

動画を送ろう!!

RTMP

2019 あれくま

目次

1. はじめに.....	6
2. RTMP とは.....	7
2.1. RTMP で出来ること・出来ないこと.....	7
2.2. 実装について.....	9
3. RTMP 概観.....	10
3.1. TCP.....	10
3.2. RTMP Chunk Stream Protocol.....	10
3.3. RTMP Message.....	11
4. RTMP Chunk Stream.....	14
4.1. バイナリフォーマットについて.....	14
4.2. TCP での接続.....	14
4.3. ハンドシェイク.....	14
4.4. チャンクの送受信.....	18
4.5. RTMP Control Message.....	25
4.6. RTMP Chunk Stream のまとめ.....	28
5. RTMP Message.....	31
5.1. RTMP Message Format.....	31
5.2. RTMP Message の種類.....	31
5.3. User Control Message (タイプ ID:4).....	32
5.4. Data Message (タイプ ID:15 または 18).....	37
5.5. Audio Message (タイプ ID:8).....	39
5.6. Video Message (タイプ ID:9).....	41
5.7. Shared Object Message (タイプ ID:16 または 19).....	43
5.8. Aggregate Message (タイプ ID:22).....	43
5.9. Command Message (タイプ ID:17 または 20).....	44
5.10. NetConnection コマンドと NetStream コマンド.....	45
5.11. connect コマンド (NetConnection).....	46
5.12. createStream コマンド (NetConnection).....	50
5.13. deleteStream コマンド (NetConnection).....	51
5.14. close コマンド (NetConnection).....	52
5.15. その他のコマンド (NetConnection).....	52
5.16. onStatus コマンド (NetStream).....	54
5.17. play コマンド (NetStream).....	55
5.18. play2 コマンド (NetStream).....	56
5.19. pause コマンド (NetStream).....	58
5.20. seek コマンド (NetStream).....	59

5.21. receiveAudio コマンド (NetStream)	59
5.22. receiveVideo コマンド (NetStream)	60
5.23. publish コマンド (NetStream).....	60
6. AMF0/3	62
6.1. AMF0	62
6.2. AMF3	66
6.3. RTMP Message での型の扱いについて	72
7. RTMP で通信しよう	74
7.1. 共通処理	74
7.2. 切断処理	76
7.3. クライアントからサーバーへ動画配信したい.....	76
7.4. 動画の視聴をしたい(サーバーからクライアントへの動画の配信をしたい)80	
8. FLV ファイル	82
8.1. FLV ファイルの構造	82
8.2. FLV ファイルまとめ	84
9. 参考文献.....	85
9.1. ドキュメント類.....	85
9.2. ツール類	87

1. はじめに

この本はいまさらながら RTMP を解説する本です。

一般ユーザーによる動画の生配信が一般的になり、動画配信サービスも沢山ある今日このごろ。

動画配信の視聴と言えはちょっと前までは Flash がよく使われておりましたが、今となってはもう過去の物となりはて、HTML5(というか Web ブラウザだけ) で視聴ができる便利な世の中になりました。

現在動画を視聴しようとする HTTP ベースのストリーミングプロトコル (HTTP Live Streaming や MPEG-DASH) がよく使われるということはご存知の方も多いでしょう。

逆に視聴でなく自分で配信したい!と思ったときにどんなプロトコルで送ってるのかな?と不思議に思った人も少なくないはずです。いや実は少ないと思っていますが、少なくないことにおきます。不思議に思わないまでも、HTTP で視聴できるんだから配信も逆にサーバーに HTTP で送ってるだけじゃー?と思っている人も居るんじゃないでしょうか。

だがしかし!実は配信元からサーバーへの動画送信には HTTP は使われておらず、Flash で使われていたプロトコルの RTMP がデファクトスタンダードになっています。Flash を使わずとも Adobe Media Server(旧 Flash Media Server) とやりとりするためのプロトコルなので、一応 RTMP は Flash とは独立しており仕様も公開されています。そんな事情もあり色々な配信用のソフトウェアやハードウェアでも実装され、逆に既存の配信用環境に合わせるようにサーバー側も Adobe Media Server でなくとも RTMP を受信できるよう対応し、というのが続いてデファクトスタンダードになってしまったようです。

