

季末的天堂

博客园 首页 新随笔 联系 订阅 管理

rtmp 协议详解

1. handshake

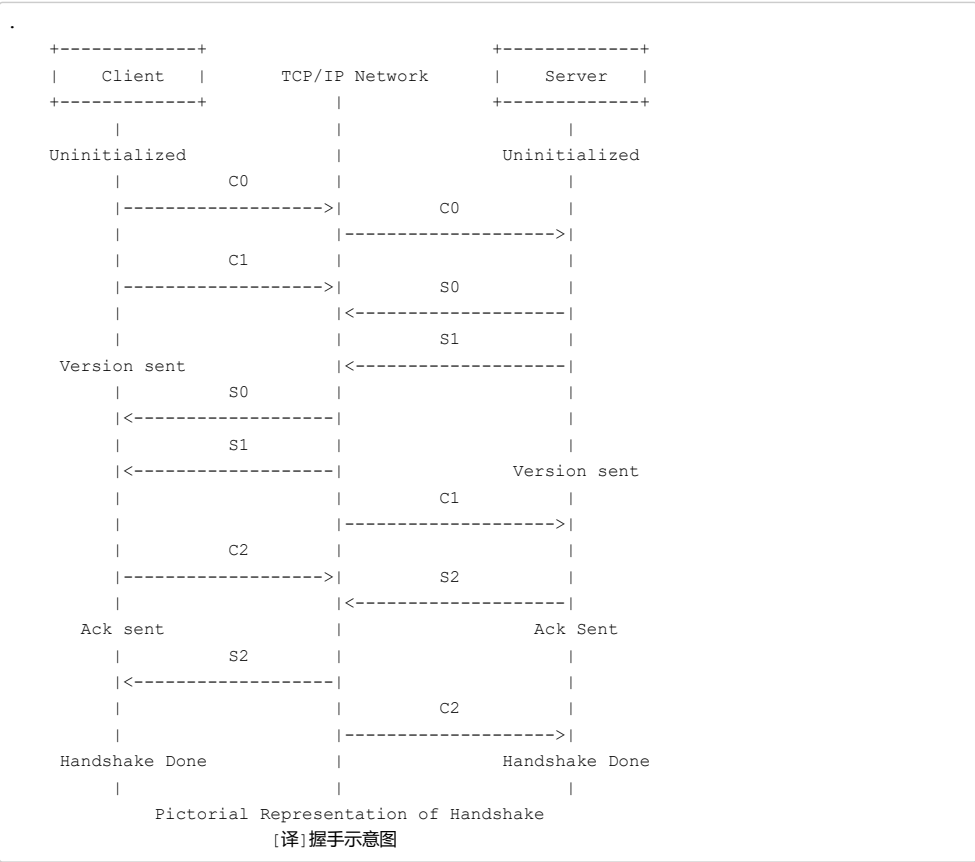
1.1 概述

rtmp 连接从握手开始。它包含三个固定大小的块。客户端发送的三个块命名为 C0,C1,C2；服务端发送的三个块命名为 S0,S1,S2。

握手序列：

- 客户端通过发送 C0 和 C1 消息来启动握手过程。客户端必须接收到 S1 消息，然后发送 C2 消息。客户端必须接收到 S2 消息，然后发送其他数据。
- 服务端必须接收到 C0 或者 C1 消息，然后发送 S0 和 S1 消息。服务端必须接收到 C2 消息，然后发送其他数据。

握手示意图



1.2 complex handshake

1.2.1 C0 和 S0 格式

C0 和 S0 包由一个字节组成，下面是 C0/S0 包内的字段：

公告

昵称：季末的天堂
园龄：2年8个月
粉丝：40
关注：0
[+加关注](#)

2021年1月						
日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索

谷歌搜索

常用链接

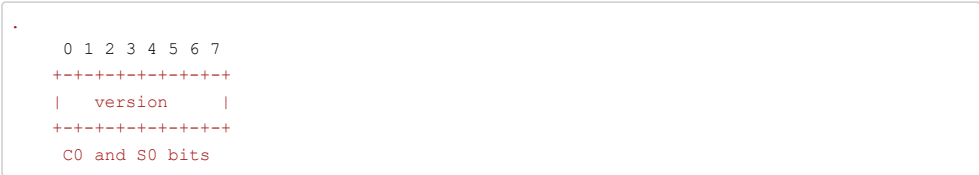
- [我的随笔](#)
- [我的评论](#)
- [我的参与](#)
- [最新评论](#)
- [我的标签](#)

我的标签

- [nginx\(36\)](#)
- [SRS\(25\)](#)
- [OpenResty\(14\)](#)
- [HTTP\(10\)](#)
- [nginx-rtmp\(7\)](#)
- [nginx事件管理\(6\)](#)
- [HLS\(5\)](#)
- [c++\(5\)](#)
- [TCP\(5\)](#)
- [State Threads\(4\)](#)
- [更多](#)

随笔分类

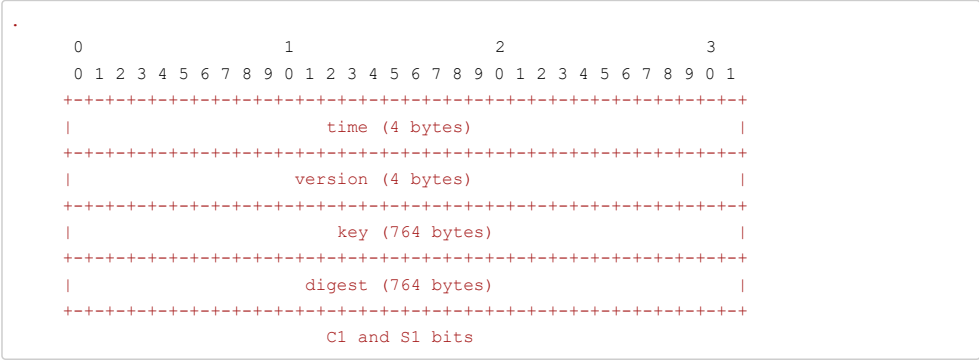
- [Apache Traffic Server\(1\)](#)
- [C++\(5\)](#)
- [CDN技术\(2\)](#)
- [FFMPEG\(3\)](#)
- [GDB\(2\)](#)
- [Golang\(6\)](#)



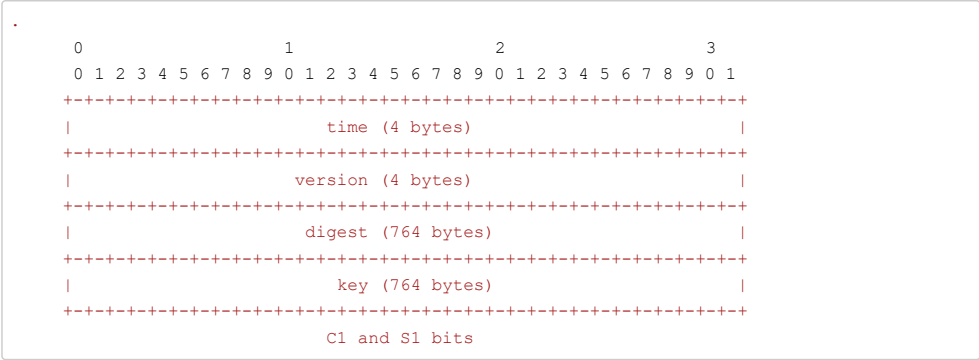
- version (1 byte) : RTMP 的版本, 一般为 3。

1.2.2 C1 和 S1 格式

C1和S1包含两部分数据: key和digest, 分别为如下:



key 和 digest 的顺序是不确定的, 也有可能是:(nginx-rtmp中是如下的顺序):



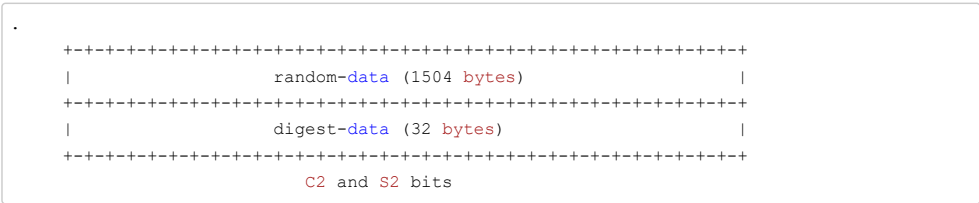
764 bytes key 结构:

- random-data: (offset) bytes
- key-data: 128 bytes
- random-data: (764 - offset - 128 - 4) bytes
- offset: 4 bytes

764 bytes digest 结构:

- offset: 4 bytes
- random-data: (offset) bytes
- digest-data: 32 bytes
- random-data: (764 - 4 - offset - 32) bytes

1.2.3 C2 和 S2 格式



handshake: S0 + S1 + S2

- H264(1)
- HLS(5)
- HTTP(10)
- Linux系统编程(4)
- Linux系统知识集锦(1)
- Lua or LuaJIT(1)
- Nginx(30)
- nginx-rtmp(6)
- OpenResty(14)
- 更多

