MASQUE Can Help!

SwiftOnSecurity  @SwiftOnSecurity  · 5h

It's the year 2042. IPv7 has been deployed. There is only one port: 443
Summary / Recap from IETF 108

CONNECT-UDP is like CONNECT, but for UDP!

When used in HTTP/3, it uses QUIC DATAGRAM frames to avoid retransmissions

Since last IETF, document was adopted by the MASQUE WG
Using QUIC DATAGRAM frames from HTTP/3

Currently relying on draft-schinazi-quic-h3-datagram

When QUIC is in use and ALPN=h3,
   Every QUIC DATAGRAM frame starts with a Flow Identifier (62-bit integer)
   Both endpoints provide a flow allocation service to get unique identifiers
   The protocol to negotiate these flow IDs is not defined in that draft

CONNECT-UDP carries the new "Datagram-Flow-Id" header to indicate flow ID

   :method = CONNECT-UDP
   :authority = server.example.com:443
   Datagram-Flow-Id = 42
We have an adoption inversion to fix

draft-ietf-masque-connect-udp

Draft

draft-schinazi-quic-h3-datagram

Normative Dependency

Adopted by MASQUE WG

Inversion

Draft

draft-ietf-quic-datagram

Normative Dependency

Adopted by QUIC WG

Individual Draft
Let's look at what's in these drafts today

- CONNECT-UDP – HTTP METHOD
- Datagram-Flow-Id – HTTP Header
- Datagram Flow ID – New Concept
- H3_DATAGRAM – HTTP SETTING
- DATAGRAM – QUIC Frame
- max_datagram_frame_size – QUIC TP

draft-ietf-masque-connect-udp

draft-schinazi-quic-h3-datagram

draft-ietf-quic-datagram
Datagram Flow Identifiers

In HTTP/3, every QUIC DATAGRAM Frame starts with a variable-length integer that represents the Datagram Flow Identifier.

HTTP SETTING exists to indicate support for this extension.

We need a namespace:

- What does each datagram flow ID map to?
- How does one allocate new unique datagram flow IDs?
- How do we ensure that our peer understands this mapping?
Two ways of solving this

Reuse the HTTP request stream ID

Simpler
Forces 1-to-1 mapping from request to ID
Only uses 25% of available IDs
Leads to longer varint encoding

Datagram-Flow-Id HTTP Header

Distinct namespace
Requires explicitly negotiating via header
Allows many-to-1 mapping from request to ID
Used by draft-pauly-masque-quic-proxy
Allows potentially reusing flow IDs
More extensible
Where do we want everything to land?

- WEBTRANS WG
  - Http3Transport
  - CONNECT-UDP
  - Datagram-Flow-Id – HTTP Header
  - Datagram Flow ID – New Concept
  - H3_DATAGRAM – HTTP SETTING
  - DATAGRAM – QUIC Frame
  - max_datagram_frame_size – QUIC TP

- draft-ietf-masque-connect-udp
- draft-schinazi-quic-h3-datagram
- draft-ietf-quic-datagram

ADOPT IN MASQUE WG?
And Now for Something Completely Different: Some GitHub Issues!
Issues **#1** and **#3** – Chaining Multiple HTTP Proxies

Even though ProxyCorp appears to be a single machine to the client, it can be implemented as one or more HTTP intermediaries leading to a backend.
Issues #1 and #3 – Chaining Multiple HTTP Proxies

Chaining is straightforward when sending UDP payloads in the request stream.

Negotiating Datagram Flow ID across multiple proxy hops is non-trivial, as flow IDs are a property of the transport, and aren't end-to-end.

Should we make "Datagram-Flow-Id" hop-by-hop, and send the "Connection" header listing it to ensure it isn't forwarded?
Issue #8 – We need a request target URI

According to HTTP Semantics, all new methods MUST have a target URI.

There was an exception for CONNECT but it doesn't help with a new method.

How about: udp://target-host:port
Issue #11 – Limit packets before server response

CONNECT inherits this protection from TCP: it won't send anything to the target other than the SYN until it receives a SYN-ACK

UDP doesn't provide the same property

Should we say that the server SHOULD NOT send more than 10 packets per second until it's received any UDP in response from the target?
Questions?
MASQUE CONNECT-UDP

draft-ietf-masque-connect-udp

IETF 109 – Virtual – 2020-11

David Schinazi – dschinazi@google.com