RTMP(Real-time Messaging Protocol)介绍

1. RTMP 定义

实时消息协议（RTMP）是 Adobe 公司开发的专用协议，是为了在 flash 播放器和服务器之间通过因特传输的音频，视频及其他数据流的协议。

2. RTMP Chunk Stream

2.1 简介

RTMP 消息块流是为多媒体流提供多路传输和数据打包而设计的应用级协议，尽管 RTMP 消息块流设计时是为了与 RTMP 协议一起工作，但是他也可以处理任何传送消息流的协议，每一个消息包含时间戳和有效负载类型标示，RTMP 消息块流和 RTMP 一起适用于多样性音视频应用程序，包括一对一和一对多直播和视频点播服务以及交互式会议应用程序。当同像 TCP 这样的可靠传输协议一起应用时，RTMP 消息块流提供可靠的时间戳进行端到端全信息传送。RTMP 消息块流不提供任何控制的优先级别和相似功能，但是可以使用更高级协议来提供这样的优先级控制操作。

2.2 专有词汇解释：

包：

一个数据包由固定的包头和有效负载数据组成，一些底层协议或许需要包的封装来被定义。

有效负载：

包含在包中的数据，就像音频样本或者压缩的视频数据。

端口：

在 TCP/ＩP 协议中定义的用正整数表示的端口号用于在传输中提取以区分目标主机的不同应用。

地址：

网络地址和端口的组合识别一个传输层终端端口，例如一个 IP 地址和 TCP 端口，数据包从一个源传输层地址传送到目标段的传输层地址。

消息流：

一个通信的逻辑通道，允许消息流通。

消息流 ID：

每一个消息拥有一个分配的 ID 来识别正在流动消息流。

消息块：

消息的片段，消息被分成小的部分，在他们在网络中发送之前交叉存储。消息块确保定制时间戳的端到端全消息传送，穿过多层流。

消息块流：

一个通信的逻辑通道，允许消息块在一个特定的方向上流通，消息块流可以从客户端传送到服务器，也可以相反。

消息块流 ID：

每一个消息块有一个分配的 ID 用于识别正在流动的消息块流。

复合技术：

把分开的音视频数据组合成一条音视频流的过程，使同时传送许多音视频数据成为可能。

逆复合技术：

复合的反向过程，交叉存取组装的音频视频数据，使他们成为最初的音视频数据。

2.3 消息格式

RTMP 消息包含两个部分，包头和有效负载。包头包含时间戳、消息长度、消息类型以及消息流 ID。有效负载是包含在消息中的实际数据，例如：它可以是一些音频样本或者压缩的视频数据。

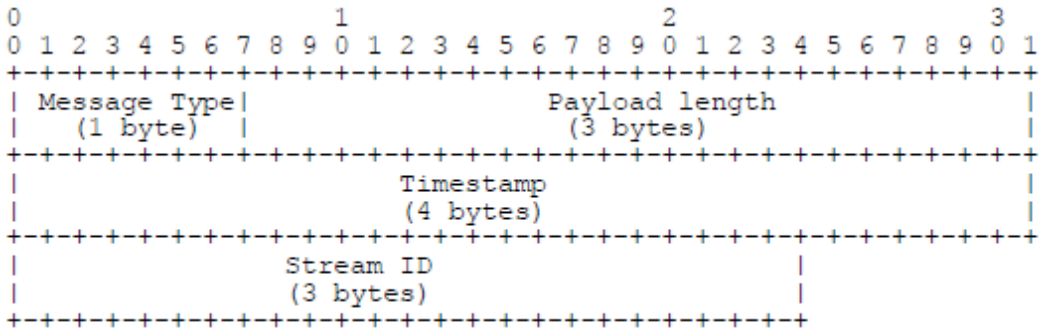


Figure 1 Message header

(1)、时间戳 delta（timestamp delta）：消息的时间戳，占 3 个字节

(2)、消息长度（message length）：3 个字节。消息的有效负载的长度，如果消息头不能被省略，他应该包含在长度中，这个字段在消息块包头中。注意这一般和消息块的有效负载长度是不一样的。消息块的有效负载长度是除了最后一个消息块外其他所有消息块都取消息块大小的最大值（通常默认为 128 字节）。

(3)、消息类型 id（message type id）：1 个字节。协议控制消息的类型字段的范围是被保留的，这些传播信息的消息被 RTMP 消息块和高层协议处理，所有其他的类型 ID 可被高层协议使用，被 RTMP 消息块当做不透明的值