随笔档案

- 2020年8月(1)
- 2018年10月(1)
- 2018年9月(2)
- 2018年8月(10)
- 2018年7月(12)
- 2018年6月(49)
- 2018年5月(34)
- 2018年4月(13)

最新评论

- 1. Re:C++之Lambda表达式
總結的很好
--是佚名
- 2. Re:Nginx事件管理之定时器事件
您好, 我也有个疑惑, nginx的定时器问题, e poll_wait在阻塞的时候, 如果这个时间又新添加一个定时时间小的定时器, 他是如何处理的呢, 求指导~
--conquerw
- 3. Re:Linux编程之共享内存
@ oyld 《Linux/Unix 系统编程手册 (下)》第 54 章 POSIX 共享内存 概述一节...
--季末的天堂
- 4. Re:Linux编程之共享内存
使用一个共享文件映射来进行 IPC 要求创建一个磁盘文件, 即使无需对共享区域进行持久存储也需要这样做。除了因需要创建文件所带来的不便之处, 这种技术还会带来一些文件 I/O 开销。 你好, 请问这一点有出...
--oyld
- 5. Re:C++之Lambda表达式
@ hejunyan可以...
--季末的天堂

阅读排行榜

- 1. C++之Lambda表达式(93749)
- 2. HLS协议解析(25504)
- 3. Linux编程之recvmsg和sendmsg函数(14267)
- 4. SRS之安装与使用(10085)
- 5. OpenResty之 lua_shared_dict 指令(9570)

推荐排行榜

- 1. C++之Lambda表达式(21)
- 2. rtmp 协议详解(4)
- 3. Nginx-rtmp直播之业务流程分析(2)
- 4. HLS协议解析(1)
- 5. SRS之安装与使用(1)

```
Real Time Messaging Protocol (Handshake S0+S1+S2)
  Handshake S0+S1+S2
    Protocol version: 03
    Handshake data: 293cdeec0d0e0a0d4cc46f697c58f4f3dcf1ea047b641b2b...
    Handshake data: 672dd09d5f9a566f9c057ec0a50d244c94619c47098e0016...

0000 03 29 3c de ec 0d 0e 0a 0d 4c c4 6f 69 7c 58 f4 .)<.... .L.oi|X.
0010 f3 dc f1 ea 04 7b 64 1b 2b 7e aa 72 de 51 a6 b6 .....{d. +~.r.Q..
0020 44 69 f7 a5 fa 6d 11 07 b9 d6 76 22 52 ce 16 45 Di...m.. ..v"R..E
0030 ab 08 2f af 83 93 cb ae 12 75 21 f0 c7 c7 a6 0b ../. .... .u!.....
0040 31 9d b1 2b 0a c2 32 c4 98 a8 e6 eb 77 fd 30 22 1..+..2. ....w.0"
0050 05 60 d1 88 f3 9c 37 05 12 58 f5 d9 1f 9b e4 50 .^ ....7. .X.....P
0060 39 95 7b 43 58 ad 07 f0 56 ee db cd eb 0c ef f0 9.{CX... V.....
0070 6c c0 78 5f 5d af 65 6f 07 5a 48 27 f6 2c 77 2f l.x_].eo .ZH'.w/
0080 c2 f3 72 1a a0 7a 0a f6 68 e6 c3 53 f2 b2 43 5e ..r..z... h..S..C^
0090 73 bb bd d0 6b 22 3f 72 7d 87 99 73 b3 11 a2 75 s...k"?r }.s....u
00a0 04 14 8f a4 8e 9a 9b f6 80 5e 49 72 11 8c d0 84 ..... .^Ir....
00b0 48 8d 54 b3 b0 93 25 2d 1a bf a0 cd d0 42 43 d4 H.T...%- ....BC.
00c0 56 d2 78 e5 6c 13 db ec 72 25 5e 83 b1 2e 07 f9 V.x.l... r%^.....
00d0 bc 5b ac 6c ee d2 99 08 91 39 d5 61 7b 18 35 d1 .[.l.... .9.a{.S.
00e0 eb ad b6 57 c1 92 44 33 b7 a2 b6 68 d1 bd 62 8d ...W..D3 ...h..b.
00f0 18 0e f9 06 e0 92 0e 71 cb e3 d2 46 fc 07 17 e7 .....q ...F....
0100 b5 ce 3e 76 60 82 a9 17 25 5f 7f f6 1c e1 83 34 ..>v`... %_.....4
0110 f0 7c 3a d0 0e 48 42 d9 2b 14 1f 27 1c 36 0e d1 .|:...HB. +..'.6..
0120 04 4d 47 64 cf f0 7b f4 4f fb ea 6b dc 6d 9f cc .MGd..{. O..k.m..
0130 e9 d9 9d f7 21 df d0 4c f3 ef 74 0f 26 82 e0 2a ....!...L .t.&..*
0140 cf 27 8f 9f 17 0a 93 66 05 7e d1 e2 eb 70 ae d5 .'.....f .~...p..
0150 49 4b cc 6a 2a 9d b7 1e 8c 2b 2d b2 ad 0e dd 7d IK.j*... .+~....}
0160 35 6c 1c 4d 76 af b3 7c 2d 85 5e 19 f5 0c ee 3f 5l.Mv..| -.^....?
0170 58 ba a9 82 57 60 a0 e4 8b ce 96 39 dc 73 b6 11 X...W`.. ...9.s..
```

1.3 simple handshake

1.3.1 C0 和 S0 格式

C0 和 S0 包由一个字节组成，下面是 C0/S0 包内的字段：

```
.
  0 1 2 3 4 5 6 7
+-+--+--+--+--+--+
|   version   |
+-+--+--+--+--+--+
C0 and S0 bits
```

- version (1 byte)：版本。在 C0 包内，这个字段代表客户端请求的 RTMP 版本号。在 S0 包内，这个字段代表服务端选择的 RTMP 版本号。当前使用的版本是 3。版本 0-2 用在早期的产品中，如今已经弃用；版本 4-31 被预留用于后续产品；版本 32-255（为了区分 RTMP 协议和文本协议，文本协议通常是可以打印字符）不允许使用。如果服务器无法识别客户端的版本号，应该回复版本 3。客户端可以选择降低到版本 3，或者终止握手过程。

1.3.2 C1 和 S1 格式

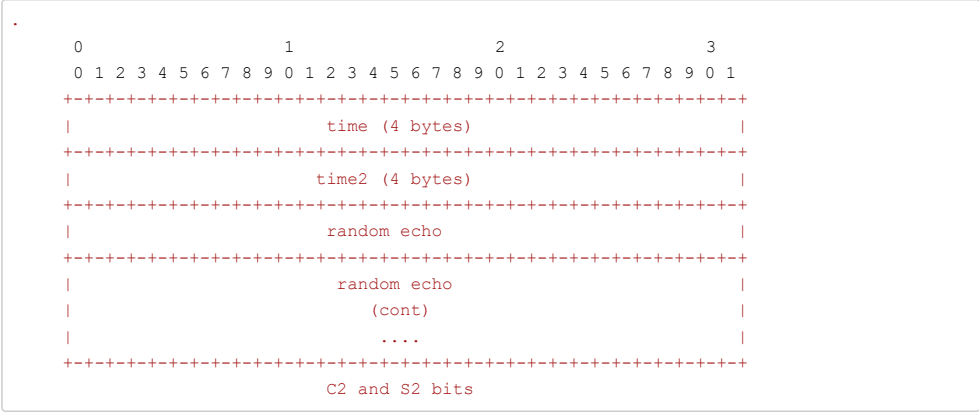
C1 和 S1 包长度为 1536 字节，包含以下字段：

```
.
  0                               1                               2                               3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                     time (4 bytes)                                     |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                     zero (4 bytes)                                     |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                     random bytes                                     |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                     random bytes                                     |
|                                     (cont)                                         |
|                                     ....                                           |
+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
C1 and S1 bits
```

- time (4 bytes)：本字段包含一个时间戳，客户端应该使用此字段来标识所有流块的时刻。时间戳取值可以为零或其他任意值。为了同步多个块流，客户端可能希望多个块流使用相同的时间戳。
- zero (4 bytes)：本字段必须为零。
- random (1528 bytes)：本字段可以包含任意数据。由于握手的双方需要区分另一端，此字段填充的数据必须足够随机（以防止与其他握手端混淆）。不过没有必要为此使用加密数据或动态数据。

1.3.3 C2 和 S2 格式

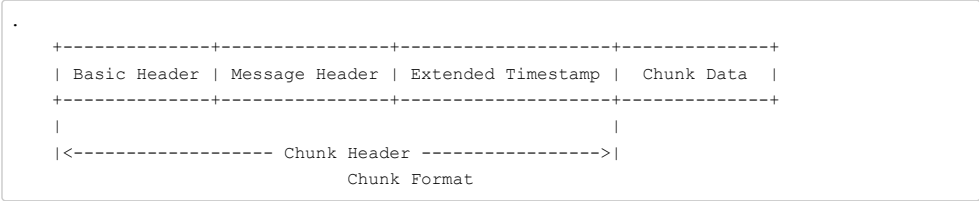
C2 和 S2 包长度为 1536 字节，作为 C1 和 S1 的回应，包含以下字段：



