draft-navarre-quic-flexicast
Flexicast QUIC

IETF 121 - Dublin
QUIC

Louis Navarre
Olivier Bonaventure

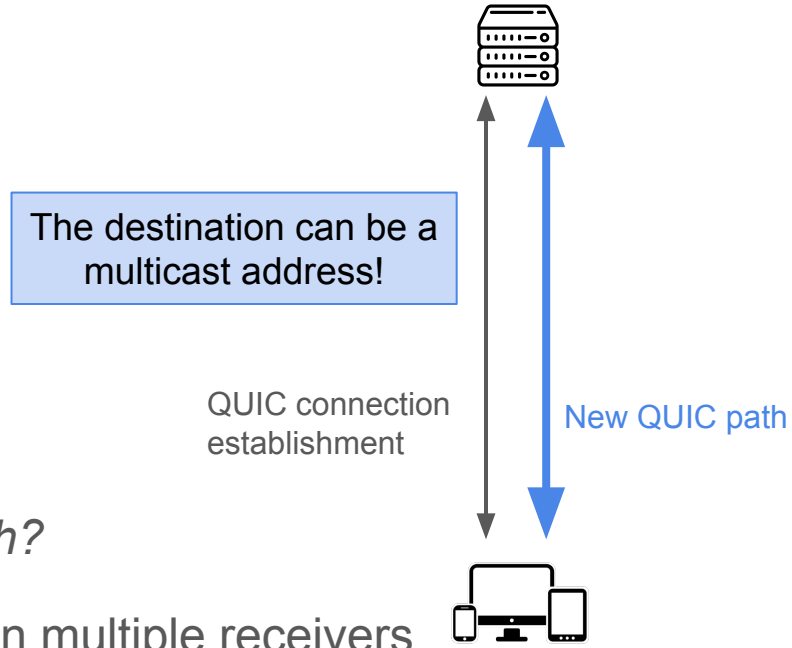
UCLouvain

Goal: enable flexible multicast inside QUIC

- Existing attempt to add multicast to QUIC: draft-jholland-quic-multicast
- 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*

