draft-navarre-quic-flexicast
Flexicast QUIC

IETF 121 - Dublin
QUIC

Louis Navarre
Olivier Bonaventure

UCLouvain

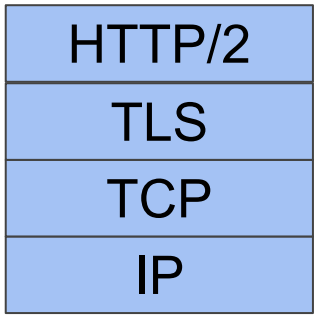
Goal: enable flexible multicast inside QUIC

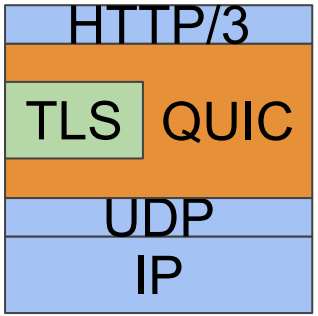
- Existing attempt to add multicast to QUIC: draft-jholland-quic-multicast
  - Discuss it later in the slides
- We present an alternative approach based on Multipath QUIC and our implementation experience
- A unique transport protocol for multicast and unicast delivery
- Working implementation based on *quiche*

The QUIC protocol

Transport protocol aiming to replace TCP for the web.

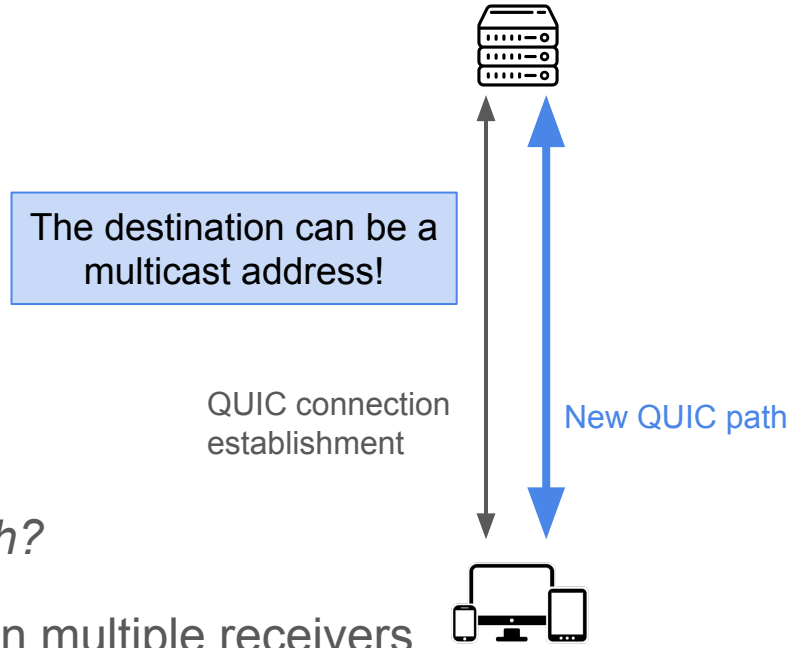
- Multi-streams and datagram capabilities
  - Gathers all the features of TCP, UDP and SCTP
- Security integrated *in* the transport
- Already serving most HTTP requests in Europe





Starting from Multipath QUIC…

- Each path uses different
  - Connection IDs
  - 4-tuples
  - Packet Number Spaces
- Same TLS keys
- Path probing to create the paths