というわけで、この本はそんな RTMP について解説する本です。

RTMP 自体はもう改定されそうもないですし、新しいコーデックへの対応もされないため、今後はまた別なプロトコルに置き換わっていくだろうと予想されます。しかしこれを書いている 2019 年初頭の時点ではまだこれといって有力な代替候補が定まっているとは言い難い状況ですので、そろそろ RTMP を解説できる最後の機会かなーと思ってこんな本を書きました。

元々 Flash のプロトコルなのですが、いまとなっては動画配信にしか使われないと思いますし、書いてる自分自身がそんなに Flash に詳しくないこともありまして、動画関係を中心に解説していきます。

2. RTMP とは

RTMP とは！ Real-Time Messaging Protocol の略で、Adobe Flash(旧 Macromedia Flash) と Adobe Media Server(旧 Flash Media Server) の通信プロトコルです。

Flash で使われる ActionScript から簡単に接続でき、動画や音声のやりとりができるため、Flash ベースのインターネットアプリケーション、ビデオ会議や動画配信を作るのに使われました。

メディアサーバーなんぞは他にも沢山ありましたが、RTMP の特徴としては ActionScript のオブジェクトをやりとりできるプロトコルという点があるでしょう。動画や音声だけでなく ActionScript のデータがいっしょに載せられることで、動画と同期してスクリプトでの動作を行ったりすることもできました。Flash Media Server はサーバー側アプリケーションも ActionScript で開発するので、クライアントとサーバーでシームレスにオブジェクトをやりとりできると非常に便利です。動画や音声を扱わない、Flash ベースのオンラインゲームにも使われました。

今となっては Flash と Adobe Media Server でアプリケーションを作ることはもうほとんど無いでしょうし、それで作るにしても RTMP という後継に近いプロトコルがあるので、RTMP が元の用途で使われることは少ないんじゃないかと思います。

しかしながら、動画をやりとりするのに Flash が便利だった時代はそれなりに長く、クライアントからサーバーに動画を送るのに RTMP を採用したソフトウェアやハードウェアがいくつも作られました。それらを使えるようにするため、また他にちょうど良いプロトコルがあったわけでもないという理由もあるためか、もう Flash も Adobe Media Server を使ってなくともクライアントからサーバーに動画(と音声)を送るプロトコルとして RTMP が採用され、デファクトスタンダードの地位を確立しました。

RTMP は仕様が公開されており、TCP で 1 本つなげばずっとやりとりできる比較的シンプルなプロトコルといったあたりも数多く実装された理由の一つでしょう。

2.1. RTMP で出来ること・出来ないこと

RTMP の機能は次のようなものがあります。

- TCP で接続
- シリアライズされた ActionScript オブジェクトの送受信
 - AMF0
 - AMF3
- 音声の送受信
 - ADPCM

1 Flash で使われるスクリプト言語。だいたい JavaScript みたいなもの。

2 UDP ベースで RTMP と名前は似てるけど中身はだいぶ別物

3 <https://www.adobe.com/devnet/rtmp.html> ただしいつまでであるかはわからない。

- MP3
- NellyMoser
- AAC
- Speex
- 動画の送受信
 - Sorenson (Flash Video)
 - Flash Video Screen V1・V2
 - On2 (VP6)
 - H.264
- 上記各種データ (オブジェクト・音声・動画) の 1 ストリームへの混載
 - オブジェクトデータも含めてタイムスタンプ付き
- 一つの接続に複数ストリームの重畳
- サーバー・クライアントの双方向通信

