channeler

Interpeer Project
https://interpeer.io/

Reliable multi-channel communications
over unreliable networks

Goals

- Use-case: peer-to-peer video streaming
  - Very high bandwidth requirements with 4K+
  - Very low latency requirements
  - Must be usable on UDP (NAT-piercing)
  - Must communicate congestion to application layer
    - Frame skip is more acceptable UX than jitter
- Generalized for other use-cases
- (Optional) transport encryption

Requirements

- Multiple (mostly) independent virtual channels
- Multiple connections of unknown reliability
- Capable of application on NAT-pierced UDP
- Congestion prediction and recovery
- Packet failure tolerant
- Communication with application layer:
  - Inform of reliability changes
  - Allow application to switch mitigation strategies on-the-fly
- Transport encryption

Multiple channels

- Packet loss in one virtual channel must not impact other channels
- Application can dedicate channel to purpose, e.g. data vs OOB
- Channel characteristics individually configurable
  - Packet loss prevention & recovery
  - Connection selection, if any

Multiple connections

- Available connections are unknown in advance (TCP, UDP, local multicast, …)
- Connections may be more or less reliable (TCP vs. UDP, link-level: cable vs. wireless)
- Receiver: must be able to seamlessly switch connections on a packet-by-packet basis
- Sender: must be able to decide packet-by-packet which connection to use
  - In practice: bind channel to connection; change binding

Reliability

- For:
  - UDP transport
  - Congestion prediction & recovery
- Implement own transport reliability
  - Resend lost packets **on request** vs. TCPs mechanism
  - Loss **prediction** via LEDBAT-like mechanism
  - Loss **recovery** via TCP-Cubic-like rampup
  - Loss **prevention** via forward error coding
    - With configurable replication factor
  - Loss **reporting** to application

Application on UDP

- Required for NAT-piercing
- Compact packet envelopes
- Configurable MTU
- Message splitting & reassembly

Transport encryption

- Initially:
  - reserve a channel for encryption handshakes
  - DTLS: applies transport encryption to all packets, across all channels
    - Lost packet leads to DTLS record processing delays
- Future:
  - reserve a channel for encryption handshakes
  - WireGuard/NOISE: message decryption is not stalled on sequence numbering
  - Give each channel unique counter

Interpeer Project
https://interpeer.io/