MASQUE
HTTP Datagrams and CONNECT-UDP

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

IETF 112 – Virtual – 2021-11-08

David Schinazi – dschinazi.ietf@gmail.com
Lucas Pardue – lucaspardue.24.7@gmail.com

Previously, on MASQUE...

We are building CONNECT-UDP, like CONNECT but for UDP!
- We want it to work over all versions of HTTP, and across intermediaries
- When over HTTP/3, we want to leverage the QUIC DATAGRAM frame

There is interest in datagrams beyond CONNECT-UDP, so we split the draft into HTTP Datagrams + CONNECT-UDP

We had an interim in 2021-04, focused on the design of HTTP Datagrams

We redesigned everything, and after discussion on the list, merged some PRs

We then re-redesigned everything, because we need job security or something

Interlude: interop results

Latest draft:

server → / client ↓	quiche	Apple QUIC	Google	Ericsson
quiche	–	–	**QHCDENRO**	QHCDEO
Apple QUIC	–	–	–	–
Google	–	–	QHCDENRO	QHCDEO
Ericsson	–	–	**QHCDENRO**	QHCDENRO

https://github.com/ietf-wg-masque/draft-ietf-masque-connect-udp/wiki/Interop

Outcome of prior meetings

Strong coupling of datagrams with request streams

Capsule protocol – sequence of TLVs inside DATA frames

Datagram capsule

QUIC DATAGRAM frame starts with Quarter Stream ID varint

HTTP/3 SETTING to indicate support for QUIC DATAGRAM frames

We're not done yet

As per discussion on list, current design doesn't appeal to everyone

In particular, we haven't quite reached consensus on extensibility and demultiplexing

Extensibility/Demultiplexing – Motivation

CONNECT-IP compression of IP header

CONNECT-UDP carrying ECN markings

CONNECT-UDP carrying received ICMP

Path MTU Discovery for HTTP Datagrams (see Ben's presentation later)

Conveying multiple priority levels in WebTransport

→ Since we're inventing how to convey datagram data, there are extensions that would like to convey multiple types of datagrams and demultiplex between them

Disclaimer: this list for illustration purposes only. Extensions not guaranteed to be useful. Your mileage may vary. Talk to your local MASQUE enthusiast to find out more.

Demultiplexing – Why do we need to care today?

We could leave demultiplexing as a problem to be solved later

That's possible, but we need to make sure the base HTTP Datagrams draft has the right extensibility point to allow that

→ What are our requirements for our future extensibility/demultiplexing?

Extensibility – Requirements / Design Goals

Ability to convey multiple types of datagrams and demultiplex between them

Intermediaries do not need to be modified to support extensions

Ability to write cross-protocol extensions without too much duplication

Make this mechanism optional: minimize both implementation cost and concept burden for implementers that do not want this
→ can be rephrased as: minimize what's required from HTTP Datagrams core

