Workgroup: Faster-Than-Light Standardization Effort
Internet-Draft: draft-ftl-specification-00
Published: 5 January 2021
Intended Status: Informational
Expires: 9 July 2021
Author: M.C. Casadevall, Ed.
Indepedent

Faster-Than-Light Streaming Protocl Specification

Abstract

With the demise of Microsoft's Mixer, the future of the Faster-Than-Light (FTL) streaming protocol has been left in doubt. As the Internet's first practical subsecond streaming protocol, several successors to Mixer have decided to re-implement FTL from the original SDK and notes. While Mixer's original FTL specification had a de-facto specification in the form of ftl-sdk, the source code was in-complete, and several aspects of the FTL were left undocumented.

In an effort to keep FTL viable and cross-service compatible, this specification denotes a canonical implementation of FTL, handshake protocols, WebRTC notes, and all relevant information as relating to FTL with the hope that FTL may still be continued as a vechile for low latency video streaming over the Internet.

Status of This Memo

Copyright Notice

Table of Contents

Author's Address

1. Introduction and Overview

This specification covers several components of the FTL protocol stream, and is primarily derieved from the implementation used at Mixer and the freely available source code in ftl-sdk. This document details the handshaking protocol known as Charon, the SRTP ingest behaviors, and defines a recommended ingest->endpoint streaming protocol, as well as notes in regards to implementation of the last mile WebRTC connections.

FTL was specifically designed with the following objectives in mind which is must handle at all times.

- Real-world 500 millsecond delay for streamer to receiver broadcast under normal cirmstances at 30 FPS or more
  - At the time of it's implementation, 720p streaming was considered the best possible for most users, however, advancements in technology have allowed for 1080p streaming.
- That FTL has be technology neutral; it should be able to use any video or audio codec as supported by WebRTC (or another last mile technology) independently. The original FTL implemntation used VP8 and Opus, later ones used H.264, keeping with the original intent.
- It is expected that to reduce latency, an FTL deployment has multiple ingest and points of presenses for client connections. A stream connects to one ingest point, and then data is routed to points of presence as necessary.
- FTL end-points must support being behind anycast, as well as use of STUN, and TURN if necessary
- Use of standards based technology for use in a web browser with no additional software or downloads for viewers

As originally designed, the following criticera were also kept in mind, although not realized at least during the initial implementation phases.

- Use of IPv6
  - IPv6 deployment and usage was considered highly desirable for multiple reasons, primarily to simplify routing and mandated multicast support; as implemented, there should be no known IPv6 problems, but the protocol never tested either.

2. Design Tradeoffs/Limtations

TBD

3. One-to-Many WebRTC

TBD

4. Charon Negotation Protocol

Charon handles negotation of RTP streams to Styx, and acts as an out of band signaling method for FTL stream behavior. The Charon connection MUST be kept-alive at all times as a TCP/IP connection on port 8084 (MC: should apply for IANA number). If the TCP/IP connection is reset, the broadcaster MUST assume that the ingest point of presense has become unavailable, and begin clean-up and teardown. Likewise, the ingest daemon MUST begin teardown and disregard any UDP traffic from the streamer upon Charon connection loss.

The Charon protocol is built upon ASCII verbs with optional arguments. The lifecycle of this connection under normal cirmstances is as follows.

- Broadcaster connects to ingest on TCP/IP 8084
- Broadcaster gets HMAC authentication
- Broadcaster generates HMAC authentication based off streamer connection ID and channel idea
- Broadcaster sends CONNECT, combined with stream paramters.
- Ingest sends FTL_OK or FTL_REJECT based on settings
- If FLT_OK, Broadcaster sends RTP streams to the media port indicated by the ingest

4.1. Message Format

4.2. Response Codes

4.3. Charon Verbs

4.3.1. HMAC

TDB

4.3.2. CONNECT

TDB

4.3.2.1. ProtocolVersion

TBD

4.3.2.2. VendorName

TBD

4.3.2.3. VendorVersion

TBD

4.3.2.4. Video

TBD

4.3.2.5. VideoCodex

TBD

4.3.2.6. VideoHeight

TBD

4.3.2.7. VideoWidth

TBD

4.3.2.8. VideoPayloadType

TBD

4.3.2.9. VideoIngestSSRC

TBD

4.3.2.10. Audio

TBD

4.3.2.11. AudioCodec

TBD

4.3.2.12. AudioPayloadType

TBD

4.3.2.13. AudioIngestSSRC

TBD

4.3.3. DISCONNECT

TBD

5. Styx Protocol Ingest Behavior

TBD

6. Babel Transcoding Behavior

TBD

7. WebRTC Last Mile Negotations

TBD

Author's Address

Michael Casadevall (EDITOR)
Indepedent
Jersey City,
United States
Email: michael@casadevall.pro