
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. <i>Indepedent</i>

Faster-Than-Light Streaming Protocol 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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 July 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

- 1. Introduction and Overview
- 2. Design Tradeoffs/Limitations
- 3. One-to-Many WebRTC
- 4. A Note on SSRCs
- 5. Charon Negotiation Protocol
 - 5.1. Message Format
 - 5.1.1. On Linebreaks
 - 5.2. Response Codes
 - 5.3. Charon Verbs
 - 5.3.1. HMAC
 - 5.3.2. CONNECT
 - 5.3.2.1. ProtocolVersion
 - 5.3.2.2. VendorName
 - 5.3.2.3. VendorVersion
 - 5.3.2.4. Video
 - 5.3.2.5. VideoCodex
 - 5.3.2.6. VideoHeight
 - 5.3.2.7. VideoWidth
 - 5.3.2.8. VideoPayloadType
 - 5.3.2.9. VideoIngestSSRC
 - 5.3.2.10. Audio
 - 5.3.2.11. AudioCodec
 - 5.3.2.12. AudioPayloadType
 - 5.3.2.13. AudioIngestSSRC
 - 5.3.3. DISCONNECT
 - 5.3.4. PING

[6. Styx Protocol Ingest Behavior](#)

[7. Babel Transcoding Behavior](#)

[8. WebRTC Last Mile Negotiations](#)

[Author's Address](#)

1. Introduction and Overview

This specification covers several components of the FTL protocol stream, and is primarily derived 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 it must handle at all times.

- Real-world 500 millisecond delay for streamer to receiver broadcast under normal circumstances at 30 FPS or more
 - At the time of its implementation, 720p streaming was considered the best possible for most users, however, advancements in technology have allowed for 1080p streaming.
- That FTL has to 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 implementation 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 presence 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 criticisms 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/Limitations

TBD

3. One-to-Many WebRTC

TBD

4. A Note on SSRCs

TBD

5. Charon Negotiation Protocol

Charon handles negotiation 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). Upon establishment of a **CONNECT**, a client must send **PING** messages once every five seconds, or the server **MAY** time-out the connection.

It is **RECOMMENDED** that the server wait 10 seconds before assuming that a client has hung and hanging up. **PING** messages must be responded to with a 201 respond code. The server should send **5XX Timeout** message before disconnecting. (Ed: fix this and actually put the client behavior)

If the TCP/IP connection is reset, the broadcaster **MUST** assume that the ingest point of presence 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 circumstances 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
- Broadcaster starts RTP streams per Ingests response
- Every five seconds, a **PING** response is sent, ingest replies with 201
- Broadcaster sends **DISCONNECT** for orderly shutdown

5.1. Message Format

Charon is directly modeled on SMTP commands and response codes. As such, commands take the form of a verb, followed by a three digital response. An example exchange looks as such:

```
Client: PING\r\n\r\n
Server: 201 Ping OK!\r\n\r\n
```

For legacy reasons, linebreaks in Charon are encoded as '\r\n\r\n' (hex 0x0D0A0D0A), with an unusually complex implementation detail. See the section "On Linebreaks" for more details.

For commands that do not need additional meta-data, the server SHALL process them immediately upon receiving a linebreak, otherwise, a multiline format using RFC822 headers is defined, with the message ended with a single period. Unknown headers MUST be disregarded by the server. This message exchange looks as follows:

```
Client: CONNECT 1234-myhashhere\r\n\r\n
      C: Video: true\r\n\r\n\r\n
      C: Audio: true\r\n\r\n\r\n
      C: .\r\n\r\n\r\n
Server: 200 Send UDP!\r\n\r\n\r\n
```

The Charon protocol does not allow for transmission of binary data without encoding. If necessary, it is RECOMMENDED that binary data base encoded in Base64.

5.1.1. On Linebreaks

The reference implementation of FTL uses the message signature '\r\n\r\n' as an end of line marker. This is an intentional implementation detail due to an unintentional similarity between FTL's CONNECT message, and the HTTP CONNECT proxy command, and Microsoft discovered that some commercial firewalls would intercept Charon's messages. The original justification was left as this comment in `ftl_helpers.c`

```
/*
    Please note that throughout the code, we send "\r\n\r\n", where a
    normal newline ("\n") would suffice. This is done due to some
    firewalls / anti-malware systems not passing our packets
    through when we don't send those double-windows-newlines. They
    seem to incorrectly detect our protocol as HTTP.
*/
```

A problem however arises that earlier implementations of FTL used "\n" as a newline indicator. While it is unlikely that any of these legacy FTL clients are still in use, clients and ingests must use the following behaviors

Charon servers MUST disregard any empty newline, and treat "\r\n" and "\n" as identical for the purposes of message parsing. In effect, all the following are the same.

```
CONNECT 1234-hash\n
Option1: 1234\n
Option2: 1234\n
.\n

CONNECT 1234-hash\r\n
Option1: 1234\r\n
Option2: 1234\r\n
.\r\n

CONNECT 1234-hash\r\n
Option1: 1234\r\n\r\n
Option2: 1234\r\n\r\n
.\r\n\r\n
```

Charon clients are MUST to use '\r\n\r\n' when transmitting commands over clear text.

5.2. Response Codes

5.3. Charon Verbs

5.3.1. HMAC

TDB

5.3.2. CONNECT

TDB

5.3.2.1. ProtocolVersion

TBD

5.3.2.2. VendorName

TBD

5.3.2.3. VendorVersion

TBD

5.3.2.4. Video

TBD

5.3.2.5. VideoCodex

TBD

5.3.2.6. VideoHeight

TBD

5.3.2.7. VideoWidth

TBD

5.3.2.8. VideoPayloadType

TBD

5.3.2.9. VideoIngestSSRC

TBD

5.3.2.10. Audio

TBD

5.3.2.11. AudioCodec

TBD

5.3.2.12. AudioPayloadType

TBD

5.3.2.13. AudioIngestSSRC

TBD

5.3.3. DISCONNECT

TBD

5.3.4. PING

TBD

6. Styx Protocol Ingest Behavior

TBD

7. Babel Transcoding Behavior

TBD

8. WebRTC Last Mile Negotiations

TBD

Author's Address

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