(4)消息流 ID（message stream id）：4 个字节。消息流 ID 可以是任意值，不同的消息复用成同一消息块流以及解复用都是基于消息流 ID。

2.4 握手

(1)、一个 RTMP 通信以握手开始，RTMP 中的握手不像其他的协议，他包含三个固定长度的块。而不是带有头的可变长的块。

客户（发起连接的终端）和服务器每次发送同样的三个块，被客户段发送的消息块被指定为 C0，C1，C2，被服务器端发送的消息被指定为 S0，S1，S2。

(2)、握手的顺序

握手以客户端发送 C0 和 C1 消息块位开始，客户端必须等到 S1 到达在发送 C2。客户端必须等到 S2 接收到才可以发送其他的数据；服务端必须等到 C0 到达才发送 S0 和 S1，也许会等到 C1 之后再发送。服务端必须等到 C1 到达才发送 S2，服务端必须等到 C2 到达后才发送其他数据。

C0/S0:

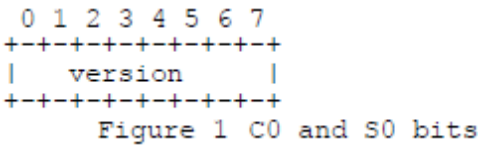


Figure 1 C0 and S0 bits

Version: 8 bits

In C0, this field identifies the RTMP version requested by the client. In S0, this field identifies the RTMP version selected by the server. The version defined by this specification is 3.

C1/S1:

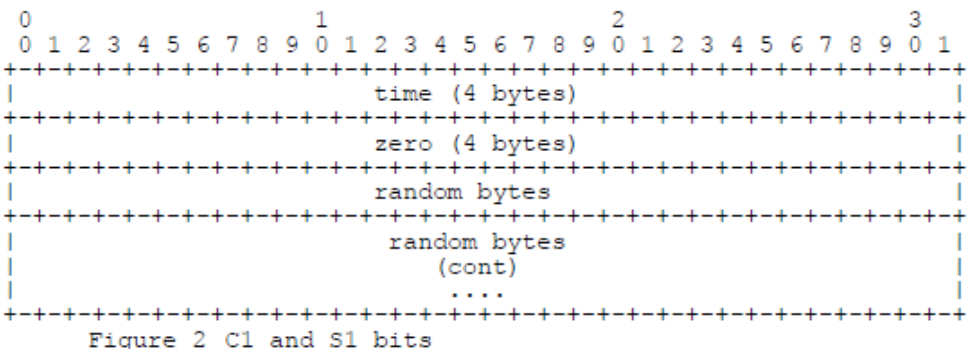


Figure 2 C1 and S1 bits

Time: 4 bytes

This field contains a timestamp, which SHOULD be used as the epoch for all future chunks sent from this endpoint. This may be 0, or some arbitrary value. To synchronize multiple chunkstreams, the endpoint may wish to send the current value of the other chunkstream's timestamp.

Zero: 4 bytes This field MUST be all 0s.

Random data: 1528 bytes

This field can contain any arbitrary values. Since each endpoint has to distinguish between the response to the handshake it has initiated and the handshake initiated by its peer,this data SHOULD send something sufficiently random. But there is no need for cryptographically-secure randomness, or even dynamic values.

C2/S2:

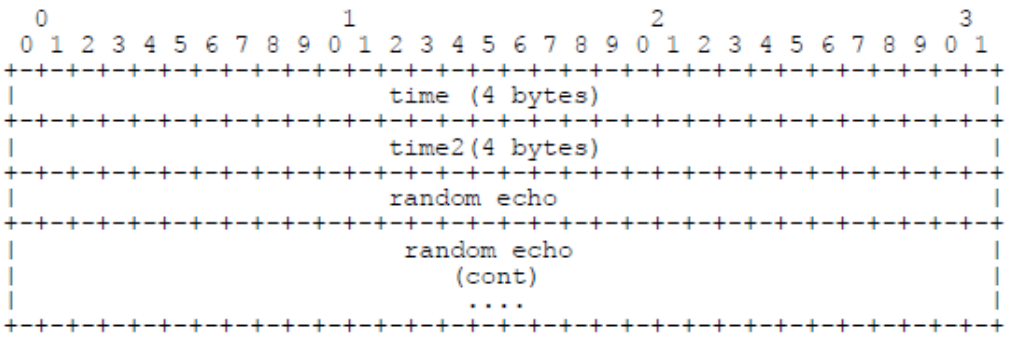


Figure 3 C2 and S2 bits

Time: 4 bytes

This field MUST contain the timestamp sent by the peer in S1 (for C2) or C1 (for S2).