- time (4 bytes)：本字段必须包含对端发送的时间戳。
- time2 (4 bytes)：本字段必须包含时间戳，取值为接收对端发送过来的握手包的时刻。
- random (1528 bytes)：本字段必须包含对端发送过来的随机数据。握手的双方可以使用时间 1 和时间 2 字段来估算网络连接的带宽和/或延迟，但是不一定有用。

2. 组块

2.1 块格式



- 块的基本头（1-3字节）：这个字段包含块流ID和块类型。块类型决定了编码过的消息头的格式。这个字段是一个变长字段，长度取决于块流ID。
- 消息头（0,3,7,11字节）：这个字段包含被发送的消息信息（无论是全部，还是部分）。字段长度由块头中的块类型来决定。
- 扩展时间戳（0,4字节）：这个字段是否存在取决于块消息头中编码的时间戳。
- 块数据（可变大小）：当前块的有效数据，上限为配置的最大块大小。

2.2 Basic Header

包含 chunk stream ID（流通道id）和chunk type（即fmt），chunk stream id 一般被简写为CSID，用来唯一标识一个特定的流通道，chunk type决定了后面Message Header的格式。Basic Header的长度可能是 1，2，或 3 个字节，其中 chunk type 的长度是固定的（占2位，单位是bit），Basic Header 的长度取决于 CSID 的大小，在足够存储这两个字段的前提下最好用尽量少的字节从而减少由于引入Header增加的数据量。

RTMP协议支持用户自定义 [3,65599] 之间的 CSID，0, 1, 2 由协议保留表示特殊信息。0 代表 Basic Header 总共要占用 2 个字节，CSID 在 [64,319] 之间; 1 代表占用 3 个字节，CSID 在 [64,65599] 之间; 2 代表该 chunk 是控制信息和一些命令信息。

2.2.1 Basic Header: 1 byte



2.2.2 Basic Header: 2 byte , csid == 0

```

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|fmt|    0      | cs id - 64  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

CSID占22bit, 此时协议将第一个字节的[2,8]bit置1, 余下的16个bit表示CSID - 64, 这样共有16个bit来存储CSID, 16bit可以表示[0,65535]共 65536 个数, 因此这种情况下 CSID 在 [64,65599], 其中65599=65535+64, 需要注意的是, Basic Header是采用小端存储的方式, 越往后的字节数量级越高, 因此通过3个字节的每一个bit的值来计算CSID时, 应该是: <第三个字节的值> * 256 + <第二个字节的值> + 64.

```

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|fmt|      1      |          cs id - 64          |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

包含了要发送的实际信息（可能是完整的，也可能是一部分）的描述信息。Message Header的格式和长度取决于Basic Header的chunk type，即fmt，共有四种不同的格式。其中第一种格式可以表示其他三种表示的所有数据，但由于其他三种格式是基于对之前chunk的差量化的表示，因此可以更简洁地表示相同的数据，实际使用的时候还是应该采用尽量少的字节表示相同意义的数据。下面按字节从多到少的顺序分别介绍这四种格式的 Message Header。

一、Chunk Type(fmt) = 0: 11 bytes

```

0                                1                                2                                3
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                timestamp                                |message length|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  message length (continue)  |message type id|msg stream id|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                msg stream id                                |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

- timestamp (时间戳)：占用3个字节，因此它最多能表示到16777215=0xFFFFF=2²⁴-1，当它的值超过这个最大值时，这三个字节都置为1，这样实际的timestamp会转存到 Extended Timestamp 字段中，接收端在判断timestamp字段24个位都为1时就会去Extended Timestamp 中解析实际的时间戳。
- message length (消息数据长度)：占用3个字节，表示实际发送的消息的数据如音频帧、视频帧等数据的长度，单位是字节。注意这里是Message的长度，也就是chunk属于的Message的总长度，而不是chunk本身data的长度。
- message type id(消息的类型id)：1个字节，表示实际发送的数据的类型，如8代表音频数据，9代表视频数据。
- message stream id(消息的流id)：4个字节，表示该chunk所在的流的ID，和Basic Header 的CSID一样，它采用小端存储方式。

```

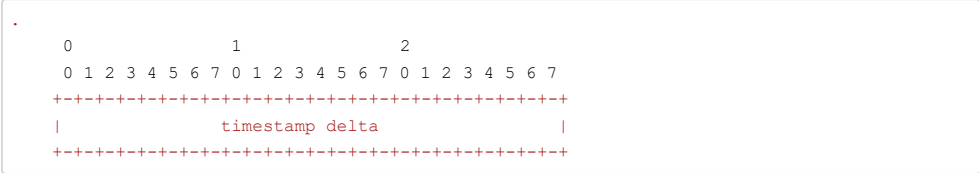
0               1               2               3
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|
|               timestamp delta               |message length |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|
|   message length (continue) |message type id|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

type为1时占用7个字节，省去了表示message stream id的4个字节，表示此chunk和上一次发的 chunk 所在的流相同，如果在发送端和对端有一个流链接的时候可以尽量采取这种格式。

- timestamp delta: 3 bytes，这里和type=0时不同，存储的是和上一个chunk的时间差。类似上面提到的timestamp，当它的值超过3个字节所能表示的最大值时，三个字节都置为1，实际的时间戳差值就会转存到Extended Timestamp字段中，接收端在判断timestamp delta字段24个bit都为1时就会去Extended Timestamp 中解析实际的与上次时间戳的差值。
- 其他字段与上面的解释相同。

三、Chunk Type(fmt) = 2: 3 bytes



type为2时占用3个字节，相对于type=1格式又省去了表示消息长度的3个字节和表示消息类型的1个字节，表示此 chunk 和上一次发送的 chunk 所在的流、消息的长度和消息的类型都相同。余下的这三个字节表示 timestamp delta，使用同type=1。

四、Chunk Type(fmt) = 3: 0 byte

type=3时，为0字节，表示这个chunk的Message Header和上一个是完全相同的。当它跟在type=0的chunk后面时，表示和前一个 chunk 的时间戳都是相同。什么时候连时间戳都是相同呢？就是一个 Message 拆分成多个 chunk，这个 chunk 和上一个 chunk 同属于一个 Message。而当它跟在 type = 1或 type = 2 的chunk后面时的chunk后面时，表示和前一个 chunk 的时间戳的差是相同的。比如第一个 chunk 的 type = 0，timestamp = 100，第二个 chunk 的 type = 2，timestamp delta = 20，表示时间戳为 100 + 20 = 120，第三个 chunk 的 type = 3，表示 timestamp delta = 20，时间戳为 120 + 20 = 140。

2.4 Extended Timestamp(扩展时间戳)

在 chunk 中会有时间戳 timestamp 和时间戳差 timestamp delta，并且它们不会同时存在，只有这两者之一大于3字节能表示的最大数值 0xFFFFF = 16777215 时，才会用这个字段来表示真正的时间戳，否则这个字段为 0。扩展时间戳占 4 个字节，能表示的最大数值就是 0xFFFFFFFF = 4294967295。当扩展时间戳启用时，timestamp 字段或者timestamp delta要全置为1，而不是减去时间戳或者时间戳差的值。

2.5 chunk 示例

2.5.1 chunk 示例1

本示例展示了一个音频消息流。流中包含有冗余信息。

+-----+-----+-----+-----+-----+							
	Message Stream ID		Message Type ID		Time	Length	
+-----+-----+-----+-----+-----+							
Msg # 1	12345		8	1000	32		
+-----+-----+-----+-----+-----+							
Msg # 2	12345		8	1020	32		
+-----+-----+-----+-----+-----+							
Msg # 3	12345		8	1040	32		
+-----+-----+-----+-----+-----+							
Msg # 4	12345		8	1060	32		
+-----+-----+-----+-----+-----+							
Sample audio messages to be made into chunks							

- 分析第一个 chunk：
 1. 首先包含第一个 Message 的 chunk 的 chunk type 为 0，因为它前面没有可参考的 chunk，timestamp 为 1000，表示时间戳。
 2. type 为 0 的 header 占用 11 个字节，假定 chunk stream id 为 3 < 127，因此 basic header 占用 1 个字节；
 3. 再加上 data 的 32 字节，因此第一个 chunk 共 44 字节 = 11 + 1 + 32 个字节。
- 分析第二个 chunk：
 1. 第二个 chunk 和第一个 chunk 的 cs id 和 chunk type id，以及 data 的长度都相同，因此采用 类型 2；
 2. 可知 timestamp delta = 1020 - 1000 = 20；
 3. 因此第二个 chunk 占用 36 = 3 (message header) + 1 (basic header) + 32

- 分析第三个 chunk:
 - 第三个 chunk 和第二个 chunk 的 cs id , chunk type id, 以及 data 的长度和时间戳的差值都相同, 因此采用 类型 3, 省去全部的 Message Header 的信息;
 - 因此占用 $33 = 1 + 32$
- 分析第四个 chunk:
 - 第四个 chunk 和第三个 chunk 情况相同, 也占用 $33 = 1 + 32$ 个字节。