Zero-latency extensibility

Extensibility – Support and Lack Thereof

Since this is optional, some implementations won't support extensions

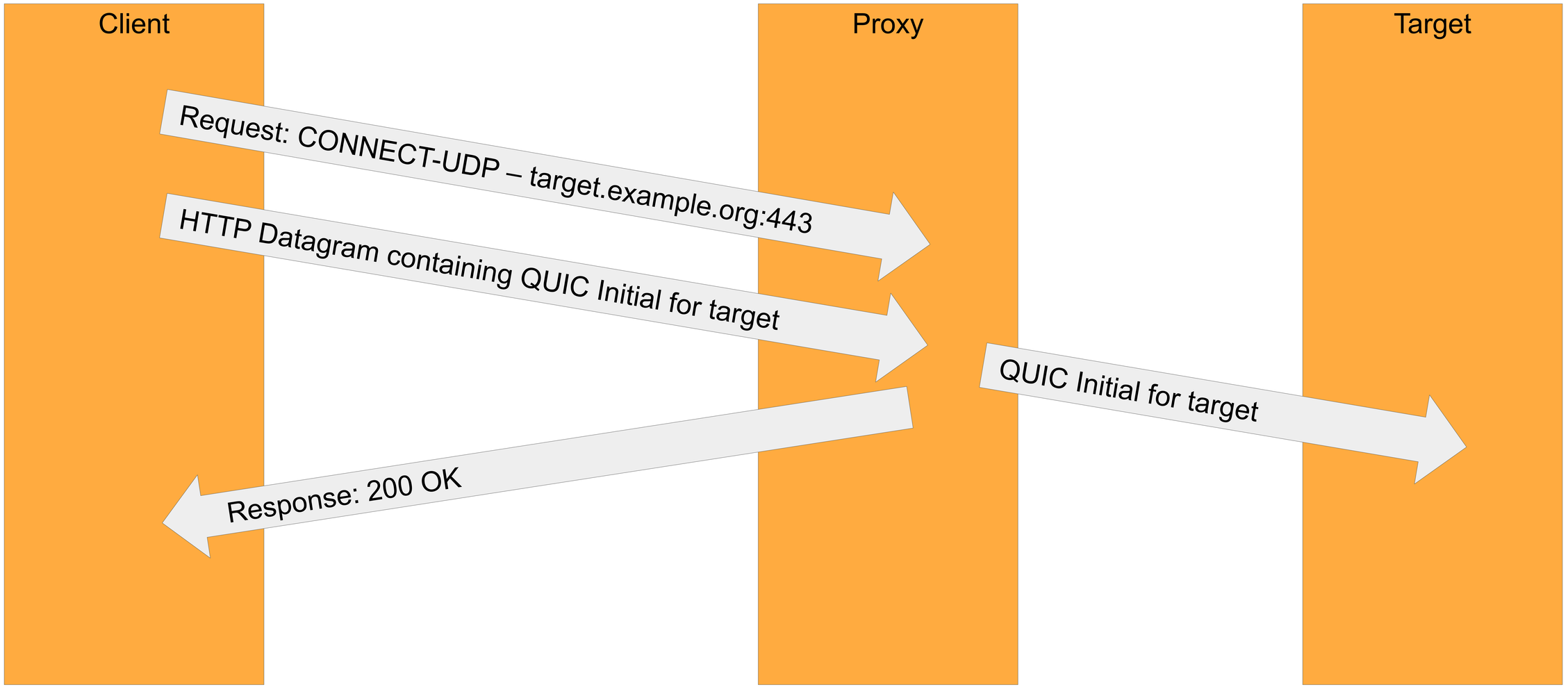
Clients don't initially know the feature set of proxies

Can't use SETTINGS when proxy is behind intermediary

Waiting one round trip for HTTP response is unacceptable

→ We need a way to use extensions optimistically with graceful fallback

Zero-latency Setup: QUIC over CONNECT-UDP



Extensibility and Demultiplexing

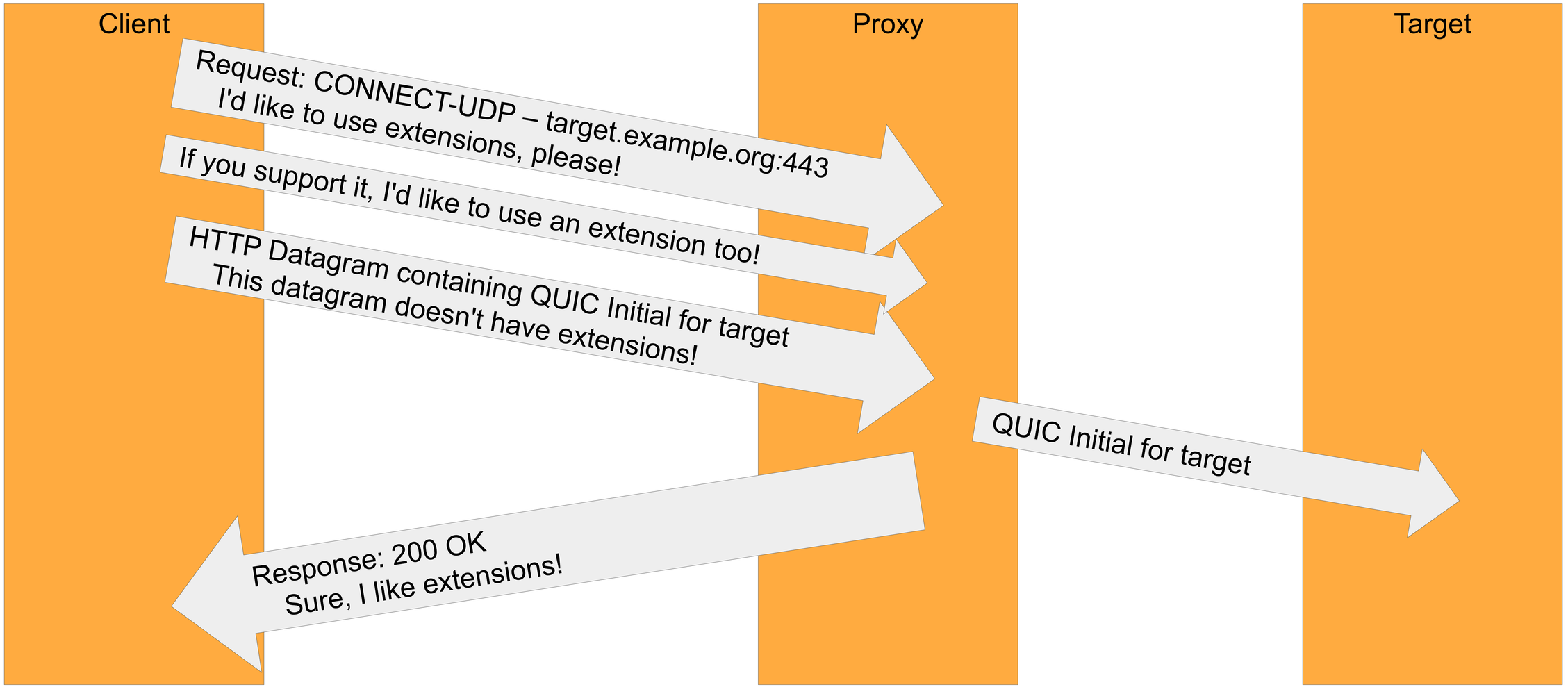
Some extensions need to send multiple types of datagrams and demultiplex between them