Time2: 4 bytes

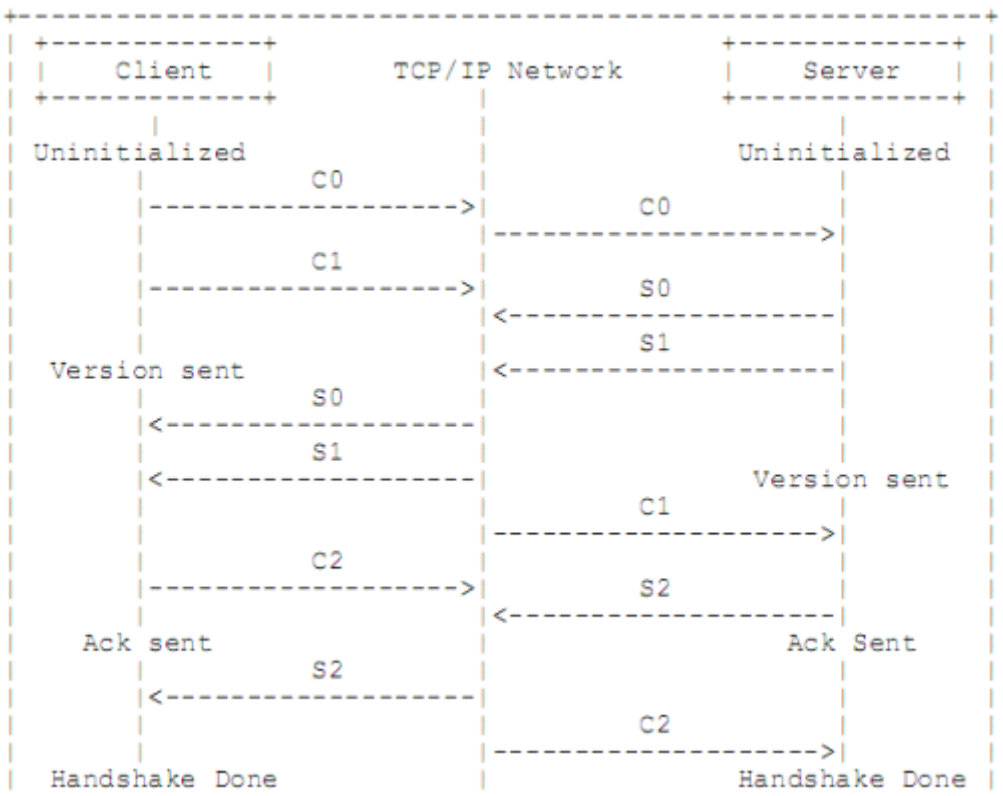
This field MUST contain the timestamp at which the previous packet(s1 0r c1) sent by the peer was read.

Random echo: 1528 bytes

This field MUST contain the random data field sent by the peer in S1 (for C2) or S2 (for C1).

Either peer can use the time and time2 fields together with the current timestamp as a quick estimate of the bandwidth and/or latency of the connection, but this is unlikely to be useful.

(3)、握手过程



下面的表格表述了提到了在握手阶段的状态

阶段	描述
未初始化	协议版本在这个阶段中被发送，客户端及服务端没有初始化，客户端在 C0 中发送协议版本，如果服务端支持版本，就在回应中发送 S0 和 S1，如果不能，服务端通过适当的行为进行恢复，在 RTMP 中，这个行为是连接结束
版本发送	在未初始化阶段以后客户端和服务端在版本发送阶段，客户端等待 S1 包，服务端等待 C1 包，当接收到想要的包，客户端发送 C2，服务端发送 S2，此时阶段变成了 ACK 的发送。
ACK 发送	客户端和服务端分别等待 S2 和 C2
握手完成	客户端和服务交换消息。