最后实际发送的chunk如下面表格所示, 该表格展示了由此音频流产生的块信息。从第 3 条信息开始, 数据传输达到最大优化。每条消息的头部只增加了 1 字节长度。

Chunk#	Stream ID	Chunk Type	Header Data	No. of Bytes After Header	Total No. of Bytes in the Chunk
Chunk#1	3	0	delta: 1000 length: 32, type: 8, stream ID: 12345 (11 bytes)	32	44
Chunk#2	3	2	20 (3 bytes)	32	36
Chunk#3	3	3	none (0 bytes)	32	33
Chunk#4	3	3	none (0 bytes)	32	33

Format of each of the chunks of audio messages

2.5.2 chunk 示例2

本示例展示了一条长消息, 由于消息的长度超过了块的最大长度 (128字节) , 此消息在传输时将被分割成若干个块。

Msg #	Stream ID	Message Type ID	Time	Length
Msg # 1	12346	9 (video)	1000	307

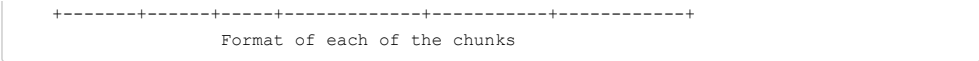
Sample Message to be broken to chunks

由表格知 data 的长度 $307 > 128$, 因此这个 Message 要分割成几个 chunk 发送:

- 第一个 chunk: type = 0, timestamp = 1000, 承担 128 个字节的 data, 因此共占用 $140 = 11 + 1 + 128$ 个字节。
- 第二个 chunk: 同样要发送 128 字节, 其他字段 (即Message Header 中的几个字段) 都与第一个相同, 因此采用 类型 3, 共 $129 = 1 + 128$ 字节。
- 第三个 chunk: 要发送的 data 的长度为 $307 - 128 - 128 = 51$ 字节, 还是采用 类型 3, 共 $1 + 51 = 52$ 字节。

下面是消息分割后产生的块:

Chunk#	Stream ID	Chunk Type	Header Data	No. of Bytes after Header	Total No. of bytes in the chunk
Chunk#1	4	0	delta: 1000 length: 307 type: 9, stream ID: 12346 (11 bytes)	128	140
Chunk#2	4	3	none (0 bytes)	128	129
Chunk#3	4	3	none (0 bytes)	51	52



第一个块的头数据显示了消息的长度为 307 字节。

在这两个示例中，类型为 3 的块有两种使用方式。第一种是说明消息的继续。第二种是说明新消息的头信息可以由前面已经存在的消息推到出来。

3. 协议控制消息

RTMP 块流使用消息类型 ID 1、2、3、5、6 作为控制消息。这些消息包含了必要的 RTMP 块流协议信息。

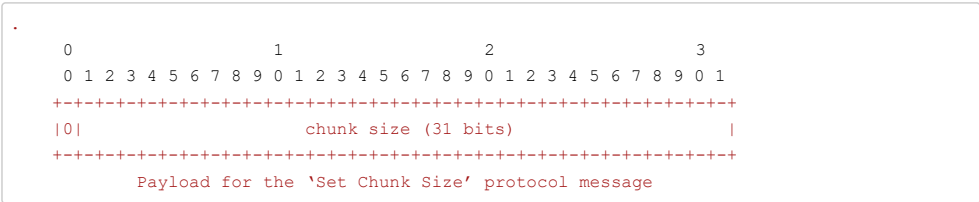
这些协议控制消息必须使用 0 作为消息流ID（作为已知的控制流ID），同时使用 2 作为块流ID。协议控制消息接收立即生效；解析时，时间戳字段被忽略。

3.1 设置块大小 (1)

协议控制消息（1），设置块大小，被用来通知对方新的最大的块大小。

默认最大的块大小为 128 字节，客户端和服务端可以使用此消息来修改默认的块大小。例如，假设客户端想要发送的音频数据大小为131 字节，而块大小为 128 字节。在这种情况下，客户端可以通知服务器新的块大小为 131 字节，然后就可以使用一个块来发送完整的音频数据了。

最大的块大小至少为 128 字节，块至少携带 1 个字节的内容。通信的每一个方向（例如从客户端到服务器）拥有独立的块大小设置。

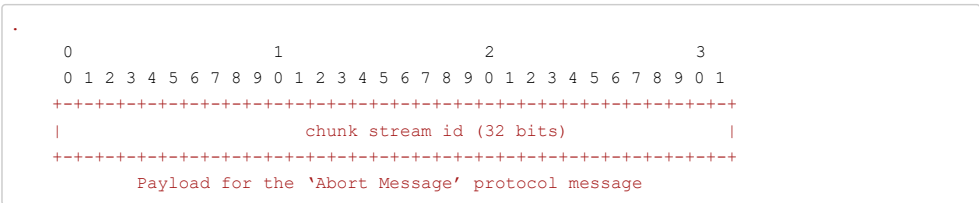


- 0: 当前比特位必须为零。
- chunk size(31 bits): This field holds the new maximum chunk size, in bytes, which will be used for all of the sender's subsequent chunks until further notice. Valid sizes are 1 to 2147483647(0x7FFFFFFF) inclusive; however, all sizes greater than 16777215(0xFFFFF) are equivalent since no chunk is larger than onemessage, and no message is larger than 16777215 bytes.
- 块大小（31比特）：本字段标识了新的最大块大小，以字节为单位，发送端之后将使用此值作为最大的块大小。本字段的有效值为 1 - 2147483647(0x7FFFFFFF)，由于消息的最大长度为 16777215(0xFFFFF)，而一个块最多只能携带一条消息，因此本字段的实际有效值为 1~16777215(0xFFFFF)。

send chunk size

3.2 中断消息 (2)

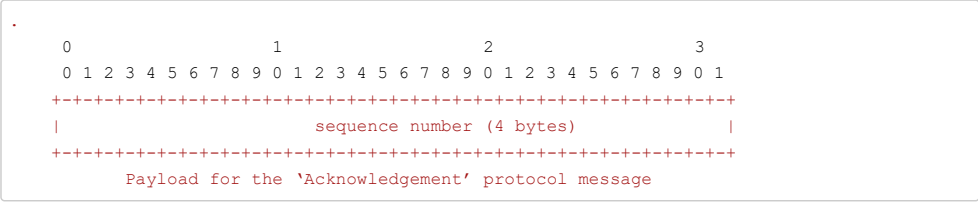
协议控制消息（2），中断消息，用来通知通信的对方，如果正在等待一条消息的部分块（已经接收了一部分），那么可以丢弃之前已经接收到的块。通信的一方将接收到块流ID作为当前协议消息的有效数据。应用程序可以发送此消息来通知对方，当前正在传输的消息没有必要再处理了。



- 块流ID（32比特）：本字段包含了块流ID，用来标识哪个块流ID的消息将被丢弃。

3.3 应答 (3)

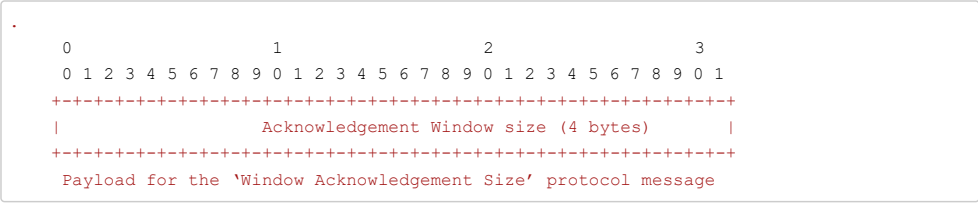
客户端和服务端在接收到与接收窗口大小相等的数据后，必须发送应答消息给对方。窗口大小的定义为发送方在接收到接收方的任何应答前，可以发送的最大数据量。本消息包含了序列号，序列号为截至目前接收到的数据总和，以字节为单位。



- 序列号（32比特）：本字段包含了截止目前接收到的数据总和，以字节为单位。

3.4 应答窗口大小（5）

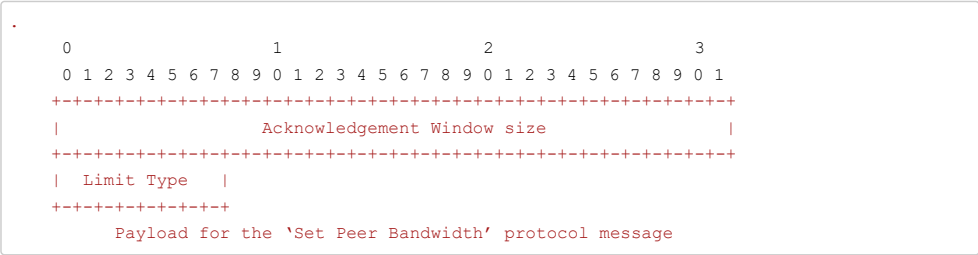
客户端和服务端发送这个消息来通知对方应答窗口的大小。发送方在发送了等于窗口大小的数据之后，等待接收对方的应答消息（在接收到应答之前停止发送数据）。接收方必须发送应答消息，在会话开始时，或从上一次发送应答之后接收到了等于窗口大小的数据。



send ack_size

3.5 设置流带宽（6）

客户端和服务端发送此消息来说明对方的出口带宽限制。接收方以此来限制自己的出口带宽，即限制未被应答的消息数据大小。接收到此消息的一方，如果窗口大小与上次发送的不一致，应该回复应答窗口大小的消息。



