

RUSH

Reliable (Unreliable) Streaming protocol

Kirill Pugin, Facebook

Motivation

- Applications have different latency requirements
 - Live streaming of soccer match may be ok with 10-30 seconds latency
 - Interactive live streams like Gaming would benefit from lower latency (< 5 seconds)
- Extensibility
 - New audio/video codec support
 - New client-server interactions
 - Multi-track support, including captions

Motivation

- Reliability
 - Disconnects can happen due to network change, errors or server maintenance
- Quality
 - Better signals from network to adjust audio/video bitrate to network conditions (Adaptive Bitrate selection)

Motivation (continued)

- RTC – focused on P2P and low latency upload, doesn't give a choice between latency and quality.
- RTMP – old, not flexible – no new codec support, some implementations don't support reconnect.
- DASH – doesn't allow per-frame level control - hard to control latency.

RUSH

RUSH is a bidirectional application level protocol designed for live video ingestion that runs on top of QUIC.

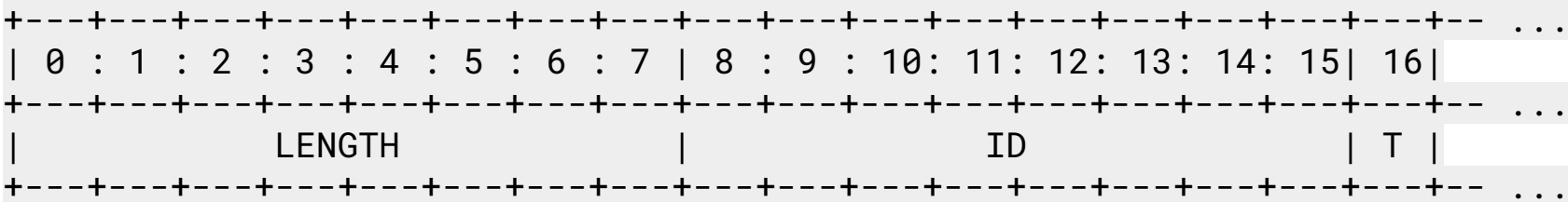
RUSH was built as a replacement for RTMP (Real-Time Messaging Protocol) with the goal to provide support for new audio and video codecs, extensibility in the form of new message types, and multi-track support.

In addition, RUSH gives applications option to control data delivery guarantees by utilizing QUIC streams.

Wire format

Client and server exchanges information using frames. Frames can be different types and data passed within a frame depends on its type.

Generic frame format:



Wire format

- **LENGTH(64)**: Each frame starts with length field, 64 bit size that tells size of the frame in bytes.
- **ID(64)**: 64 bit frame sequence number, every new frame **MUST** have a sequence ID greater than that of the previous frame within the same track.
- **TYPE(8)**: 1 byte representing type of the frame.

Frames

RUSH defines 7 frames types:

- Connect frame
- Connect Ack frame
- End Of Video frame
- Error frame
- Audio frame
- Video frame
- GoAway frame

How it works

There are two modes defined:

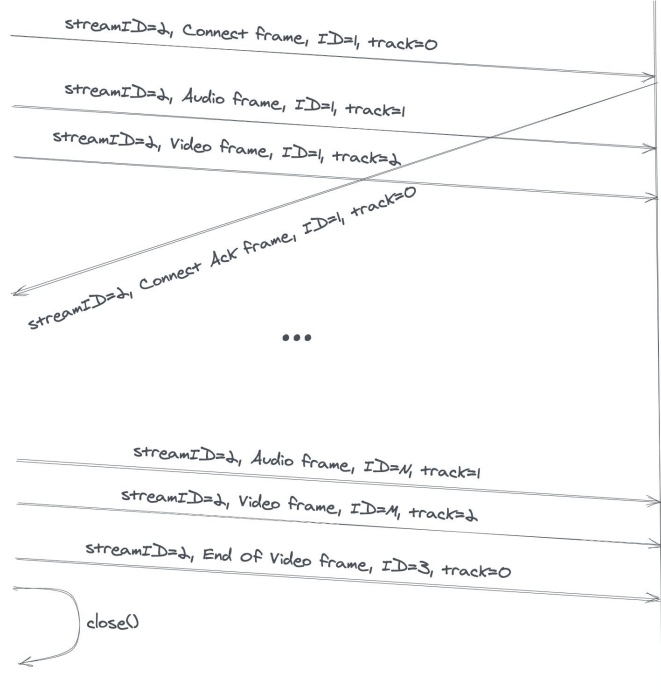
- **Normal mode:** RUSH uses one bidirectional QUIC stream to send data and receive data. Using one stream guarantees reliable, in-order delivery - applications can rely on QUIC transport layer to retransmit lost packets. The performance characteristics of this mode are “similar” to RTMP over TCP.

