Warp in QUICR, Datagrams and Congestion

Christian Huitema

January 31, 2023

QUICR Prototype (QUICRQ)

- Implemented on top of Picoquic
  - Layered directly over QUIC Transport (RFC 9000)
  - Webtransport version “in progress”
  - Model MoQ as media streams: URI, Group of objets, objects
- Supports clients, origin server, relays
  - Clients can SUBSCRIBE to stream(s) or POST stream(s)
- Supports 3 transport modes
  - Single stream: all Groups and objects in sequence on single stream.
  - Datagrams: one QUIC datagram per “fragment of object”
  - WARP: one unidirectional stream per “group of objects”

Control stream per Media stream

- Control stream = bidirectional QUIC stream
  - From client to relay, or from client to origin
  - Carries control messages:
    - Subscribe to URI
    - Post URI / Accept
    - Last Group/Object indication (for clean termination)
- Closing the control stream closes the media stream
  - Similar behavior as Webtransport
- Negotiate Transport Mode, Media ID, Start Point (first group, object)
  - TODO: authentication

Datagram mode

- Fragment:
  - Media ID, Group ID, Object ID, Priority flags, Offset, Data
  - If first object of group, Number of Objects in Previous Group
- Relays manage cache of objects/fragments
  - Manage duplicates
  - Forward as soon as available
  - Track progress (full objects receive)
- Nodes implement ARQ:
  - Notification from Picoquic if datagram ACK or presumed lost
  - Repeat fragments if presumed lost

Stream Mode

- Send media as series of fragments on control stream
- Same fragments of datagrams, but in order
- Relays can forward fragments as soon as received

- Mostly used as reference point, to compare with other solutions

WARP

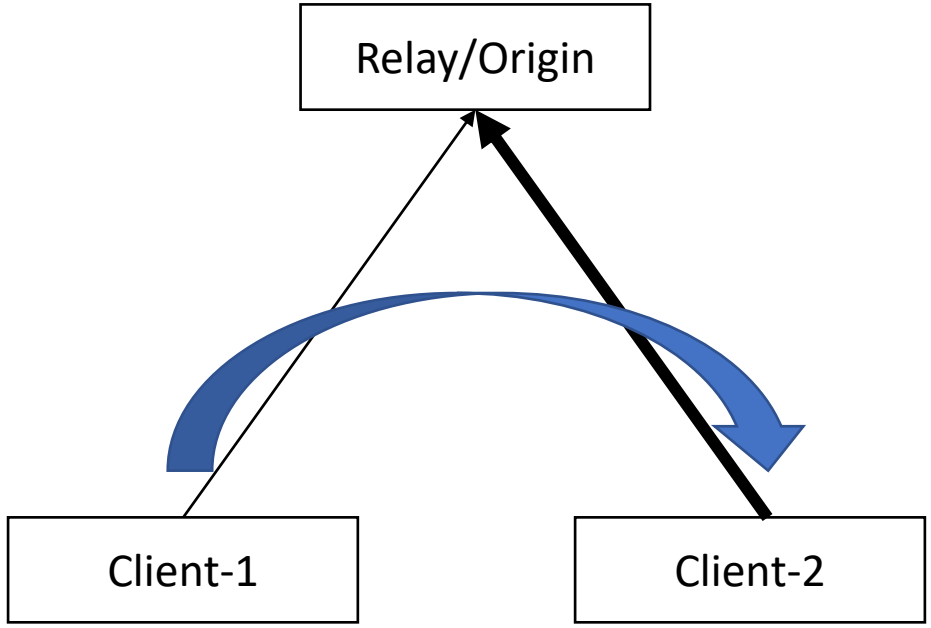
- One unidirectional stream per group of block
- Start with “WARP Header”:
  - Media ID, Group ID, Nb objects previous block
- Then series of “objects”
  - Object ID, Priority flags, Offset, Data
  - Whole object, not fragment
- Relays:
  - Cache objects
  - Forward as soon a possible

Picoquic specific: sending “just in time”

- Datagrams
  - Call back when ready to send a datagram (pacing, congestion, etc.)
  - Application chooses which stream to serve, what fragment to send
- Streams
  - Callback when ready to send on stream X (flow control, congestion, etc.)
  - Application places fragment in available buffer (similar to datagram)
  - Packet sent immediately
- Just-in-time sending relates to congestion control
  - If too congested to send all the data, send “place holder” messages, such as “object X in group G was skipped”