Starting from Multipath QUIC…

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

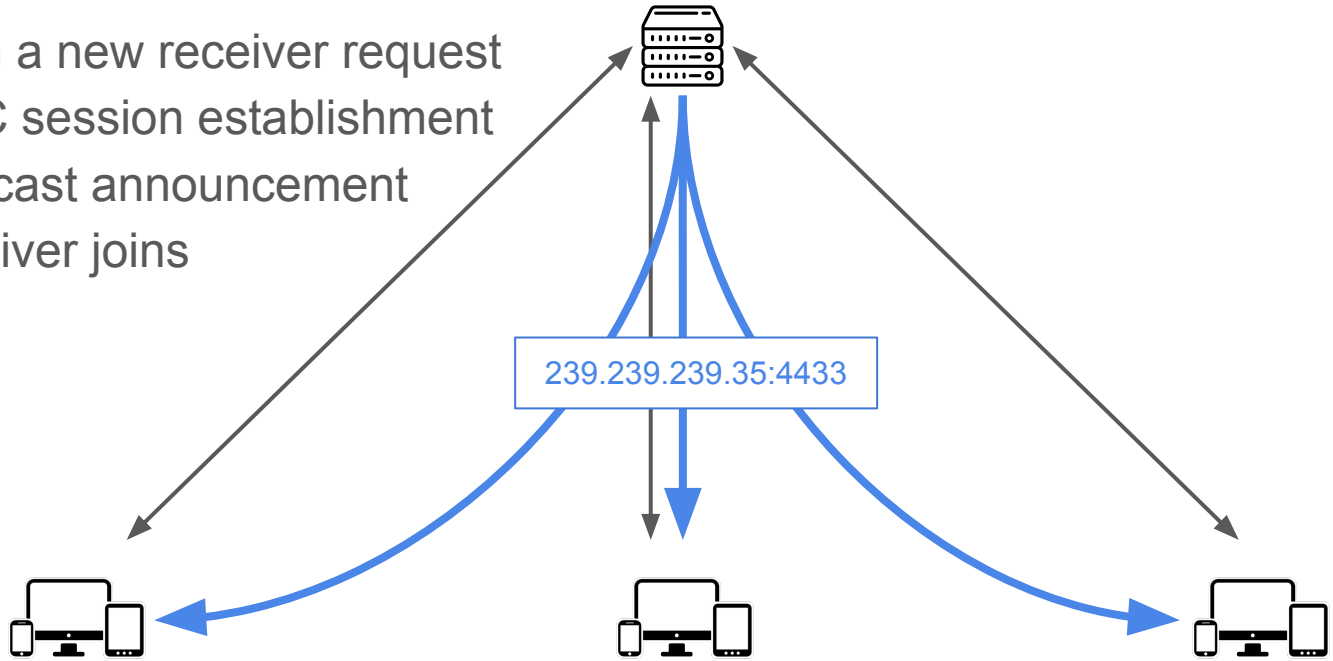


What if we use separate keys for each path?

⇒ The second path can be shared between multiple receivers

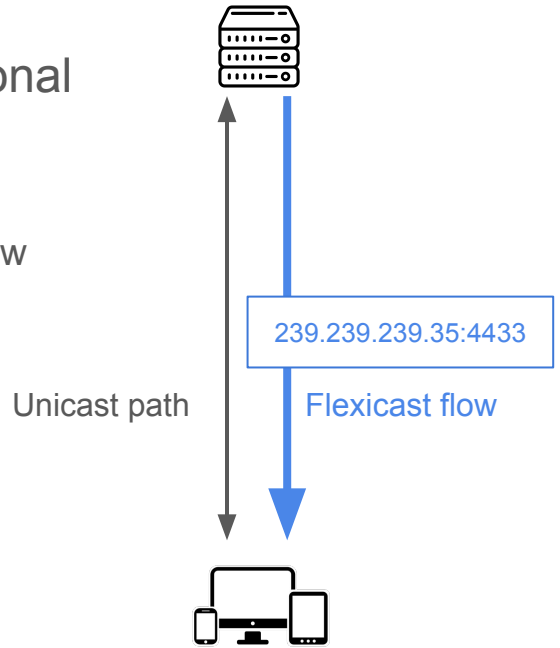
Flexicast QUIC: a separate bidirectional unicast path, and a shared unidirectional multicast flow

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



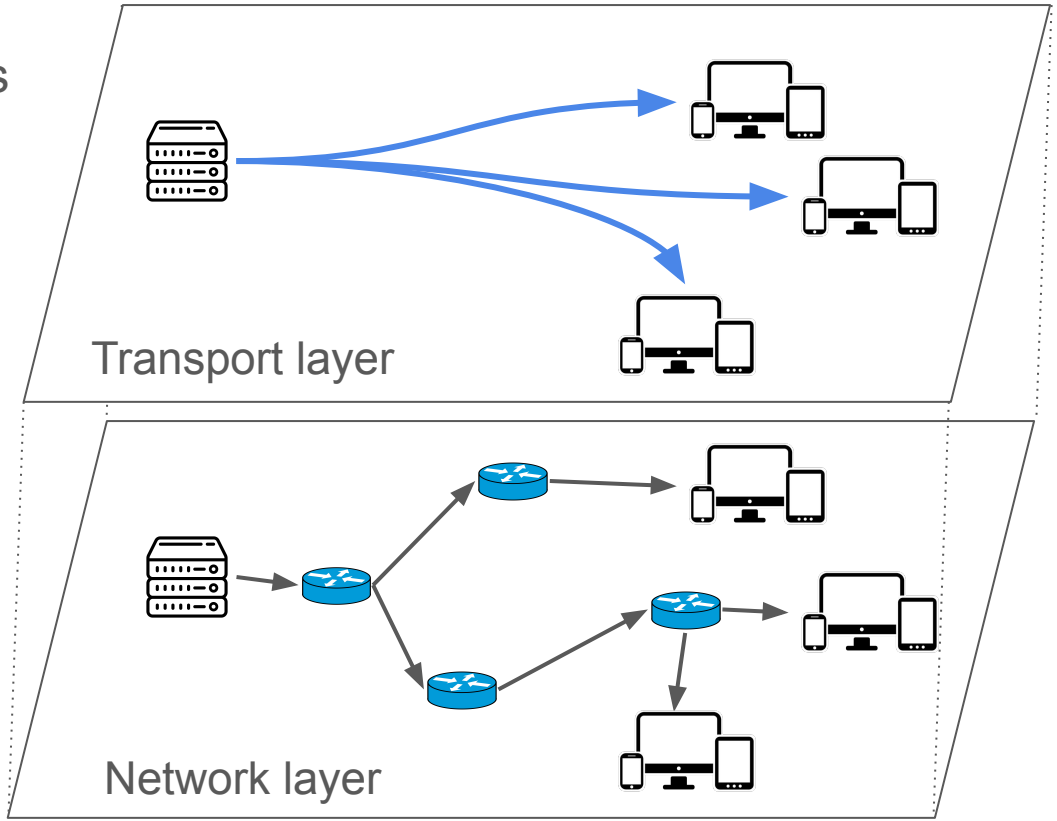
We change Multipath QUIC to create Flexicast QUIC

