QUIC for games

Robin Marx

IETF 104 – QUIC DATAGRAM side-meeting – 27 March 2019

Game networking types: intro

- Deterministic lockstep: reliable, less latency sensitive
  - E.g., traditional real-time strategy games
- P2P with custom authority: mixed, medium latency sensitive
  - E.g., racing games
- Server-authority with prediction: unreliable, highly latency sensitive
  - E.g., First person shooters

https://www.researchgate.net/publication/228932160_1500_archers_on_a_288_Network_programming_in_Age_of_Empires_and_beyond

Unreliable + reliable + mixed

- Reliable: RPCs
  - Opened door, picked up weapon, changed loadout, …
- Unreliable
  - Client sends: Controller input (bitmaps)
  - Server updates simulation
  - Server broadcasts: Position/rotation/velocity/… updates (3D vectors)
- Mixed
  - Delta-compression for the unreliable updates
  - Need to know which base state to base deltas on (need ACKs)

https://gafferongames.com/post/networked_physics_in_virtual_reality/

What is reliable?

- Typical: with retransmits when deemed lost
- Alternative: keep repeating until ACKed (bandwidth vs responsiveness tradeoff)
- Alternative: just need ACKs, no retransmits (e.g., delta-state is the last one we know the peer has received)
    - Draft currently has this as a MUST + that they can be delayed (5.2) → not for games

https://gafferongames.com/post/reliable_ordered_messages/

Ordering…

- Games don’t care about old data if they’ve already applied the new one

- DATAGRAMS need a strict ordering
    - Either add this in the application-level payload (overhead!)
    - Or use existing QUIC packet numbers for this
        - But then these need to bubble-up to the application-level somehow

Other performance aspects

- Bandwidth overhead limiting (e.g., DATAGRAM headers)
  - Typically bundle large amount of very small frames into 1 packet
  - Length = 0 means : just keep reading till stream is closed
    - OR mark full stream as unreliable and drop DATAGRAM entirely?

- Prioritization (e.g., area of interest)
  - Mapping of game-streams to QUIC streams might not be simple
  - Question is where to abstract this: application-layer or transport-layer?

- Also: things like match making, lobbies, chat, voice chat, …
  - Usually done via separate stacks now, can/should we allow combination?
  - Shared congestion context + prioritization could be quite interesting here!

Tentative summary for current DATAGRAM draft

- Need for ACKs and sending should be relaxed / configurable
- Think about exposing ordering information (recommendation?)
- Allow to cut as much overhead as possible (e.g., length = 0)

- Just the tip of the iceberg
  - But I need more sleep for the rest…

References

- Glenn Fiedler
  - https://gafferongames.com/
  - https://gafferongames.com/post/reliable_ordered_messages/
  - https://gafferongames.com/post/networked_physics_in_virtual_reality/
- Steam open-source UDP protocol
  - https://steamcommunity.com/groups/steamworks#announcements/detail/1791775741704351698
  - https://github.com/ValveSoftware/GameNetworkingSockets
    - Actually mentions QUIC at various points (ACK format, gQUIC crypto)