Normal mode

Normal Mode

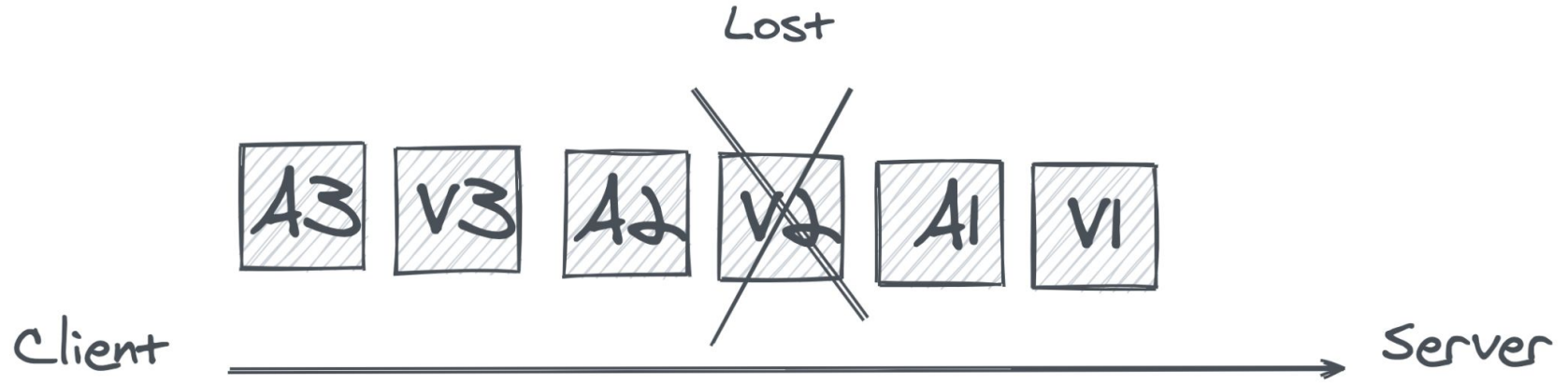
Client

Server



- Client sends **Connect** frame on bidirectional QUIC stream
- Client sends audio and video data on the same QUIC stream
- Only one video can be send on the same Connection
- Server replies with **Connect Ack** frame on the same QUIC stream
- Frames arrive **in order**
- Client sends **End Of Video** frame to indicate that video is done

Normal mode (what can go wrong?)



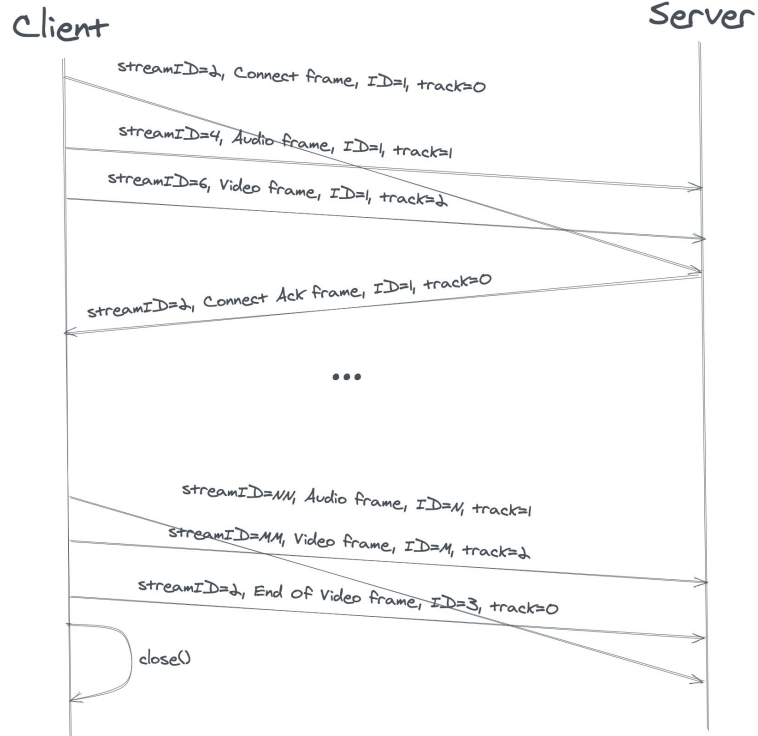
If **V2** frame is lost all frames sent after it will be not available to the server, until **V2** retransmitted - this is variation of head of line blocking and can affect latency and introduce jitter.

How it works

- **Multi-stream mode:** To address head of line blocking and also to give more control to application over delivery guarantees, in multi-stream mode, every new frame is sent on new QUIC bidirectional stream. Since QUIC streams are independent of each other this allows server receive data as it arrives and not wait for retransmissions of lost packets.