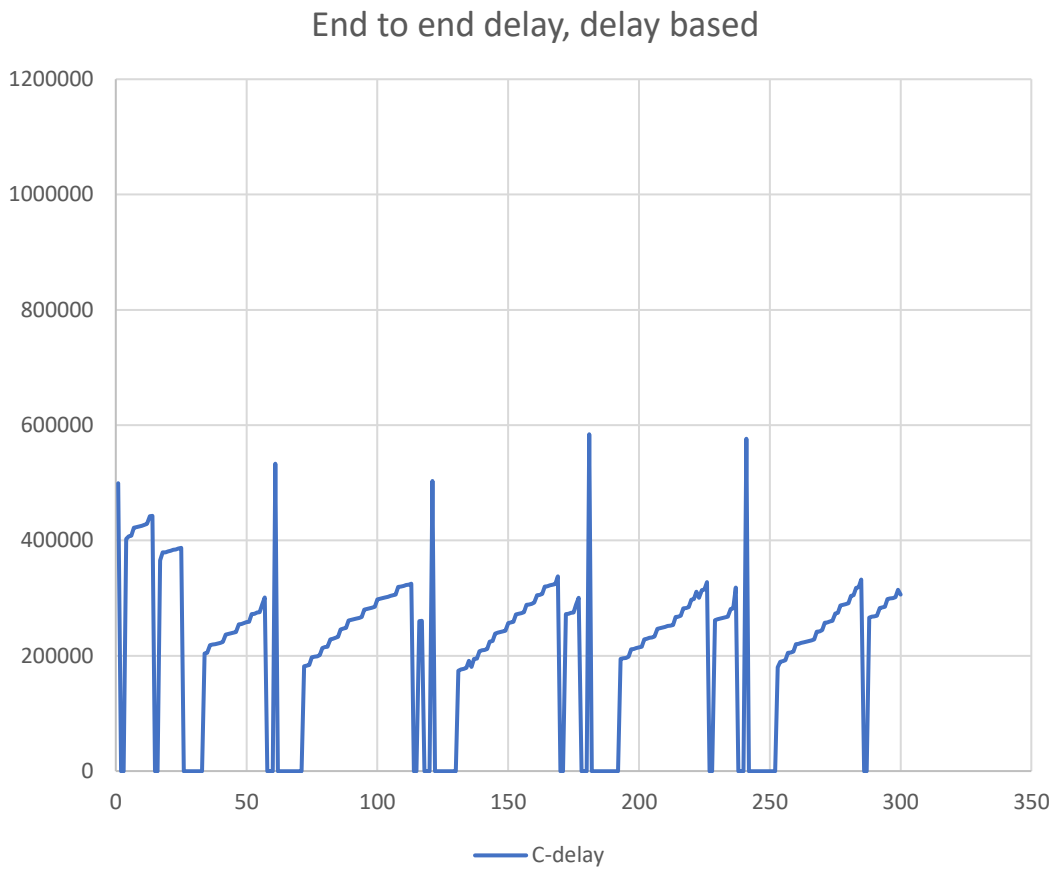
Testing congestion issues

- Simulate a simple network:
  - Client 1 posts URI to origin
  - Client 2 subscribes to URI
  - Media flows from client 1 to origin relay to client 2
- Bandwidth from client 1 to origin too low for media stream
- Test variety of transports, congestion control modes
  - Part of QUICR functional tests