2.5 消息分块

(1)、在握手以后，连接一个或者多个复用消息块，每个消息块流负载来自消息流中同一类的消息，每一个消息块的产生配有一个唯一的 ID，被称为消息块流 ID，消息块流在网络上传输。传输的时候每一个消息块在下一个消息块之前必须完全被发送，当接收结束，消息块基于消息块流 ID 被组装成消息。消息分块允许高层协议的大的消息被分割成小的消息，消息分块也允许小的消息被传送时带有小的包头，消息块的包头包含了信息的压缩表示，否则信息必须在消息自身中包含。

(2)、消息块的字节顺序：

RTMP 的字节序和大多数网络协议一样是大端序，也有一些字段是小端序的，不过都有特殊的说明。

1).RTMP 的 head 组成

RTMP 的 head 在协议中的表现形式是 chunk head，前面已经说到一个 Message+ head 可以分成一个和多个 chunk，为了区分这些 chunk，肯定是需要一个 chunk head 的，具体的实现就把 Message head 的信息和 chunk head 的信息合并在一起以 chunk head 的形式表现。

一个完整的 chunk 的组成如下图所示

Chunk basic header：1 to 3

该字段包含 chunk 的 stream ID 和 type 。chunk 的 Type 决定了消息头的编码方式。该字段的长度完全依赖于 stream ID，该字段是一个可变长的字段。

Chunk Msg Header：0, 3 ,7, 11

该字段包含了将要发送的消息的信息（或者是一部分，一个消息拆成多个 chunk 的情况下是一部分）该字段的长度由 chunk basic header 中的 type 决定。

Extend Timestamp： 0 ,4 bytes

该字段发送的时候必须是正常的时间戳设置成 0xffffff 时，当正常时间戳不为 0xffffff 时，该字段不发送。当时间戳比 0xffffff 小该字段不发送，当时间戳比 0xffffff 大时该字段必须发送，且正常时间戳设置成 0xffffff。

Chunk Data 实际数据（Payload），可以是信令，也可以是媒体数据。

Chunk basic header：

chunk basic head 的长度为 1~3 个字节，具体长度主要是依赖 chunk stream ID 的长度，所谓 chunk stream ID 是 flash server 用来管理连接的客户端的信令交互的标识，协议最大支持 65597 个 streamID 从 3~65599。ID 0, 1, 2 为协议保留，0 代表 ID 是 64~319 (第二个 byte + 64)；1 代表 chunk stream ID 为 64~65599 ((第三个 byte) * 256 + 第二个 byte + 64)（小端表示）；2 代表该消息为低层的协议（在 RTMP 协议中控制信令的 chunk stream ID 都是 2）。3~63 的 chunk stream ID 就是该 byte 的值。没有附加的字段来标识 chunk stream ID。

目前 chunk basic head 的长度一般为 1 个字节。这一个字节由两部分组成

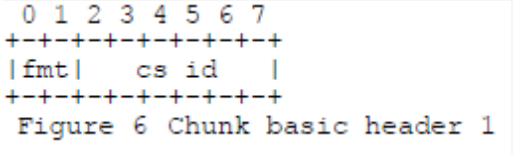


Figure 6 Chunk basic header 1

fmt 占两个 bit 用来标识紧跟其后的 chunk Msg Header 的长度，cs id 占六个 bit。

两位的 fmt 取值为 0~3，分别代表的意义如下：

case 0：chunk Msg Header 长度为 11；

case 1：chunk Msg Header 长度为 7；

case 2：chunk Msg Header 长度为 3；

case 3：chunk Msg Header 长度为 0；所以 只有一个字节的 chunk basic header 取值为 chunk basic header = (fmt << 6) | (cs id).

其他的两种的字节情况分别为：

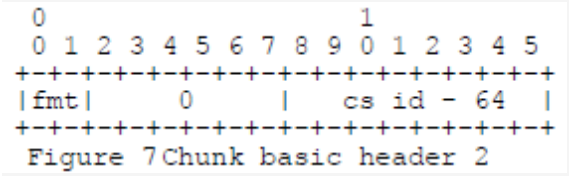


Figure 7 Chunk basic header 2

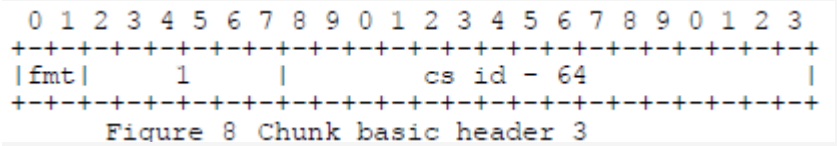


Figure 8 Chunk basic header 3

Chunk Msg Header：

Chunk Msg Header 的长度是可变的，Chunk Msg Header 可变的原因是为了压缩传输的字节数，把一些相同类型的 chunk 的 head 去掉一些字节，换句话说就是四种类型的包头都可以通过一定的规则还原成 11 个字节，这个压缩和还原在 RTMP 协议中称之为复用/解复用。