Simple solution: add an identifier at the start of HTTP Datagrams

Since not everyone wants this, make the identifier optional

→ How do we know if the identifier is there or not?

Zero-latency Extensibility



Extensibility – Current Design

Concept: Datagram Format Types (IANA-registered varint)

Negotiation: REGISTER_DATAGRAM capsule

Optional (negotiated via Sec-Use-Datagram-Contexts HTTP header):

- Concept: Context (per-stream varint at the start of each datagram)
- Negotiation: REGISTER_DATAGRAM_CONTEXT

No clear consensus on this yet: some might prefer to remove the concept of Datagram Format Type because concepts aren't free Issue#84

Extensibility – Potential new Design – PR#115

Remove Datagram Format Types, because concepts have a cost

PR also removes registration capsules and close capsules

Only one extension joint: new Capsule Types

QUIC DATAGRAM frame has a context ID only if negotiated by header
DATAGRAM capsule is the same way, intermediaries can simply convert
Neither can be used until headers are received

Before headers, use DATAGRAM_WITH_CONTEXT and DATAGRAM_WITHOUT_CONTEXT capsules

Extensibility – New Design vs Requirements

✅ Multiplexing
✅ Intermediaries are oblivious
✅ Simple cross-protocol extensions
✅ Zero-latency

Optionality: if we were to split contexts into their own draft, what's left here?

→ Two datagram capsules instead of one
- DATAGRAM capsule – same semantics as QUIC DATAGRAM frame
- DATAGRAM_WITHOUT_CONTEXT capsule – used before headers
  - If we removed the concept of context, rename to UNEXTENDED_DATAGRAM

Implementations that don't care use exact same implementation for both

Extensibility and Demultiplexing – Let's Chat

Do you have requirements that haven't been captured yet?

Does the latest proposal have properties that you object to?

Do you have other thoughts on how to proceed?

Kindly note that the horror movie titled "Masquerade" which came out this year is considered out of scope for this discussion

Extensibility – An Alternate Proposal – PR#114

Make the context varint a part of HTTP Datagrams

Use one bit in QUIC DATAGRAM frames to encode whether it's present

#111: RELIABLE_DATAGRAM

In current draft:

- intermediaries can convert between QUIC DATAGRAM frame and capsule
- intermediaries SHOULD NOT convert to capsule unless forced

Breaks down in the unlikely scenario:

- if the clients wants to PMTUD, it wants the middle link to use QUIC DATAGRAM
- if the clients wants to send IPv6 at 1280 bytes, it wants middle link capsules

We have a pretty simple solution: RELIABLE_DATAGRAM capsule

- allows sender to convey semantics it wants to intermediaries

CONNECT-UDP

Some changes in draft to stay in sync with HTTP Datagrams

CONNECT-UDP → Extended CONNECT with :protocol = connect-udp
  HTTP/1.1 uses Upgrade: connect-udp

Configuration via URI Template
  scheme and path are decided at configuration time, path contains target host/port
  authority like normal methods

```
https://masque.example.org/{target_host}/{target_port}/
https://proxy.example.org:4443/masque?h={target_host}&p={target_port}
https://proxy.example.org:4443/masque{?target_host,target_port}
```

#65: URI template or HTTP headers?

URI template has downsides:

- Requires parsing URI templates which is a pain
- Prevents reusing the same configuration with CONNECT-IP

Alternatively, replace the configuration URI template with a configuration URL and convey the target host/port in separate HTTP headers

#57: HTTP/1.1 Method for Upgrade

Current draft says to use CONNECT with Upgrade

WebSocket uses GET

Any reason to pick one over the other?

MASQUE
HTTP Datagrams and CONNECT-UDP

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

IETF 112 – Virtual – 2021-11-08

David Schinazi – dschinazi@google.com
Lucas Pardue – lucaspardue.24.7@gmail.com