Multi-stream mode

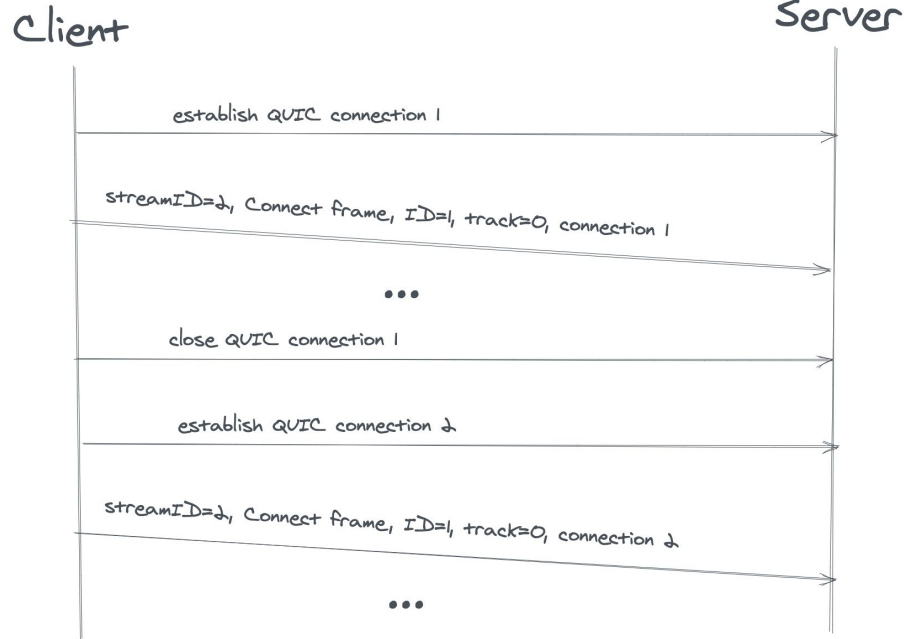
Multi-stream Mode



- Frames arrive out of order
- Server uses frame IDs within a track to detect missing frames, it's up to server to "restore" order
- Client can stop retransmission by resetting the corresponding QUIC stream

How it works (reconnect)

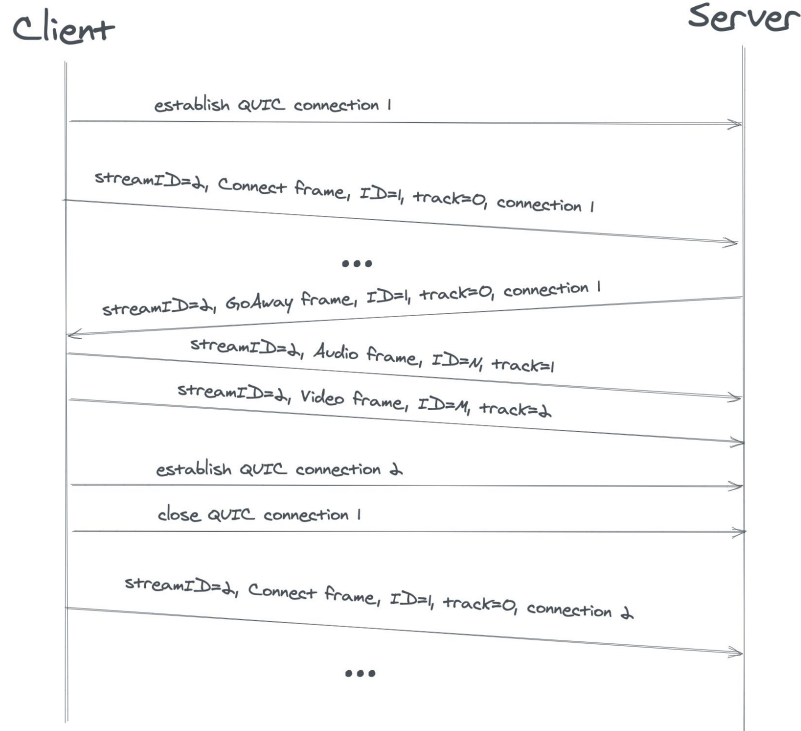
Reconnect by client



- Client opens new QUIC connection
- Client closes currently used QUIC connection.
- Client follows normal connect flow and continue sending data on new QUIC connection

How it works (reconnect)

Reconnect by server



- Server sends GoAway method
- Client may send frames on current connection
- Client establishes new connection
- Client follows normal connect flow and continues sending data on new QUIC connection
- **NOTE:** server may close connection after sending GoAway, but before client finished sending frames on that connection - this may result in data loss

Questions?

More Questions?

Come to the Video Ingest over QUIC side meeting

Friday, 7/30 18:00 UTC

Video conference details will be available on the IETF 111 Side Meetings Wiki

<https://trac.ietf.org/trac/ietf/meeting/wiki/111sidemeetings>