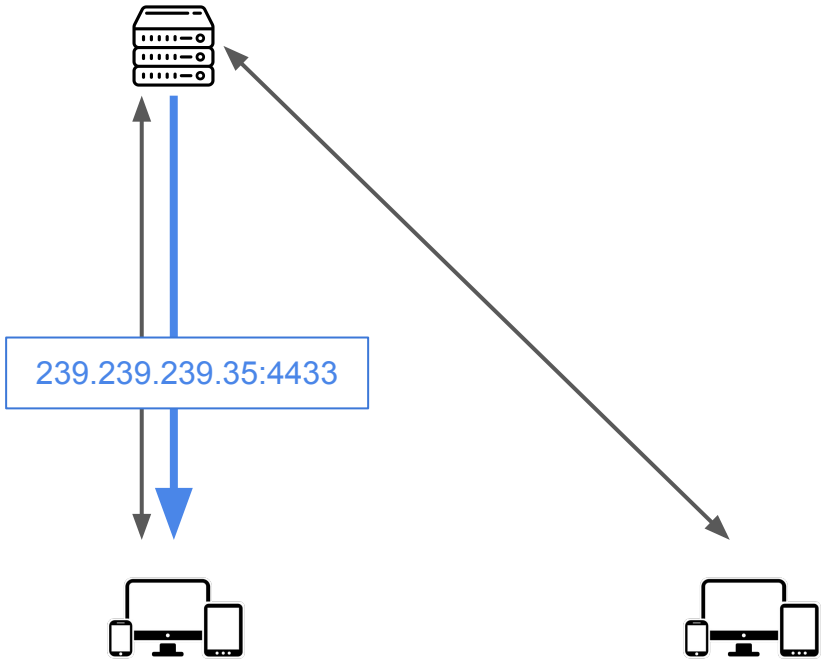


What if we use separate keys for each path?

⇒ The second path can be shared between multiple receivers

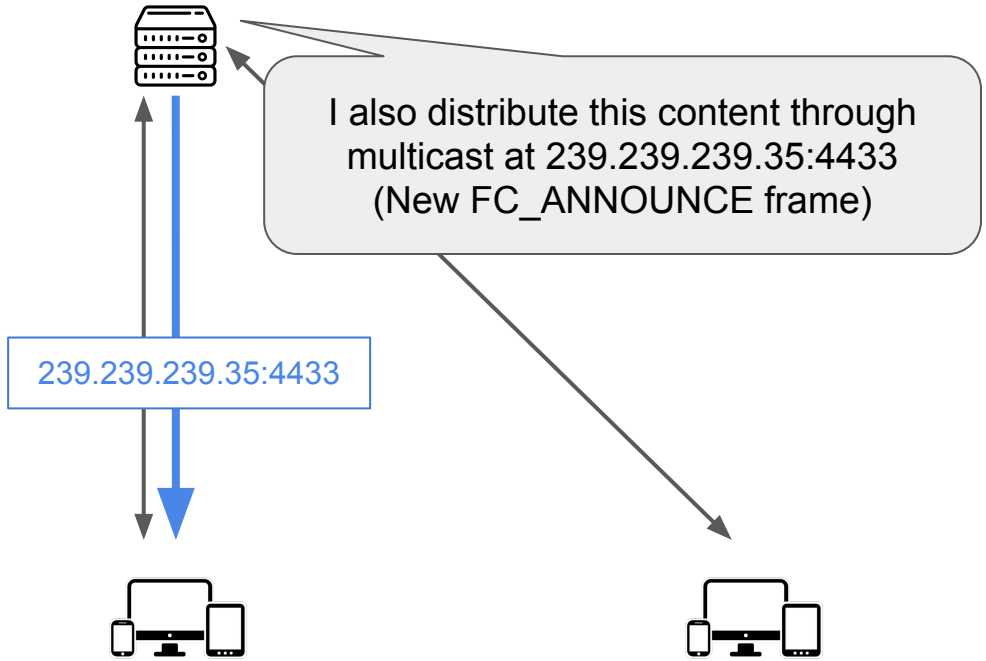
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment



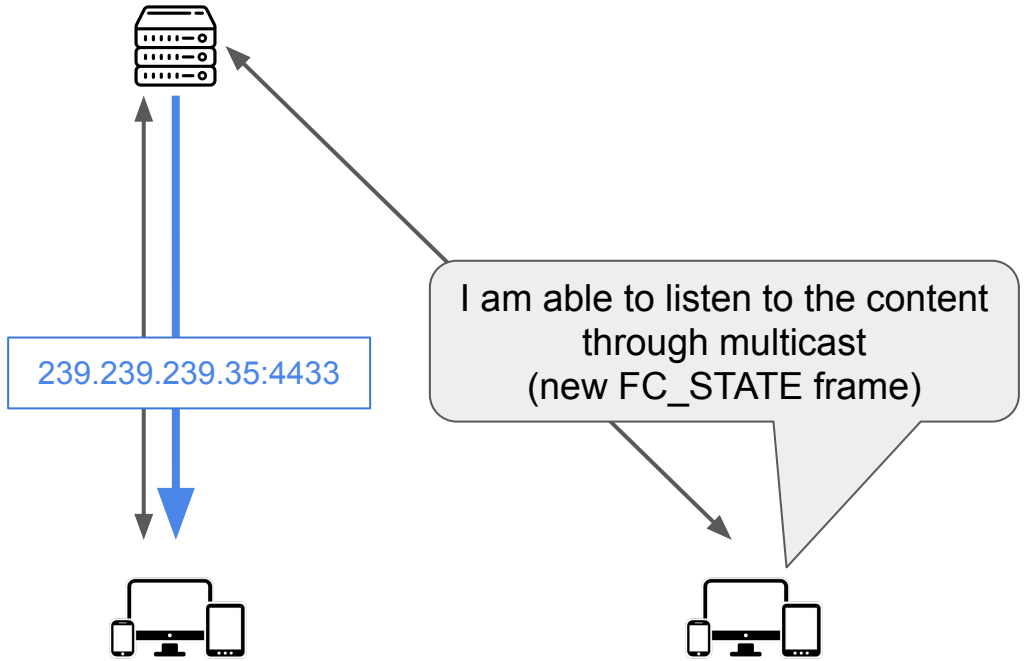
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement



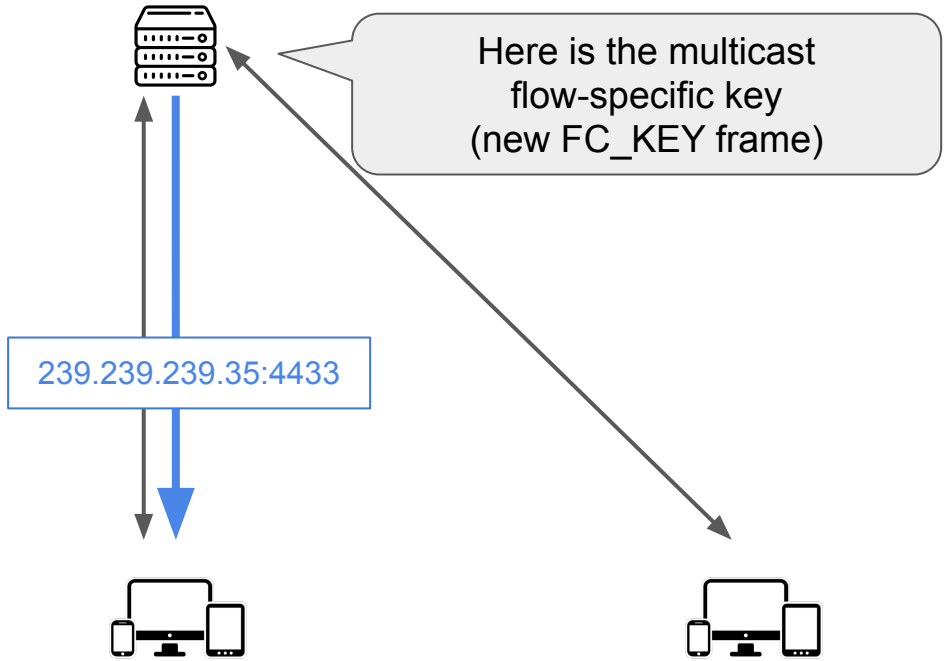
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement



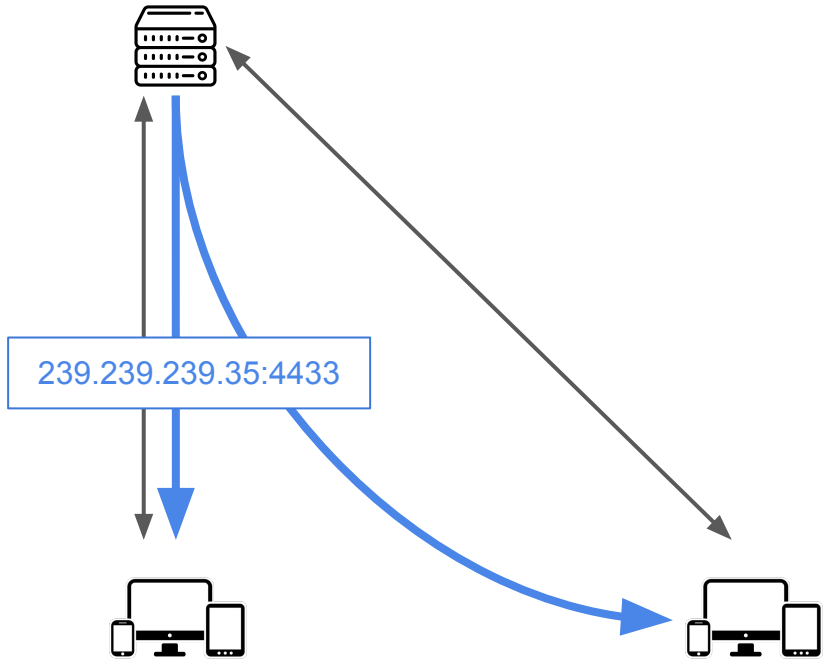
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement



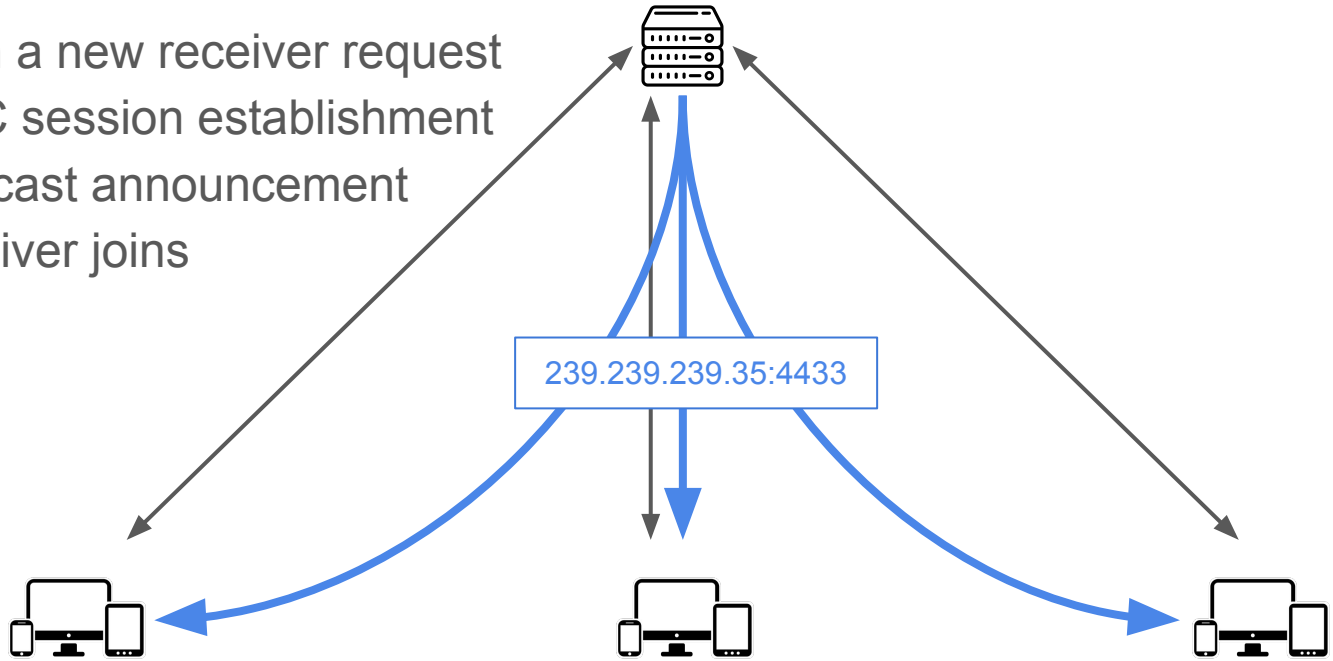
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement
- Receiver joins