那我们以 11 个字节的完整包头来解释 Chunk Msg Header，如图所示

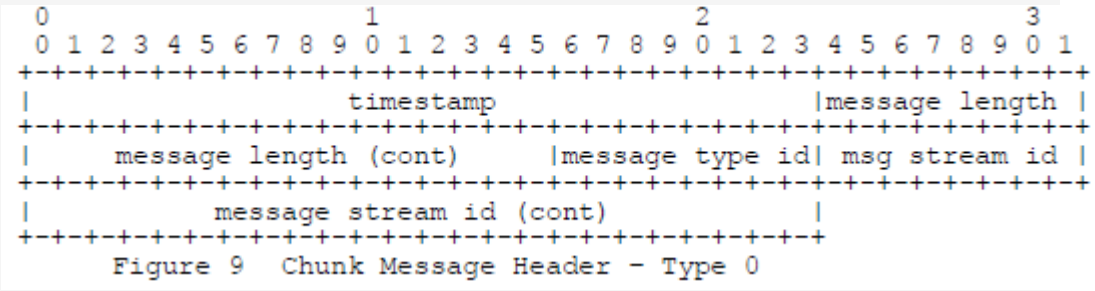


Figure 9 Chunk Message Header - Type 0

Timestamp：3bytes

对于 type 0 的 chunk，绝对时间戳在这里表示，如果时间戳值大于等于 0xffffff（16777215），该值必须是 0xffffff,且时间戳扩展字段必须发送，其他情况没有要求。

message length：3bytes

Message 的长度，注意这里的长度并不是跟随 chunk head 其后的 chunk data（Payload）的长度，而是前文提到的一条信令或者一帧视频数据或音频数据的长度。前文提到过信令或者媒体数据都称之为 Message，一条 Message 可以分为一条或者多条 chunk。

message type id：1byte

message stream id：4bytes

message stream id的字节序是小端序，这个字段是为了解复用而设计的，RTMP文档上说的相当的模糊：The message stream ID can be any arbitrary value. Different message streams multiplexed onto the same chunk stream are demultiplexed based on their message stream IDs. Beyond that, as far as RTMP Chunk Stream is concerned, this is an opaque value. This field occupies 4 bytes in the chunk header in little endian format.

长度是 7 bytes 的 chunk head，该类型不包含 stream ID，该 chunk 的 streamID 和前一个 chunk 的 stream ID 是相同的，变长的消息，例如视频流格式，在第一个新的 chunk 以后使用这种类型，注意其中时间戳部分是相对时间，为与上一个绝对时间之间的差值 如图所示：

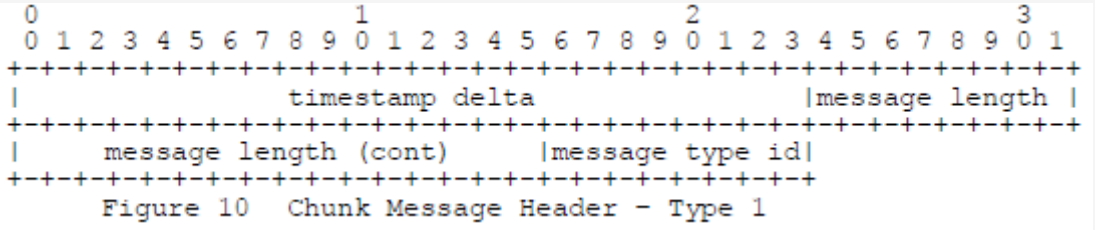


Figure 10 Chunk Message Header - Type 1

3 bytes 的 chunk head，该类型既不包含 stream ID 也不包含消息长度，这种类型用于 stream ID 和前一个 chunk 相同，且有固定长度的信息，例如音频流格式，在第一个新的 chunk 以后使用该类型。如图所示：