オブジェクトも含めてタイムスタンプが付いているので、オブジェクトと動画・音声を同期することができるのは嬉しいですね。

複数ストリームの重畳ができるので、1 接続で複数の動画やオブジェクトをそれぞれ非同期で流したりすることもできます。

もちろんいずれにせよアプリケーションが対応しないと意味ないですけど……。

逆に出来ないことは以下のようなものがあります。

- 暗号化
- P2P 通信
- 新しいコーデックへの対応

TLS などを使わない普通の TCP 通信なので暗号化はされていません。抜けば抜けます。SSL で暗号化する RTMPS や、もう少し軽い暗号化である RTMPE というプロトコルがあるようですが、詳細なドキュメントがないためわかりません。まあ少なくとも現在のクライアントからサーバーへの動画配信では使われてはいないようです。

TCP であるため P2P 通信はできません。いや TCP だからというわけではないですが NAT 越えなどを考えると TCP は不利です。UDP ベースの RTMFP だと P2P 通信もできるようになっています。

音声や動画のコーデックは独自の値が割り当てられていますので、仕様の改定がない限りは増えませんし、増やすこともできません。最新のコーデックが AAC・Speex・H.264 なので、もっと新しい H.265 や AV1、Opus なんかも使いたくても対応できません。この点が RTMP は実用的限界でもありますので、万が一バージョンアップでもされない限りは今後は使われなくなることでしょう。

2.2. 実装について

前にも述べた通り、仕様は公開されていますし、それほど複雑なものでもないため仕様をよく読めば誰でも実装することができます。

……と言いたいところだが。

仕様書が公開されているのはいいんですが、あちこち曖昧だったり省略されていたり古かったりして、他のデータの仕様書やドキュメントと突き合わせないとわからないところがあるところがあります。公式のドキュメントを見て実装しようとするとは何度も首をかしげることになるでしょう！さすが旧 Macromedia だぜ！！

この本ではなんとかそれなりに実装できる程度にはまとめたつもりなので、参考になれば幸いです。

3. RTMP 概観

詳細に入るまえにざっくりと RTMP 全体を見渡してみましょう。

RTMP はいくつかのプロトコル階層によって成り立っています。

1. TCP
2. RTMP Chunk Stream Protocol
3. RTMP Message

建前的にはそれぞれの階層は独立したプロトコルになっています。実際には密接に関連している部分があるのですが、ある程度独立になってる建前で仕様書が書かれているので逆に読みづらくなっているのが困ったところです。

3.1. TCP

まずベースは TCP です。普通の TCP なので特に説明するところもないです。

TLS/SSL は使わず、素の TCP を使います。TLS/SSL を使う RTMPS というのもあるらしいですが、詳細はよくわからない (仕様書にはない) ので割愛します。まあたぶん普通に TLS/SSL の上に乗せただけのものですけど。

ポート番号は 1935 です。もちろんサーバーとクライアント双方が指定すれば他のポートでも使えますが、どのアプリも特にポート番号を指定しなければ標準の 1935 を使うでしょう。

3.2. RTMP Chunk Stream Protocol

ここから RTMP 独自のプロトコルになります。

TCP の上に RTMP Chunk Stream Protocol というバイナリフォーマットで接続を確立します。TCP 接続が確立されると、次にこのレイヤーでハンドシェイクが行われ、さらに上位の通信を行います。

Chunk Stream Protocol という名の通り、このプロトコルではチャンク (= バイナリの塊) 単位でやりとりを行います。

一つのチャンクには次のような情報が含まれます。

- チャンクフォーマット
- チャンクストリーム ID
- タイムスタンプ
- メッセージ長
- メッセージタイプ ID
- メッセージストリーム ID

- ペイロード (メッセージボディ)

Chunk Stream Protocol にはチャンクストリーム ID というのが含まれますが、この ID が異なるチャンクは仮想的に異なる接続として扱われます。つまり一つの TCP 接続上に ID が異なるチャンクストリームを多重化することで、仮想的に複数の接続を確立することができます。

それと別にメッセージストリーム ID というのも持っており、これはより上位の RTMP Message のストリーム ID で、チャンクストリーム ID とは別物です。混同しないように注意しましょう。

3.2.1. Protocol Control Message

Chunk Stream Protocol でやりとりされるメッセージは基本的にはより上位の RTMP Message になるのですが、一部 Chunk Stream レベルで設定や通知するメッセージがあり、これらは Protocol Control Message と呼ばれます。

