HTTP Datagrams, UDP Proxying, and Extensible Prioritization

draft-pardue-masque-dgram-priority
IETF 116 – Yokohama – 2023-03

Lucas Pardue
HTTP Extensible Priorities

- RFC 9218
- Priorities $\iff$ Resource Usage
- Aka - prioritisation matters most when there are a bottleneck
- Deployment topology variation
  - Sometimes a bottleneck
  - Sometimes not
MASQUE multiplexing and bottlenecks
MASQUE multiplexing and bottlenecks

HTTP

Client

Proxy

Server
Server
Server
Server
Server
Server
Server
Server

L4 or L3
MASQUE multiplexing and bottlenecks

HTTP

Client

Proxy

More likely to be bottleneck

Less likely to be bottleneck

L4 or L3

Server

Server

Server

Server

Server
MASQUE multiplexing and bottlenecks

Client → Proxy → L4 or L3

More likely to be bottleneck
Knows intent

Less likely to be bottleneck
Does not know intent
When a stream carries a CONNECT request, the scheduling guidance in this document applies to the frames on the stream. A client that issues multiple CONNECT requests can set the incremental parameter to `true`. Servers that implement the recommendations for handling of the incremental parameter (Section 10) are likely to schedule these fairly, preventing one CONNECT stream from blocking others.

Effectively shares bandwidth

Works pretty well :thumbsup:
… but CONNECT UDP doesn’t use streams

No formal means for the client to express the desired property
What works well is splicing DATAGRAMs with STREAMs
Bandwidth sharing
But implementation specific and not clear if suits client needs
Clearer problem statement:

A client that is running multiple TCP and UDP tunnels in one connection has few tools available to mark some more important than others.

Same applies to the server
We can solve this

By integrating datagram flows into the extensible priority model.

By just adding a simple parameter with some basic guidance.

draft-pardue-masque-dgram-priority
Time to do something or do nothing

Questions?