- 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



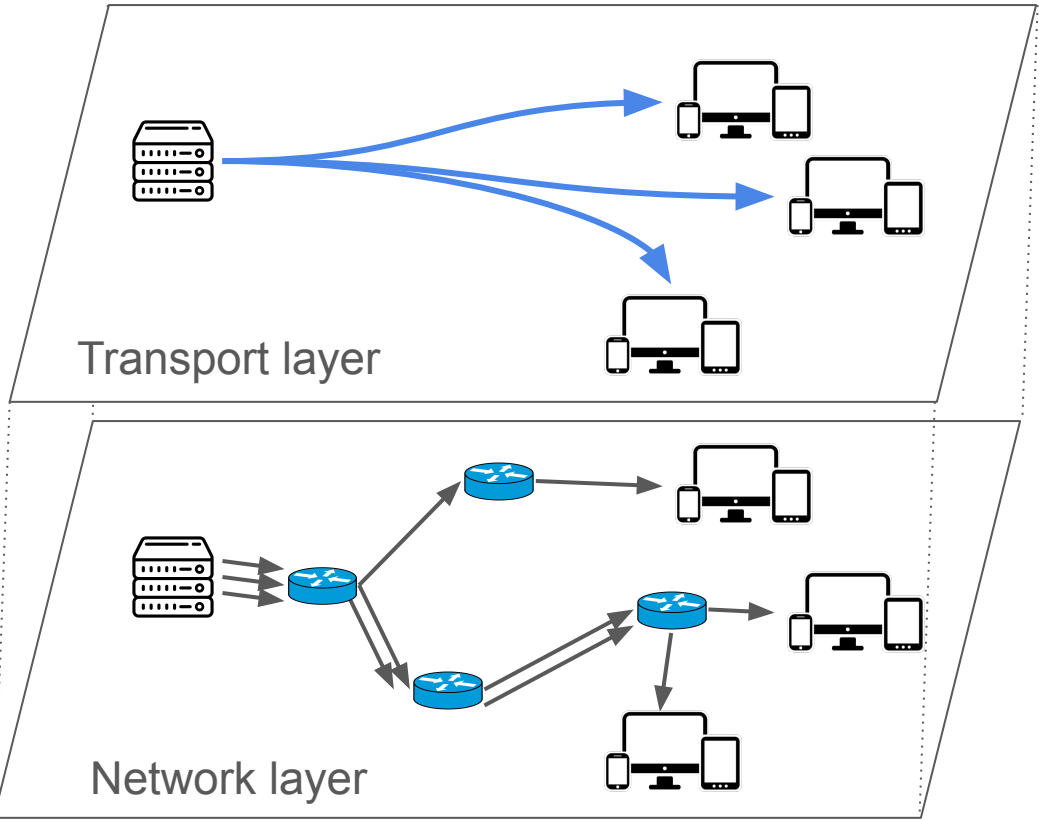
Carrying a flexicast flow over a multicast tree…