限制类型的取值为下面之一：

- 硬限制（0）：应该限制出口带宽为指明的窗口大小。
- 软限制（1）：应该限制出口带宽为指明的窗口大小，或已经生效的小一点的窗口大小。
- 动态限制（2）：如果上一次为硬限制，此消息被视为硬限制，否则忽略此消息。

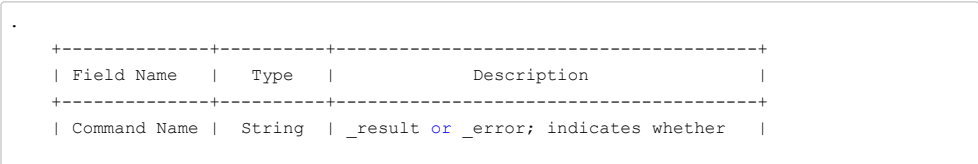
send bandwidth

4. Command Message（17 或 20）

Command Message（命令消息，Message Type ID = 17 或 20）：表示在客户端和服务端间传递的在对端执行某些操作的命令消息，connect 表示连接对端，对端如果同意连接的话就会记录发送端信息并返回连接成功消息，publish 表示开始向对方推流，接收端接收到命令后准备好接收对端发送的流信息。当信息使用 AMF0 编码时，Message Type ID = 20，AMF3 编码时为 17。

服务器和客户端之间使用 AMF 编码的命令消息交互。一些命令消息被用来发送操作指令，比如 connect，createStream，public，play，pause。另外一些命令消息被用来通知发送方请求命令的状态，比如 onstatus，result 等。一条命令消息包括命令对称、交互 ID、包含相关参数的命令对象。服务器和客户端通过在创建的流中远程调用的方式，使用命令消息来进行交互。

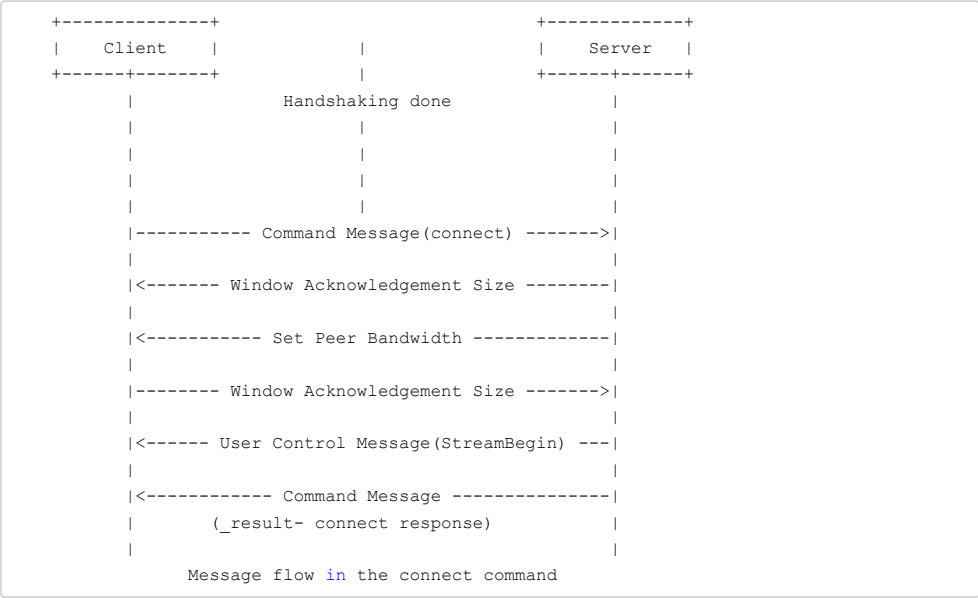
服务器发送给客户端的命令结构如下：



		the response is result or error.	
Transaction ID	Number	Transaction ID is 1 for connect responses	
Properties	Object	Name-value pairs that describe the properties(fmsver etc.) of the connection.	
Information	Object	Name-value pairs that describe the response from the server. 'code', 'level', 'description' are names of few among such information.	

命令执行过程中的消息流如下:

- 客户端发送连接命令给服务器, 获得与服务器连接的实例。
- 服务器在接收到连接命令后, 发送应答窗口大小的消息给客户端。同时与连接命令中接到的应用建立连接。
- 服务器发送设置流带宽消息给客户端。
- 客户端在接收并处理了设置流带宽的消息后, 发送应答窗口大小的消息给服务器。
- 服务器接着发送开始流的用户控制消息给客户端。
- 服务器发送 result 命令消息给客户端, 通知连接状态是成功或失败。命令消息中包含了事务ID。消息中还包含了像 FMS 版本之类的属性, 以及级别, 编码, 描述, 对象编码等信息。



4.1 命令类型

客户端和服务端通过 AMF 编码的数据交换命令。发送者发送包含命令名称, 事务ID, 包含相关参数的命令对象的消息。例如, 通过连接命令中包含的 APP 参数来告诉服务器连接的对方是哪个客户端。接收方处理命令消息, 并使用相同的事务ID应答。应答字符串为 _result 或 _error 或方法名, 例如 verifyClient 或 contactExternalServer。事务 ID 标明了应答指向的命令。事务ID相当于 IMAP 协议或其他协议中的标签。命令字符串中的方法名, 表明了发送端想要在接收端执行的方法。

下面的类对象被用来发送各种命令:

- NetConnection: 服务器和客户端之间进行网络连接的一种高级表示形式。
- NetStream: 代表了发送音频流, 视频流, 或其他相关数据的频道。当然还有一些像播放, 暂停之类的命令, 用来控制数据流。

4.2 网络连接命令

网络连接管理着客户端和服务端之间的双向连接。另外, 它也支持异步远程命令调用。

网络连接允许使用以下的命令:

- 连接 connect
- 调用 call

- 停止 close
- 创建流 createStream

4.2.1 connect: 连接

客户端发送连接命令给服务器，来获取一个和服务器通信的实例。客户端发送给服务器的命令结构如下：

Field Name	Type	Description
Command Name	String	Name of the command. Set to "connect".
Transaction ID	Number	Always set to 1.
Command Object	Object	Command information object which has the name-value pairs.
Optional User Arguments	Object	Any optional information

下面是连接命令的命令对象里包含的键值对的说明：

Property	Type	Description	Example Value
app	String	The Server application name the client is connected to.	testapp
flashver	String	Flash Player version. It is the same string as returned by the ApplicationScript getVersion () function.	FMSc/1.0
swfUrl	String	URL of the source SWF file making the connection.	file:///C:/FlvPlayer.swf
tcUrl	String	URL of the Server. It has the following format: protocol://servername:port/appName/appInstance	rtmp://localhost:1935/testapp/instance1
fpad	Boolean	True if proxy is being used.	true or false
audioCodecs	Number	Indicates what audio codecs the client supports.	SUPPORT_SND_MP3
videoCodecs	Number	Indicates what video codecs are supported.	SUPPORT_VID_SOIRENSEN
videoFunction	Number	Indicates what special video functions are supported.	SUPPORT_VID_CLIENT_SEEK
pageUrl	String	URL of the web page from where the SWF file was loaded.	http://somehost/sample.html
objectEncoding	Number	AMF encoding method.	AMF3

音频编码属性的可选值：

- 原始 PCM, ADPCM, MP3, NellyMoser (5, 8,11, 16, 22, 44kHz) , AAC, Speex.

Codec Flag	Usage	Value
SUPPORT_SND_NONE	Raw sound, no compression	0x0001
SUPPORT_SND_ADPCM	ADPCM compression	0x0002
SUPPORT_SND_MP3	mp3 compression	0x0004
SUPPORT_SND_INTEL	Not used	0x0008

SUPPORT_SND_UNUSED	Not used	0x0010
SUPPORT_SND_NELLY8	NellyMoser at 8-kHz compression	0x0020
SUPPORT_SND_NELLY	NellyMoser compression (5, 11, 22, and 44 kHz)	0x0040
SUPPORT_SND_G711A	G711A sound compression (Flash Media Server only)	0x0080
SUPPORT_SND_G711U	G711U sound compression (Flash Media Server only)	0x0100
SUPPORT_SND_NELLY16	NellyMouser at 16-kHz compression	0x0200
SUPPORT_SND_AAC	Advanced audio coding (AAC) codec	0x0400
SUPPORT_SND_SPEEX	Speex Audio	0x0800
SUPPORT_SND_ALL	All RTMP-supported audio codecs	0x0FFF

视频编码属性的可选值:

- Sorenson, V1, On2, V2, H264.

