ICE usage for QUIC.

The key design decision to make is what is the signalling protocol to
be used with ICE when establishing QUIC connections? The multimedia
conferencing uses of ICE naturally use an SDP offer/answer exchange

[I-D.ietf-mmusic-ice-sip-sdp] to convey this information, but asking
QUIC endpoints to implement SDP is not acceptable. An alternative is
needed.

There are two approaches. Firstly, the candidate exchange can be
performed out-of-band from the point of view of QUIC, by some higher
level signalling protocol. This assumes that the application that
uses QUIC will perform the necessary signalling. The application
will extract the set of local candidates from the QUIC stack and pass
them to the peer, then later pass in the set of remote candidates
received from the peer to the QUIC stack to trigger the systematic
probing of candidate addresses. This requires no QUIC-specific
protocol standardisation, other than to specify when the exchange
happens, since the signalling protocol is application specific and
the interface between the QUIC stack and that signalling protocol is
implementation specific. Standards are likely needed to define the
signalling for specific applications that use QUIC in this way.

Alternatively, the candidate exchange can be embedded into QUIC to
provide a general way for an indirect QUIC connection, made via a
middlebox relay of some sort, to signal candidates to the endpoints,
allowing them to bootstrap a direct peer-to-peer QUIC connection.
This is a significant change to the QUIC security model, since it
introduces a trusted middlebox that can relay candidate information,
so exposing that communication is taking place. However, such a
trusted device is necessary to bootstrap any direct peer-to-peer
connection in the presence of NATs, since it is not possible to
establish a direct connection without the help of a relay. Adding
the candidate exchange mechanism to QUIC allows the provision of
generic NAT traversal bootstrapping, allowing for reusable libraries
and common mechanisms, but potentially risks missing out on
application integration that could provide useful peer identity
guarantees and end-to-end security mechanisms for the signalling.

The right approach is likely to define a common abstract interface
between QUIC and the candidate exchange signalling, specifying what
information is to be exchanged and when. Then, also define a common
signalling protocol that MAY be used if there is no more suitable
application specific mechanism.

3.3. Connectivity Checks

The assumption is that connectivity checks proceed using STUN, as is
described in Section 7 of [RFC8445]. This has the benefit of working
with existing STUN servers, reusing existing specifications, and of
allowing code reuse. It also continues to exercise the ability of
middleboxes to pass STUN traffic, which is necessary for WebRTC NAT
traversal and to support other peer to peer applications. It further

ossifies STUN as the commonly used protocol for connectivity checks
in the Internet.

Alternatively, a QUIC-specific connectivity check protocol could be
developed. This would fulfil the same role STUN plays in [RFC8445],
but could use a syntax that is closer to that of other QUIC packets,
and could perhaps better integrate with the QUIC security mechanisms.
It is the belief of the author that the benefits of such a new syntax
for a STUN-like service do not outweigh the complexity of developing
the new mechanism, and that it is beneficial to have a single, widely
used, connectivity checking standard. It is, however, possible to
define a new connectivity check protocol, and such a protocol could
avoid some of the complexity and backwards compatibility features
included in STUN.

3.4. Connection Establishment

The connectivity checks determine that it is possible to send STUN
packets in UDP on an particular 5-tuple. Once the connectivity
checks have concluded, and a candidate pair has been selected, the
QUIC endpoints will attempt to establish a QUIC connection on the
chosen path.

To ensure the NAT ports are open, it is necessary that traffic flows
in both directions. It might be that it is sufficient to pick a peer
to act as the server (e.g., highest numbered IP address becomes the
server), or it might be necessary to perform a simultaneous open of
the QUIC connection (in a similar manner to TCP simultaneous open).
This will depend on the extent to which middleboxes attempt to detect
and control QUIC connection establishment. Simultaneous open seems
most robust, but at the expense of complexity.

4. Demultiplexing QUIC and STUN

STUN and QUIC packets can be demultiplexed based on the value of the
first octet, as described in [I-D.aboba-avtcore-quic-multiplexing].
This provides a mechanism for middleboxes to distinguish peer to peer
and regular QUIC flows on the wire. This is beneficial in that it
further establishes STUN as the connectivity check mechanism for the
Internet, supporting its use in protocols such as WebRTC. It is
problematic, in that it provides a mechanism that can be used to
detect and prevent peer to peer uses of QUIC.

5. Security Considerations

To be completed. Key initial topics include: 1) changes to the QUIC
security model, since any attempt to provide relayed connectivity
checks for NAT traversal requires some degree of trust in the relay

server; 2) information leaks via STUN, if used for connectivity
checks, even if just leaking that a peer-to-peer connection
establishment is ongoing (the latter can likely be inferred from the
range of IP addresses used, but there might be other information
leaks); and 3) security properties of the signalling protocol used to
communicate with the relay server.

6. IANA Considerations

No IANA actions needed. Yet.

7. Acknowledgements

8. References

8.1. Normative References

[I-D.aboba-avtcore-quic-multiplexing]
          Aboba, B., Thatcher, P., and C. Perkins, "QUIC
          Multiplexing", draft-aboba-avtcore-quic-multiplexing-03
          (work in progress), January 2019.

[I-D.ietf-quic-transport]
          Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed
          and Secure Transport", draft-ietf-quic-transport-18 (work
          in progress), January 2019.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.,
          and E. Lear, "Address Allocation for Private Internets",
          BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996,
          <https://www.rfc-editor.org/info/rfc1918>.

[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/info/rfc5389>.

[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/info/rfc8445>.

8.2. Informative References

   [I-D.ietf-mmusic-ice-sip-sdp]
              Petit-Huguenin, M., Nandakumar, S., and A. Keranen,
              "Session Description Protocol (SDP) Offer/Answer
              procedures for Interactive Connectivity Establishment
              (ICE)", draft-ietf-mmusic-ice-sip-sdp-24 (work in
              progress), November 2018.

Author's Address

   Colin Perkins
   University of Glasgow
   School of Computing Science
   Glasgow G12 8QQ
   United Kingdom

   Email: csp@csperkins.org