- Multicast routing properties
  - Fewer bytes in the network
- Source scalability
  - Generate, encrypt and send a single packet



… But the source could also replicate the packets

- ~~Multicast routing properties~~
  - ~~Fewer bytes in the network~~
- Source scalability
  - Generate, encrypt and **replicate the packet** E.g., using *sendmmsg*
- More efficient than generate, encrypt and send unicast packets



Some design aspects are taken from draft-jholland-quic-multicast

- 3 new frames
  - FC_ANNOUNCE: announce a new flexicast flow
  - FC_STATE: management of the receiver in the flow
  - FC_KEY: forwarding of the TLS keys of the flexicast flow
- Flow control to be discussed

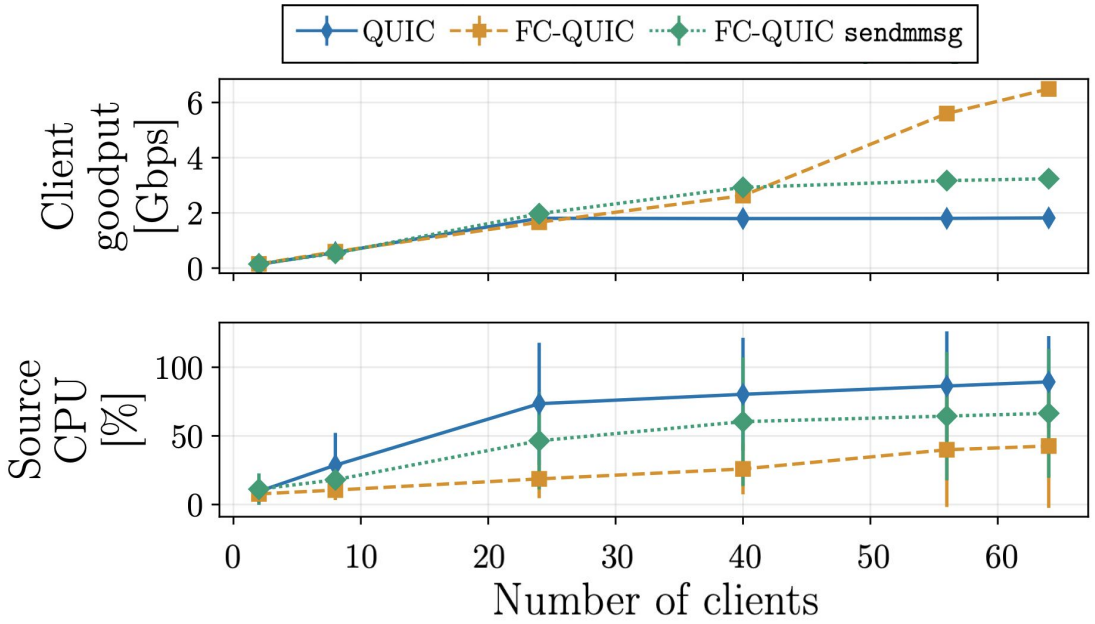
```
FC_ANNOUNCE Frame {
    Type (i),
    Flexicast Group ID (8..160),
    Authentication method (64),
    IP Version (8),
    Source IP (32, 128),
    Group IP (32, 128),
    UDP Port (16),
    Flexicast timer (64),
    [Public Key Length (i)],
    Public Key (..),
}

FC_STATE Frame {
    Type (i),
    Flexicast Group ID (8..160),
    Action (u64),
}
```

```
FC_KEY Frame {
    Type (i),
    Flexicast Group ID (8..160),
    [Key Length (i)],
    Key (..),
    Algorithm (u64),
}
```

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



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!