Codec Flag	Usage	Value
SUPPORT_VID_UNUSED	Obsolete value	0x0001
SUPPORT_VID_JPEG	Obsolete value	0x0002
SUPPORT_VID_SORENSEN	Sorenson Flash video	0x0004
SUPPORT_VID_HOMEBREW	V1 screen sharing	0x0008
SUPPORT_VID_VP6 (On2)	On2 video (Flash 8+)	0x0010
SUPPORT_VID_VP6ALPHA (On2 with alpha channel)	On2 video with alpha channel	0x0020
SUPPORT_VID_HOMEBREWV2 (screensharing v2)	Screen sharing version 2 (Flash 8+)	0x0040
SUPPORT_VID_H264	H264 video	0x0080
SUPPORT_VID_ALL	All RTMP-supported video codecs	0x00FF

视频函数属性的可选值:

Function Flag	Usage	Value
SUPPORT_VID_CLIENT_SEEK	Indicates that the client can perform frame-accurate seeks.	1

对象编码属性的可选值:

Encoding Type	Usage	Value
AMF0	AMF0 object encoding supported by Flash 6 and later	0

	AMF3	AMF3 encoding from		3	
		Flash 9 (AS3)			
+-----+-----+-----+					

示例

C -> S: 服务器接收客户端 connect 命令消息

S -> C: 服务器响应客户端 connect 成功消息

4.2.2 call: 调用

网络连接对象中包含的 call 方法，会在接收端执行远程过程调用（RPC）。被调用的 RPC 方法名作为 call 方法的参数传输。

从发送端到接收端的命令结构如下：

Field Name	Type	Description
Procedure Name	String	Name of the remote procedure that is called.
Transaction ID	Number	If a response is expected we give a transaction Id. Else we pass a value of 0
Command Object	Object	If there exists any command info this is set, else this is set to null type.
Optional Arguments	Object	Any optional arguments to be provided

应答的命令结构如下：

Field Name	Type	Description
Command Name	String	Name of the command.
Transaction ID	Number	ID of the command, to which the response belongs.
Command Object	Object	If there exists any command info this is set, else this is set to null type.
Response	Object	Response from the method that was called.

4.2.3 createStream: 创建流

客户端通过发送此消息给服务器来创建一个用于消息交互的逻辑通道。音频，视频，和元数据都是通过 createStream 命令创建的流通道发布出去的。

NetConnection 是默认的交互通道，流 ID 为0. 协议和一部分命令消息，包含 createStream，都是使用默认的交互通道发布的。

从客户端发送给服务器的命令结构如下：

Field Name	Type	Description
Command Name	String	Name of the command. Set to "createStream".
Transaction ID	Number	Transaction ID of the command.
Command	Object	If there exists any command info this

	Object		is set, else this is set to null type.	
+	-----	+	-----	+

从服务器发送给客户端的命令结构:

Field Name	Type	Description
Command Name	String	_result or _error; indicates whether the response is result or error.
Transaction ID	Number	ID of the command that response belongs to.
Command Object	Object	If there exists any command info this is set, else this is set to null type.
Stream ID	Number	The return value is either a stream ID or an error information object.

示例

C -> S: 服务器接收客户端 createStream 命令消息

C -> S: 服务器响应客户端 createStream 成功消息

4.3 网络流命令

网络流定义了通过网络连接把音频，视频和数据消息流在客户端和服务端之间进行交换的通道。一个网络连接对象可以有多个网络流，进而支持多个数据流。

客户端可以通过网络流发送到服务器的命令如下：

- 播放play
- 播放2 play2
- 删除流 deleteStream
- 关闭流 closeStream
- 接收音频 receiveAudio
- 接收视频 receiveVideo
- 发布 publish
- 定位 seek
- 暂停 pause

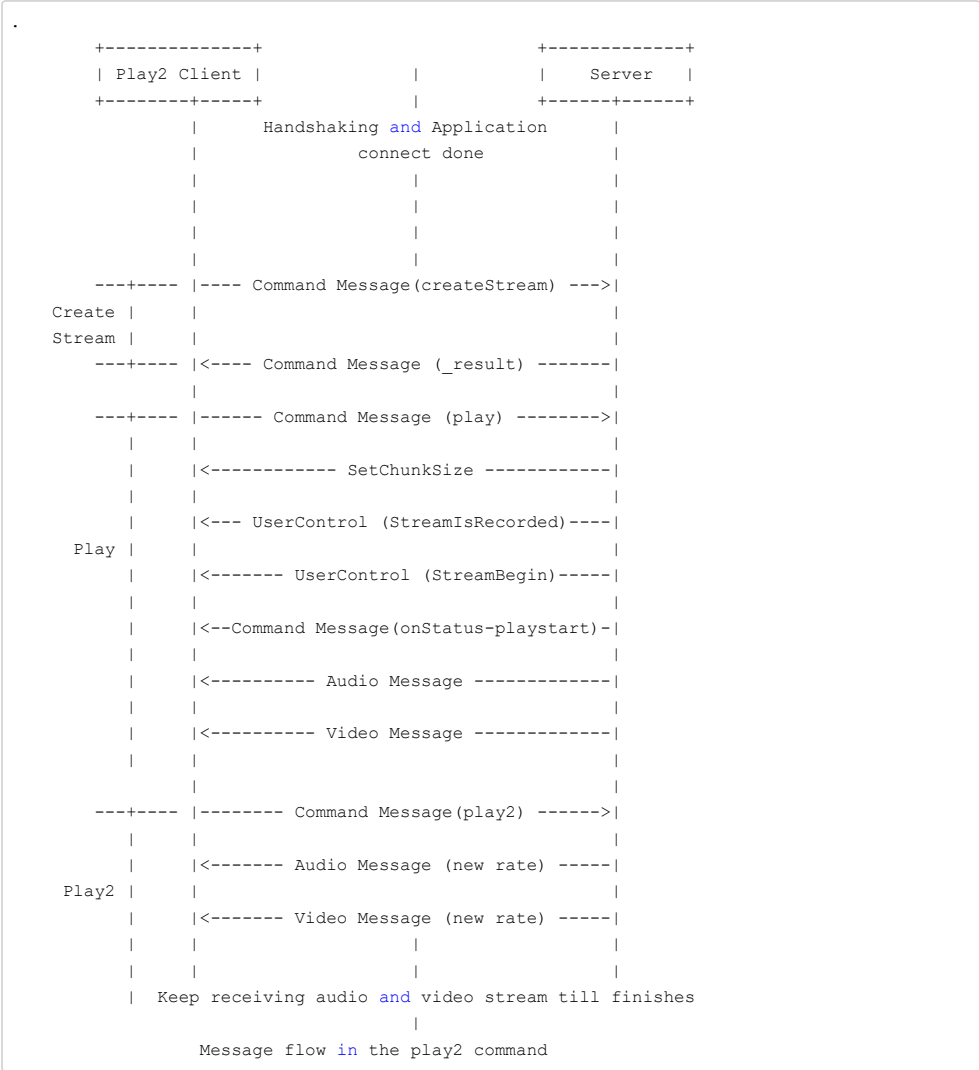
服务器通过发送 onStatus 命令给客户端来通知网络流状态的更新。

Field Name	Type	Description
Command Name	String	The command name "onStatus".
Transaction ID	Number	Transaction ID set to 0.
Command Object	Null	There is no command object for onStatus messages.
Info Object	Object	An AMF object having at least the following three properties: "level" (String): the level for this message, one of "warning", "status", or "error"; "code" (String): the message code, for example "NetStream.Play.Start"; and "description" (String): a human-readable description of the message. The Info object MAY contain other

ID		
Command	Null	Command information does not exist.
Object		Set to null type.
Parameters	Object	An AMF encoded object whose properties are the public properties described for the flash.net.NetStreamPlayOptions ActionScript object.

有关 NetStreamPlayOptions 对象的公开属性的说明详见 AS3 语言的文档。

此命令的消息流如下所示:



4.3.3 deleteStream: 删除流

如果需要销毁网络流对象，可以通过网络流发送删除流消息给服务器。

客户端发送给服务器的命令结构如下:

The command structure from the client to the server is as follows:

Field Name	Type	Description
Command Name	String	Name of the command, set to "deleteStream".
Transaction ID	Number	Transaction ID set to 0.
ID		
Command	Null	Command information object does not exist. Set to null type.
Object		
Stream ID	Number	The ID of the stream that is destroyed on the server.

服务器接收到此消息后，不做任何回复。

4.3.4 receiveAudio: 接收音频

网络流发送此消息通知服务器，是否要发送音频数据给客户端。

The command structure from the client to the server is as follows:

Field Name	Type	Description
Command Name	String	Name of the command, set to "receiveAudio".
Transaction ID	Number	Transaction ID set to 0.
Command Object	Null	Command information object does not exist. Set to null type.
Bool Flag	Boolean	true or false to indicate whether to receive audio or not.