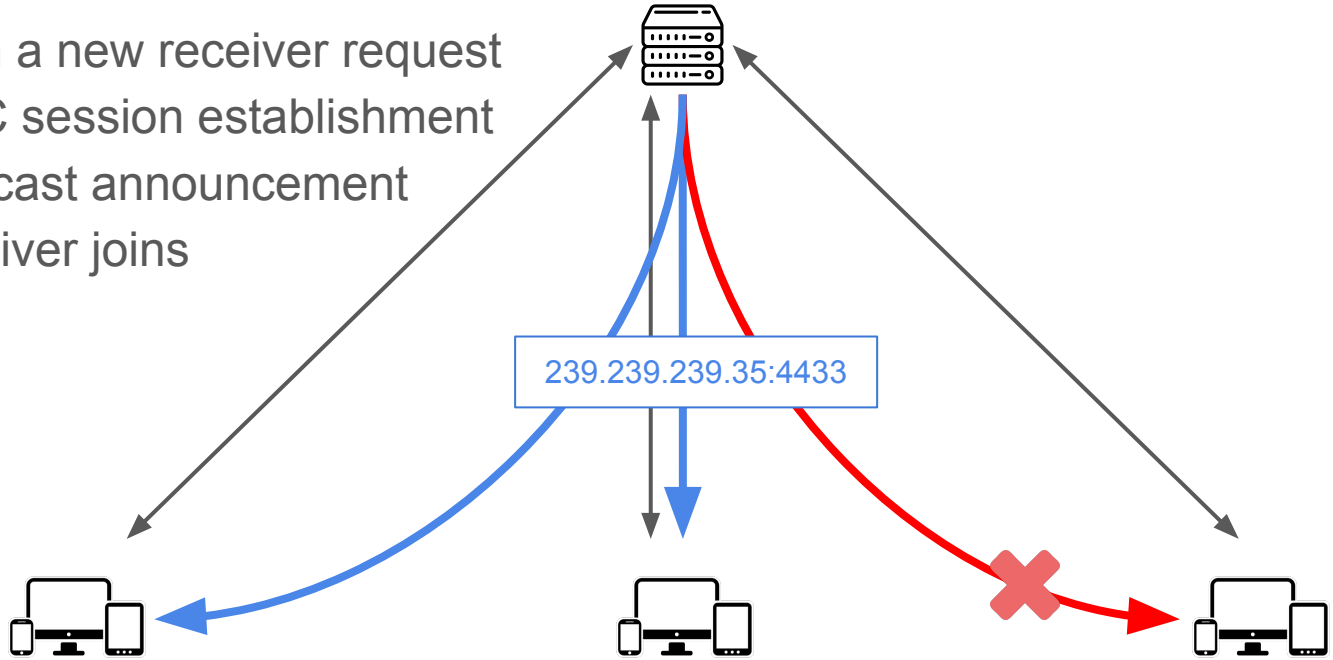
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement
- Receiver joins



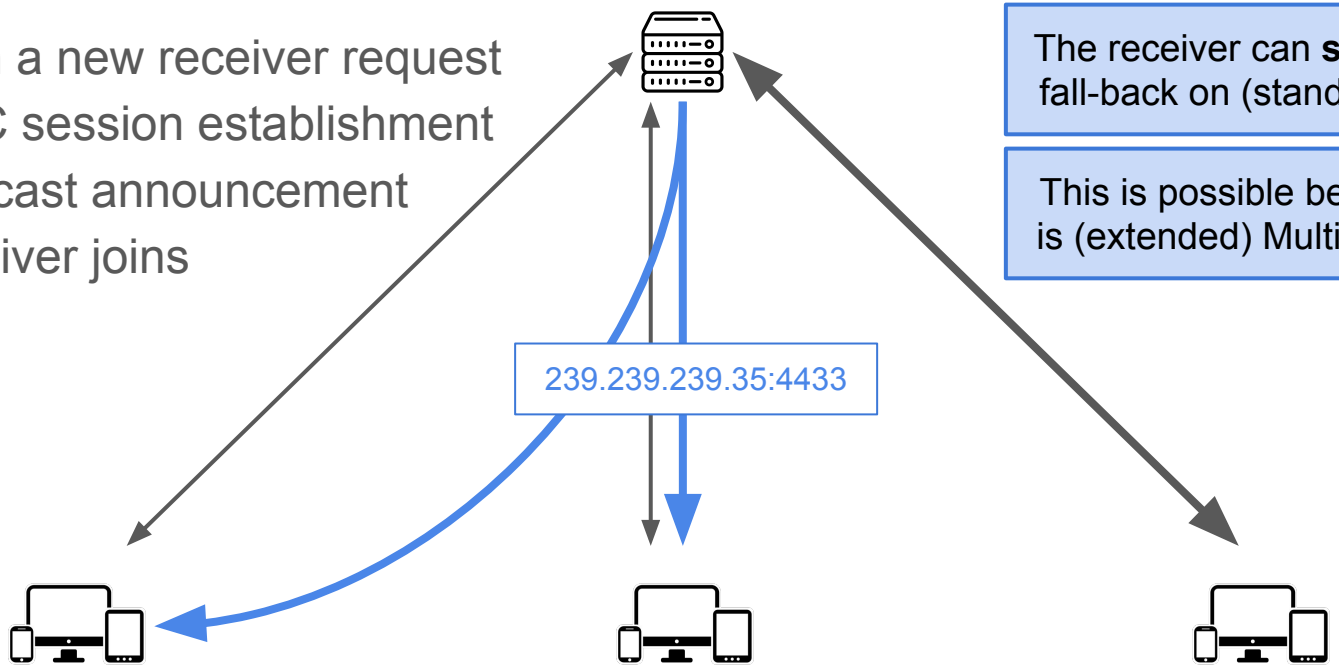
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement
- Receiver joins



Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

- Upon a new receiver request
- QUIC session establishment
- Multicast announcement
- Receiver joins

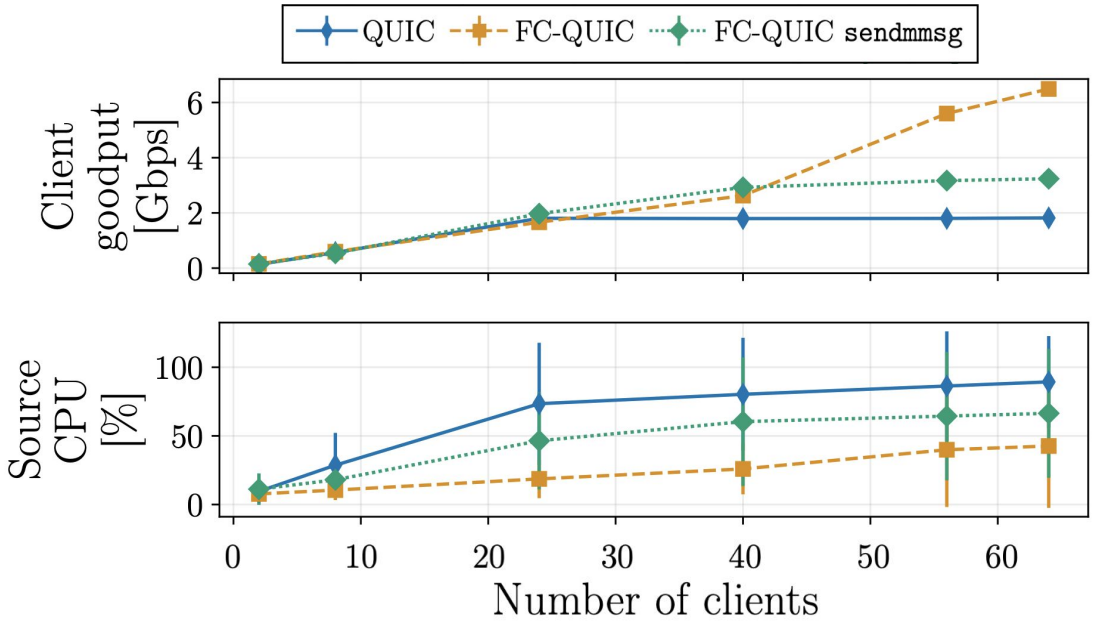


The receiver can **seamlessly** fall-back on (standard) QUIC

This is possible because this is (extended) Multipath QUIC

Some results on a (limited) benchmark setup

- 80 Mbps source bit-rate
- Baseline: UDP without connection
- The source scales with Flexicast QUIC
- Flexicast QUIC + *sendmmsg* improves compared to QUIC



And what about draft-jholland-quic-multicast?

- Some mechanisms from Jake Holland's draft are/must be used
  - Let's be clear: I do not try to reinvent the wheel here
- This draft attempts to provide a more detailed view on **how** to implement **flexible** multicast by leveraging **Multipath** QUIC
- Benefit from **experiment** with a **working** implementation
- Must find some middle ground if this goes further

However, the missing point remains the same:

And what about draft-jholland-quic-multicast?

- Some mechanisms from Jake Holland's draft are/must be used
  - Let's be clear: I do not try to reinvent the wheel here
- This draft attempts to provide a more detailed view on **how** to implement **flexible** multicast by leveraging **Multipath** QUIC
- Benefit from **experiment** with a **working** implementation
- Must find some middle ground if this goes further

However, the missing point remains the same:

Find that application that will make flexible multicast mandatory

Happy to get some feedback

- Extending Multipath QUIC to support flexible multicast
- Working implementation

If your use-cases may benefit from Flexicat QUIC:

- Discuss on the mailing list or slack
- Send us an email to collaborate: louis.navarre@uclouvain.be
- Implement Flexicast QUIC based on the draft for interop
- We are open to collaboration, do not hesitate to contact us!

We change Multipath QUIC to create Flexicast QUIC

- Remove the path probing **if** the destination address is a **multicast** address
- Different TLS key for each “path”
- The new “path” (let’s call it **flexicast flow**) is unidirectional
  - E.g., receivers send PATH_ACK on its unicast path only
- The flexicast flow is “just” another path
  - Retransmissions can either be on the unicast path or flexicast flow
  - Bottleneck receivers can fall-back on unicast seamlessly