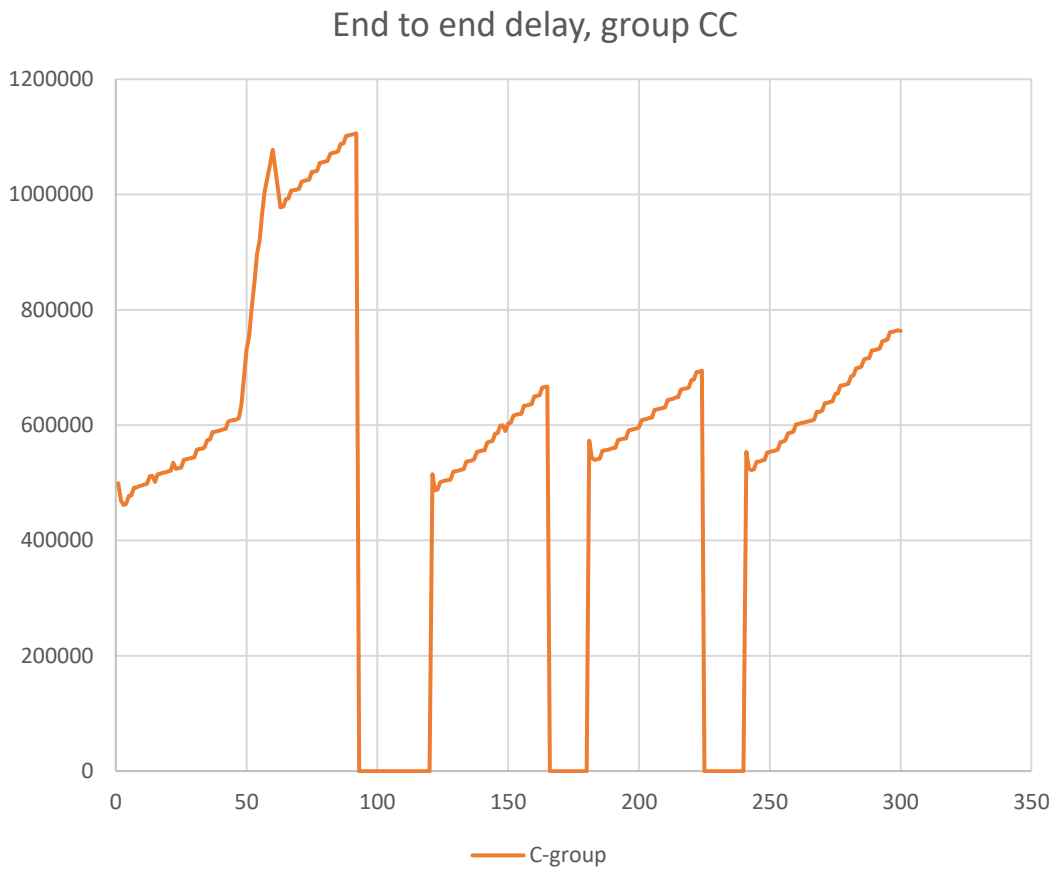
Congestion control mode: delay

- Monitor queues
  - Congested if > 5 objects
- Compute “cut priority” for the whole connection (all streams)
  - Increase if congestion persists
  - Decrease if congestion recedes
- If congested, drop objects with priority after cut
- Graph: WARP simulation, same priority for all GOP



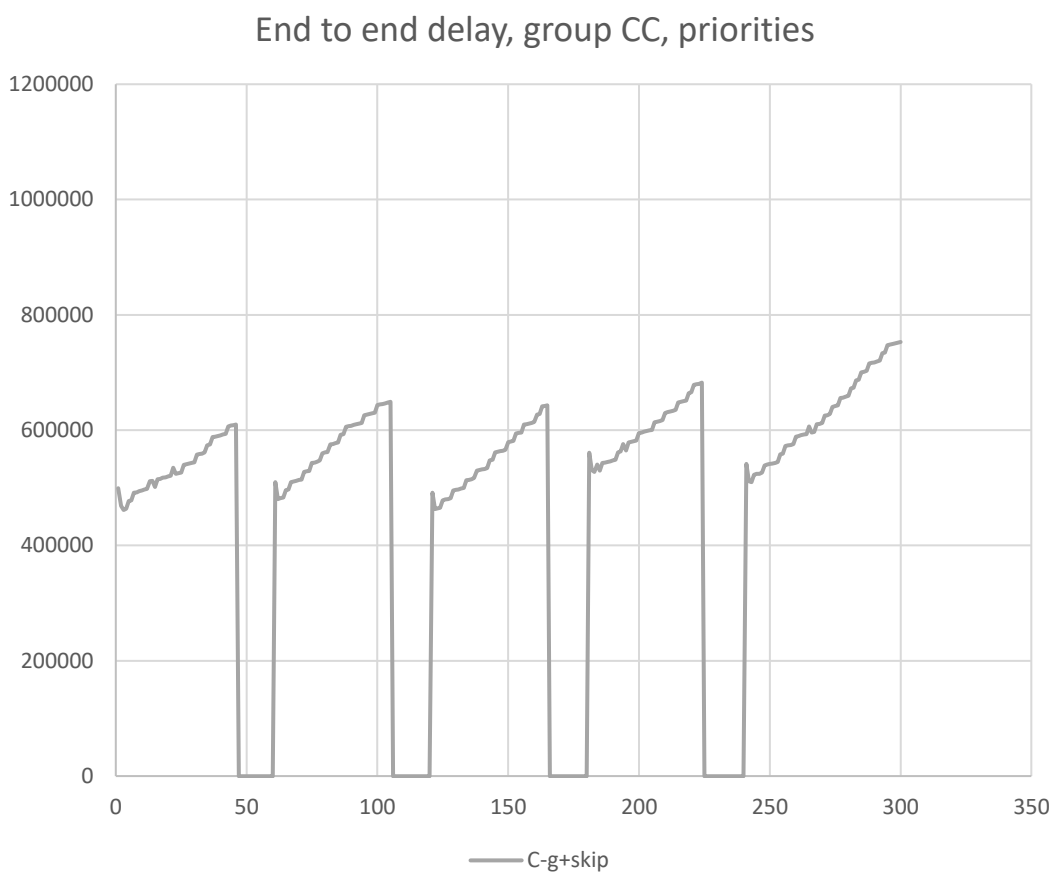
Congestion control mode: Group

- Monitor arrival of new GOB
  - As datagram fragments
  - Or, as unidirectional streams
- If more than 3 objects left in old GOB:
  - Detect congestion,
  - Skip these objects
- Graph: WARP simulation, same priority for all GOP