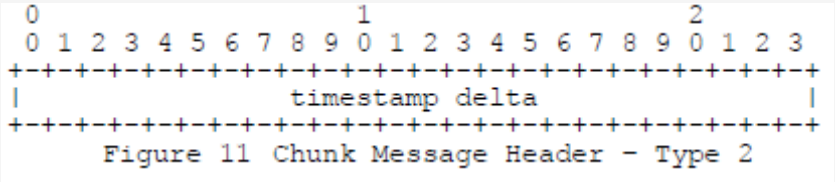


Figure 11 Chunk Message Header - Type 2

0 bytes 的 chunk head，这种类型的 chunk 从前一个 chunk 得到值信息，当一个单个消息拆成多个 chunk 时，这些 chunk 除了第一个以外，其他的都应该使用这种类型。

注：chunk 的长度：

chunk 的长度初始长度固定为 128 个字节，但是这个值并不是不可变的，在客户端和服务端建立连接以后，客户端和服务端都可以通过发送信令的方式来通知对端修改 chunk 的长度，理论上来说可以修改 chunk 的最长长度为 65536。这里 chunk 的长度是指 chunk 的数据部分的长度，即 chunk data（payload）的长度，如果一条 Message 的数据长度超过了 chunk 的长度，就必须把 Message 分割成多条 chunk，即如果一条视频类型 Message 长度为 2000 个 byte，chunk 长度为 1500，则该 Message

将会分割成两条 chunk，第一条的 chunk data 长度为 1500，第二条的 chunk data 长度为 500。当然这两条 chunk 的 chunk head 肯定是不同的，其中第二条 chunk 的 chunk head 就是 0 字节的。

2.6 示例：

说明明一个很长的消息被分割成很多消息块：

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

Figure 15 Sample Message to be broken to chunks

这里是分割出来的消息块：

	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

Figure 16 Format of each of the broken chunk.

注：消息块 1 的包头数据详细介绍了 307 个字节的消息的全部内容。前面一图为消息的格式，后面的图为消息块的格式，注意理解两者的关系。

3．RTMP 消息类型

3.1 简介

(1)、服务端和客户端交换的消息的不同类型包括用于发送音频数据的音频消息，发送视频的数据的视频消息，发送用户数据的数据消息，共享对象消息，和命令消息。共享对象消息一般提供在多个的客户和一个服务段之间管理分布式数据的方法。

RTMP 为协议控制消息保留了 1-7 的类型 ID，这些消息包含 RTMP 消息块流协议(有两个)或者 RTMP 本身需要的信息。

ID 为 1 和 2 时：协议消息被保留用作 RTMP 消息块流协议的使用；

ID 为 3-6：协议消息被保留用作 RTMP 的使用；

ID 为 7 时：协议消息被用在边缘服务和原服务之间

注意：本节重点介绍协议控制消息(protocol control messages)和 RTMP 命令消息(RTMP commands message)

(2)、专有词汇解释：

远程调用（RPC）：一个允许客户端或者服务端调用一个子程序或者进程的请求。

元数据：关于数据的表述，影片的元数据包括影片标题，持续时间，创造的数据等等。

应用实例：客户端通过发送连接请求连接服务器的应用程序的实例。
动作消息格式（AMF）：一种简洁的 2 进制编码用来连续化 ActionSctipt 对象。

3.2 协议控制消息 (protocol control messages)

RTMP Chunk Stream supports some protocol control messages. These messages contain information required by RTMP Chunk Stream protocol and will not be propagated to the higher protocol layers. Currently there are two protocol messages used in RTMP Chunk Stream. One protocol message is used for setting the chunk size and the other is used to abort a message due to non-availability of remaining chunks Protocol control messages SHOULD have message stream ID 0(called as control stream) and chunk stream ID 2, and are sent with highest priority.

协议控制消息主要有以下几个：

(1)、设置消息块大小(ID1)：协议控制消息 1，设置消息块大小，用来通知对方新的可供利用的最大的消息块的大小

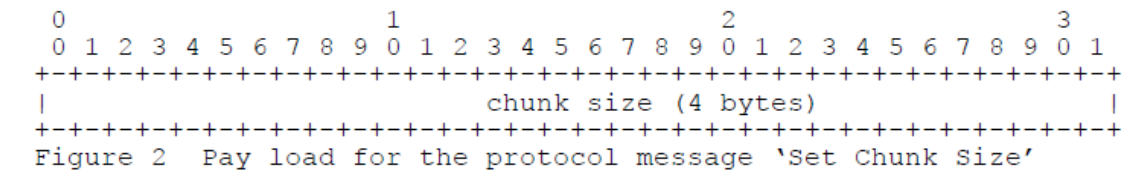


Figure 2 Pay load for the protocol message 'Set Chunk Size'

chunk size: 32 bits

This field holds the new chunk size, which will be used for all future chunks sent by this chunk stream.