Protocol Control Message は接続全体のメッセージとなります。多重化はされずに常にチャンクストリーム ID が 2 番が使われます。

3.3. RTMP Message

RTMP Message はよりアプリケーションに近いメッセージです。

RTMP Chunk Stream Protocol まではアプリケーションに関係なく、RTMP プロトコルであれば必ず決まった動作をするものですが、RTMP Message は主にアプリケーションの責任で送受信するものです。RTMP Message が正しく送受信できるところまで作れば、あとはアプリケーションを作るだけと言えるでしょう。やったね！

RTMP Message には以下のような情報で構成されます。

- メッセージ長
- メッセージタイプ ID
- メッセージストリーム ID
- ペイロード (メッセージボディ)

おっとどこかで見たような……。Chunk Stream のチャンクに含まれる情報ですね。

RTMP ではこれらは Chunk Stream のチャンクに載ってきますので、チャンクのやりとりさえできていれば、RTMP Message で独自に規定されるのはメッセージタイプ毎の内容くらいです。

3.3.1. User Control Message

RTMP Message でもアプリケーションが基本的に処理するのは RTMP Command Message なのですが、他に接続制御みたいなものを行う User Control Message というのが存在します。Chunk Stream Protocol における Protocol Control Message が、RTMP Message レベルでは User Control Message として用意されています。

内容は Protocol Control Message ほど重要なものでもなく、ヒント程度の情報を通知するものです。

3.3.2. RTMP Command Messages

RTMP Command Messages はアプリケーションがなんか処理する必要があるメッセージです。

この中でもいくつかメッセージの種類があります。

- **Command Message**

サーバー (またはクライアント) への関数呼び出しをリクエストします。

- **Data Message**

なんらかのデータを送ります。が、通常は動画や音声のメタデータ (タイトル情報とか) を送信するのに使われます。

- **Video Message**

動画データを送ります。

- **Audio Message**

音声データを送ります。

- **Shared Object Message**

ActionScript の共有オブジェクトやそれらに対する変更について送ります。

- **Aggregate Message**

複数のメッセージをまとめるメッセージです。何に使うんだろう……。

アプリケーションで使うのは基本的に Command Message で、動画の再生開始だとか配信開始だとかをこれで通知しますし、受け取った側はその内容に応じた処理をします。あとは再生や配信が開始されると Data Message や Video・Audio Message がだらだらと流れます。

Shared Object Message や Aggregate Message は (少なくとも動画配信では) 使いません。

3.3.3. Action Message Format

Command Message では ActionScript の関数 (メソッド) 呼び出しの形でリクエストが送られますが、引数や戻り値も ActionScript のオブジェクトとして送受信されます。

そのため ActionScript のオブジェクトをエンコード (シリアライズ) する必要があるのですが、そのフォーマットが Action Message Format (AMF) です。AMF は 0 と 3 という 2 つのバージョンがありますが、そのいずれもが RTMP で使えるようになっています。ちょっと面倒ですがどちらも扱えるように実装しておかないといけません。

ちなみに AMF3 は ActionScript3 でのフォーマットで、AMF0 はそれ以前の ActionScript でのフォーマットのようなので、AMF0 の方が当然簡単です。

3.3.4. 動画・音声のフォーマット

Video Message や Audio Message には動画や音声は格納されますが、その内容は仕様書には書かれていません。

Flash で使う動画ファイルフォーマットの仕様書にある、FLV ファイルの仕様を見るとそれらしきものが見つかります。⁴ 基本的にはコーデック毎の生データが入っていますが、いくらか付加情報があるのでこれを見ないと再生はできないでしょう……。

4 <https://www.adobe.com/devnet/f4v.html> の FLV and F4V File Format Specification。例によっていつまであるかわからない。