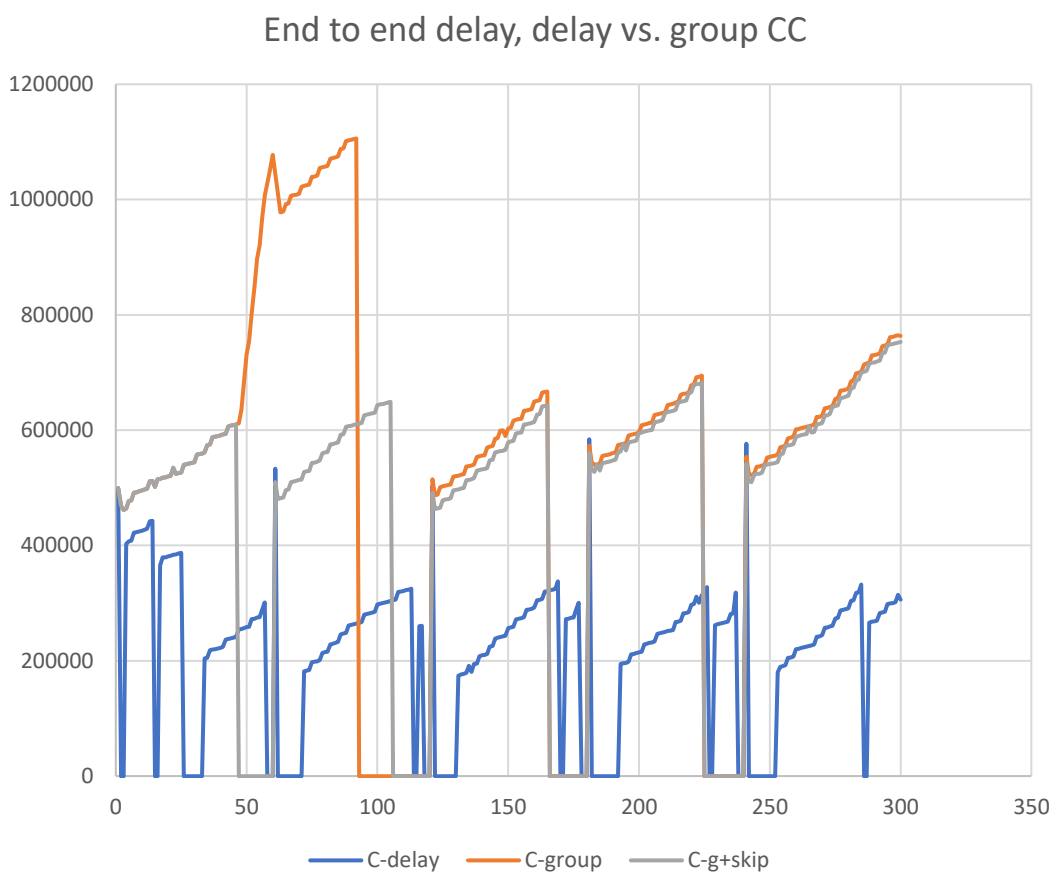
Congestion control mode: Group + skip ahead

- Default client behavior:
  - Delivery and consumption in order of transmission
- Skip ahead behavior:
  - If first object of new group received, skip to that
- Pair with WARP priorities:
  - If new group started, lower priorities of old groups



Congestion control mode comparisons

- Control for delay results in shorter delays (DOH!)
  - But improper priority settings can result in pixelation
- WARP reall works better with priorities and skip ahead
  - But maybe jarring experience of “accelerate every seconds then slow down”
- WARP is very robust
  - Restart cleanly after congestion



Summary

- Control stream per URI enables negotiation, control
- Pay attention to end condition to enable testing (and more)
  - How many objects per GOB
  - How many groups/objects in media stream
- Stream, datagrams, WARP
  - Almost identical when “all is fine”
  - Second order effects when packet loss (datagram > WARP > streams)
    - Can I speak another 30 minutes?
- Congestion happens on I-Frames
  - WARP is robust, but periodic slow down could be jarring
  - Delay based deliver better delays, about same drop rate