如果服务器接收到带有 false 标签的消息后，不做任何回复。如果接收到带有 true 标签的消息，服务器回复带有 NetStream.Seek.Notify 和 NetStream.Play.Start 的消息给客户端。

4.3.5 receiveVideo: 接收视频

网络流发送此消息通知服务器，是否要发送视频数据给客户端。

客户端发送给服务器的命令结构如下：

Field Name	Type	Description
Command Name	String	Name of the command, set to "receiveVideo".
Transaction ID	Number	Transaction ID set to 0.
Command Object	Null	Command information object does not exist. Set to null type.
Bool Flag	Boolean	true or false to indicate whether to receive video or not.

如果服务器接收到带有 false 标签的消息后，不做任何回复。如果接收到带有 true 标签的消息，服务器回复带有 NetStream.Seek.Notify 和 NetStream.Play.Start 的消息给客户端。

4.3.6 publish: 发布

客户端发送此消息，用来发布一个有名字的流到服务器。其他客户端可以使用此流名来播放流，接收发布的音频，视频，以及其他数据消息。

客户端发送给服务器的命令结构如下：

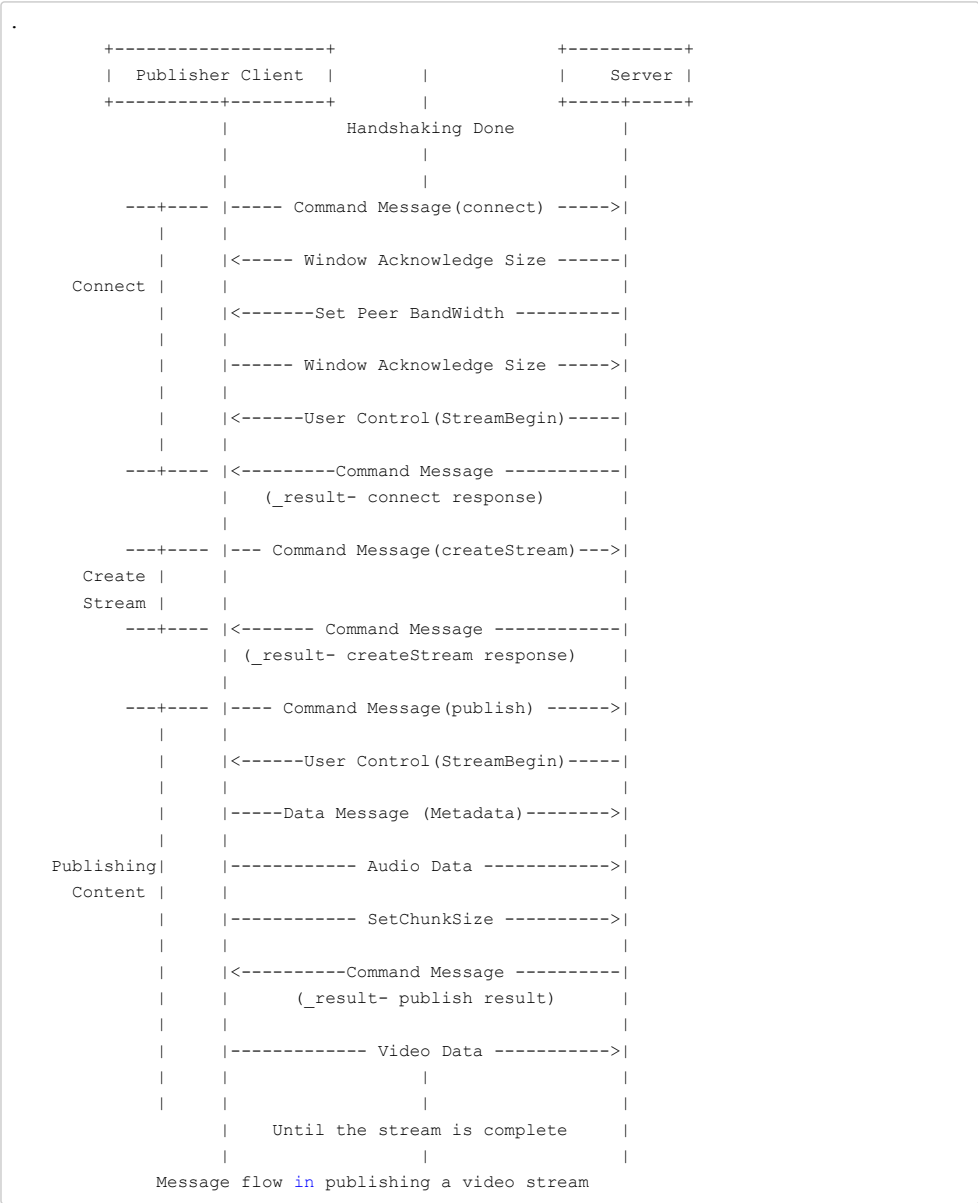
Field Name	Type	Description
Command Name	String	Name of the command, set to "publish".
Transaction ID	Number	Transaction ID set to 0.
Command Object	Null	Command information object does not exist. Set to null type.
Publishing Name	String	Name with which the stream is published.
Publishing Type	String	Type of publishing. Set to "live", "record", or "append".

```
|
|
| record: The stream is published and the
| data is recorded to a new file.The file
| is stored on the server in a
| subdirectory within the directory that
| contains the server application. If the
| file already exists, it is overwritten.
| append: The stream is published and the
| data is appended to a file. If no file
| is found, it is created.
| live: Live data is published without
| recording it in a file.
+-----+-----+-----+
```

服务器接收到此消息后，回复 onStatus 命令来标记发布的开始。

示例1：发布录制的视频

此示例阐述了发布者如果发布视频流到服务器。其他客户端可以订阅并播放此视频流。



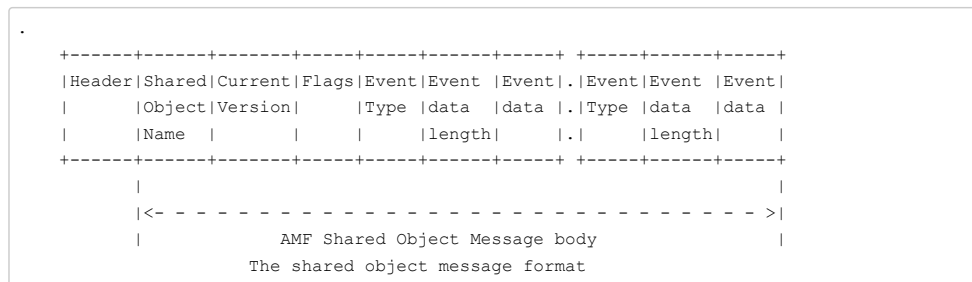
示例2：从录制的流发布元数据

本示例展示了发布元数据的消息交换过程。



Shared Object Message (共享消息, Message Type ID = 16 或 19) : 表示一个 Flash 类型的对象, 由键值对的集合组成, 用于多客户端, 多实例时使用。AMF0 时 Message Type ID 为 19, AMF3 为 16。每个消息可以包含多个事件。

共享消息格式：

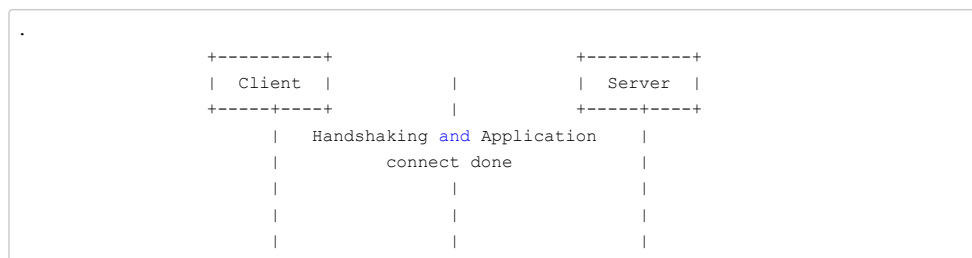


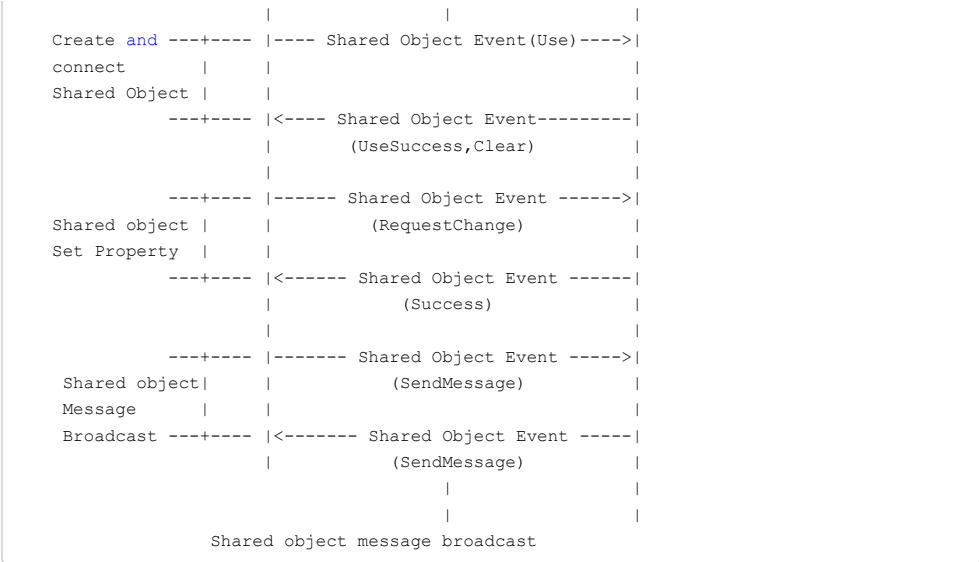
下面是共享消息支持的事件类型:

Event	Description
Use(=1)	The client sends this event to inform the server about the creation of a named shared object.
Release(=2)	The client sends this event to the server when the shared object is deleted on the client side.
Request Change(=3)	The client sends this event to request that the change the value associated with a named parameter of the shared object.
Change(=4)	The server sends this event to notify all clients, except the client originating the request, of a change in the value of a named parameter.
Success(=5)	The server sends this event to the requesting client in response to RequestChange event if the request is accepted.
SendMessage(=6)	The client sends this event to the server to broadcast a message. On receiving this event, the server broadcasts a message to all the clients, including the sender.
Status(=7)	The server sends this event to notify clients about error conditions.
Clear(=8)	The server sends this event to the client to clear a shared object. The server also sends this event in response to Use event that the client sends on connect.
Remove(=9)	The server sends this event to have the client delete a slot.
Request Remove(=10)	The client sends this event to have the client delete a slot.
Use Success(=11)	The server sends this event to the client on a successful connection.

示例：广播共享对象消息

此示例阐述了流创建过程中的消息交换，和共享对象的变换过程。同时还展示了共享对象消息广播的处理过程。





7. Audio Message (8)

Audio Message (音频信息, Message Type ID = 8) : 音频数据

8. Video Message (9)

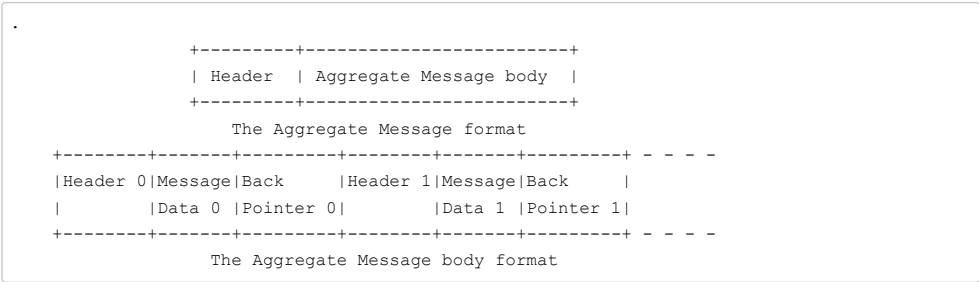
Video Message (视频信息, Message Type ID = 9) : 视频数据

9. Aggregate Message (22)

Aggregate Message (聚集信息, Message Type ID = 22) : 多个 RTMP 子消息的集合。

集合消息由消息头和消息内容组成。

消息内容由子消息组成, 子消息由消息头, 消息数据, 回放指针组成。



集合消息的消息流 ID 覆盖此消息内的子消息流的ID。

集合消息和第一个子消息的时间戳之间的偏移量, 用来将子消息的时间戳处理为流的时间刻度。每个子消息的时间戳可以通过添加偏移量来处理为正常的流时间。第一个子消息的时间戳应该和集合消息的时间戳相同, 因此偏移量应该为零。

方向指针包含了以前的消息 (包含头信息) 的大小。集合消息包含此字段, 一是为了适配 FLV 文件格式, 二是为了回放定位。

使用集合消息有如下几种优势:

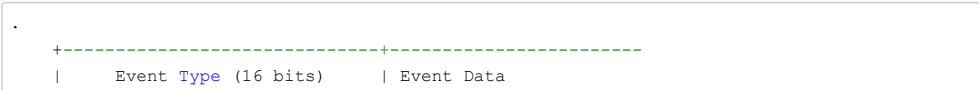
- 1. 块流在一个块内至多可以携带一条完整的消息。使用集合消息之后, 不仅可以增加块大小, 同时还减少了发送的块数量;
- 2. 集合消息的子消息可以连续的存储在内存中。当系统调用网络发送数据时更高效。

10. User Control Message Events (4)

User Control Message Events (用户控制消息, Message Type ID = 4) : 告知对方执行该信息中包含的用户控制事件, 比如 Stream Begin 事件告知对方流信息开始传输。和前面提到的协议控制信息 (Protocol Control Message) 不同, 用户控制消息是在 RTMP 协议层的, 而不是在 RTMP chunk 流协议层, 这个很容易弄混。该信息在 chunk 流中发送时, Message Stream ID = 0, Chunk Stream id = 2, Message type id = 4。

用户控制消息应该使用 0 作为消息流 ID, 当通过 RTMP 块流发送此消息时, 块流 ID 为 2。RTMP 流中的用户控制消息在接收时立即生效, 消息中的时间戳被忽略。

客户端或服务器发送此消息用来通知对方用户控制事件。此消息包含事件类型和事件数据。



```
+-----+
| Payload for the 'User Control' protocol message
```

用户控制消息的前 2 个字节数据用来标识事件类型。事件类型后面是事件数据。事件数据字段是可变的。由于此消息是通过 RTMP 块流层发送的，块大小的最大值应该满足在一个块里包含此消息。

用户控制事件支持如下类型：

Event	Description
Stream Begin (=0)	The server sends this event to notify the client that a stream has become functional and can be used for communication. By default, this event is sent on ID 0 after the application connect command is successfully received from the client. The event data is 4-byte and represents the stream ID of the stream that became functional.
Stream EOF (=1)	The server sends this event to notify the client that the playback of data is over as requested on this stream. No more data is sent without issuing additional commands. The client discards the messages received for the stream. The 4 bytes of event data represent the ID of the stream on which playback has ended.
StreamDry (=2)	The server sends this event to notify the client that there is no more data on the stream. If the server does not detect any message for a time period, it can notify the subscribed clients that the stream is dry. The 4 bytes of event data represent the stream ID of the dry stream.
SetBuffer Length (=3)	The client sends this event to inform the server of the buffer size (in milliseconds) that is used to buffer any data coming over a stream. This event is sent before the server starts processing the stream. The first 4 bytes of the event data represent the stream ID and the next 4 bytes represent the buffer length, in milliseconds.
StreamIs Recorded (=4)	The server sends this event to notify the client that the stream is a recorded stream. The 4 bytes event data represent the stream ID of the recorded stream.
PingRequest (=6)	The server sends this event to test whether the client is reachable. Event data is a 4-byte timestamp, representing the local server time when the server dispatched the command. The client responds with PingResponse on receiving MsgPingRequest.
PingResponse (=7)	The client sends this event to the server in response to the ping request. The event data is a 4-byte timestamp, which was received with the PingRequest request.

10.1 Stream Begin

10.2 record

10.3 Stream EOF


分类: [RTMP](#)

标签: [rtmp](#)

好文要顶

关注我

收藏该文



季末的天堂

关注 - 0

粉丝 - 40

+加关注

4

0

« 上一篇: [Nginx-rtmp之 ngx_rtmp_send.c 文件分析](#)
» 下一篇: [Nginx-rtmp直播之业务流程分析](#)

posted @ 2018-05-04 11:36 季末的天堂 阅读(9237) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能发表评论, 立即 [登录](#) 或 [注册](#), [访问](#) [网站首页](#)

- 【推荐】阿里出品, 对标P7! 限时免费, 七天深入MySQL实战营报名开启
- 【推荐】与开发者在一起, 云计算领导者AWS入驻博客园品牌专区
- 【推荐】大型组态、工控、仿真、CADGIS 50万行VC++源码免费下载
- 【推荐】第一个NoSQL数据库, 在大规模和一致性之间找到了平衡
- 【推荐】了不起的开发者, 挡不住的华为, 园子里的品牌专区
- 【推荐】未知数的距离, 毫秒间的传递, 声网与你实时互动
- 【推荐】七牛云新老用户同享 1 分钱抢 CDN 1TB流量大礼包!

相关博文:

- 悄悄直播 (三) ——搭建rtmp服务器 (smart_rtmpd-rtmp服务器搭建)
- rtmp服务器创建和设置
- linuxCentOS7nginxnginx-rtmp-module搭建直播
- 海康摄像头web观看操作rtmp/hls推流rtsp流转rtmp流
- ffmpeg+nginx实现rtsp转rtmp并通过nginx转发
- » 更多推荐...

最新 IT 新闻:

- 疫情防控 《王者荣耀》冬季冠军杯总决赛不公开售票
- 比特币挖矿约12个月才回本: 去年涨幅超900%
- 饿了么申请“饿了伐”、“饿了妹”、“饿鸟撒”等商标
- 新版Flash Player推送: Win7以下系统不再支持视频格式内容播放
- 盒马鲜生吐槽商标被抢注: 怎么多了一堆亲戚
- » 更多新闻...