4. RTMP Chunk Stream

ここからは、ついに RTMP の詳細に入っていきます。

なんとなく実装できそうなくらいには解説するつもりですが、コードとかは無いので本気で書く場合には頑張ってください。

4.1. バイナリフォーマットについて

詳しく見ていく前にちょっとバイナリフォーマットについて書いておきましょう。

まず 1 バイトは 8 ビットとします。1 オクテットと書く方が正確でしょうけど面倒なので 1 バイトと書きます。1 バイトが 8bit じゃない環境は今や見たことないくらい稀ですしね。

複数バイトにまたがる数値がある場合は、主にネットワークバイトオーダー (= ビッグエンディアン) が使われます。主に、ということだまーにリトルエンディアンなどところがありますが、そこはその都度注記します。特に記述が無い限りはビッグエンディアンです。

ビット単位で見ていくこともありますが、その場合は 0 ビット目を MSB(最上位ビット) とします。1 バイトのデータの 0 ビット目と言えば一番上のビットですし、7 ビット目と言えば一番下のビットになります。

実際に実装しない限りは流しちゃっていいですが、実装する時には気を付けて見てもらえばいいかと思います。

4.2. TCP での接続

TCP については普通なので特に解説しません。TCP でつないでください。ポート番号は 1935 です。

IPv6 でも IPv4 でもプロトコル上は関係ないので、どちらでも安心して使えます。

RTMPS のプロトコルを使う場合は TCP でつないだ上で SSL/TLS を使うと良いようです。ポート番号とかはよくわからないですね……。

4.3. ハンドシェイク

TCP でつながったら、まずは RTMP Chunk Stream のハンドシェイクを行います。ハンドシェイクとは直訳してしまうと握手ですが、つまりお互いにデータをやりとりしはじめる前の初期通信です。

ハンドシェイクのやり方は仕様書に書いてあるのですが、実はこれは古いハンドシェイクで

5 ネットワークバイトオーダーとかビッグエンディアンとかいまいちわからんという人は調べてみてください。そんなに難しくはないんですが長くなるのでここでは解説しません。

Flash のいつからか (10 くらい?) 新しいハンドシェイクが行われているようです。が、仕様書に書いてないので詳細はわかりません。

とりあえず古い方のハンドシェイクでも問題なくつながるので、まずは仕様書通りにやるのが良いでしょう。

4.3.1. 仕様書に書いてあるハンドシェイク

ハンドシェイクではクライアントとサーバーが互いに 3 つのパケットを送り合います。

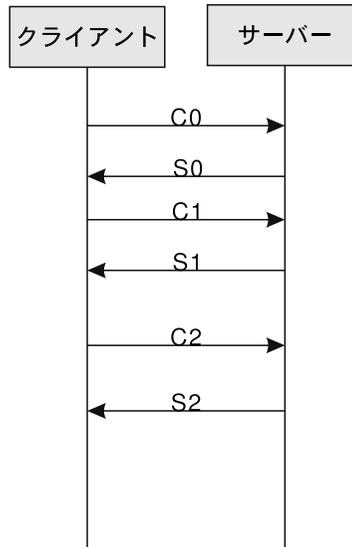


図 1. ハンドシェイクの手順

それぞれ C0、C1、C2 と S0、S1、S2 という名前が付いていて、C0～C2 はクライアントから送るパケット、S0～S2 はサーバーから送るパケットです。C0～C2 の C はクライアント (Client) の頭文字、S0～S2 の S はサーバー (Server) の頭文字ですね。

まずクライアントが C0 パケットを送信します。中身は 1 バイトのバージョン番号が入ります。



図 2. C0/S0 パケットの構造

バージョン番号は 3 固定です。以前は 0～2 もあったようですが仕様もないのでわかりませんが、どっちにしろ使われてないので気にしなくて良いでしょう。4 以降は現在無いようです。

サーバーは C0 パケットを受け取って中身が 3 だったら S0 パケットを返します。中身は同じく 1 バイトのバージョン番号で 3 を入れます。

サーバーが S0 パケットを送ったら、次はクライアントから C1 パケット、サーバーから S1 パケットを送り合います。どちらが先でも大丈夫そうに見えますが、とある事情により C1 が先になるよ