(2)、中断消息(abort message:ID2): 这个消息在发送方已经发送部分消息后发送，但是想要告诉接收方余下的消息不用发送。

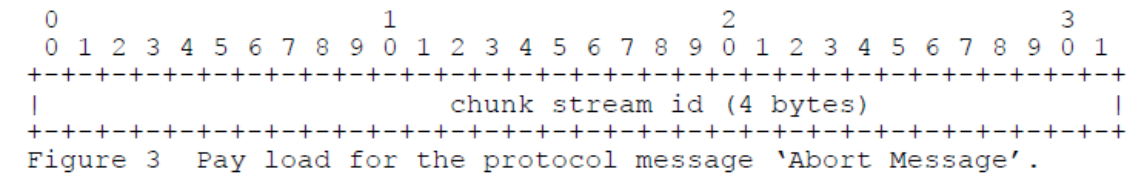


Figure 3 Pay load for the protocol message 'Abort Message'.

(3)、确认消息(Acknowledgement)：客户端或者服务端当接收到的字节和窗口大小一样时发送确认给对方，窗口大小是发送方发送的在没有从接收方收到确认之前的最大字节数，服务端在应用程序接通后给客户端发送窗口大小，这个消息指定了序号，就是目前为止收到的字节数。

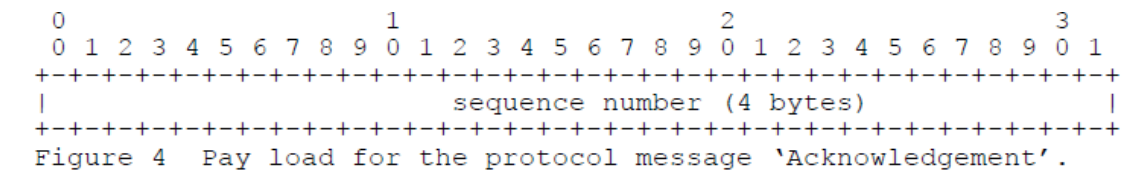


Figure 4 Pay load for the protocol message 'Acknowledgement'.

(4)、用户控制(User Control Message)：客户端或者服务端发送这个消息用来通知对方用户控制事件，这个消息负载事件类型和事件数据.

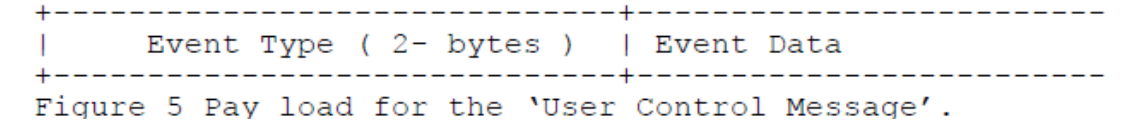


Figure 5 Pay load for the 'User Control Message'.

(5)、窗口确认大小：客户端或者服务端在发送确认时发送这个消息来告知对方使用的窗口大小。

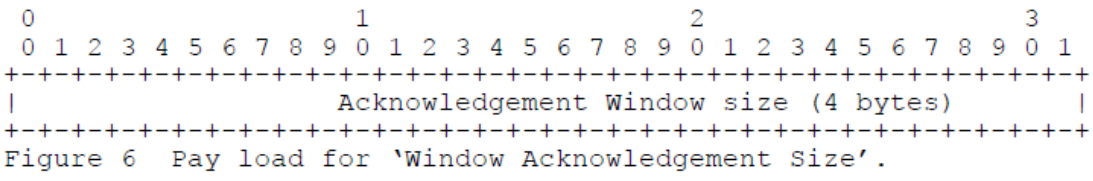


Figure 6 Pay load for 'Window Acknowledgement Size'.

(6)、设置带宽：客户端或者服务端发送这个消息用来更新对方的输出带宽，输出带宽的值和对方窗口大小一样。如果自己当前窗口大小和消息中接收到得不一样，对方发送“窗口确认大小“。

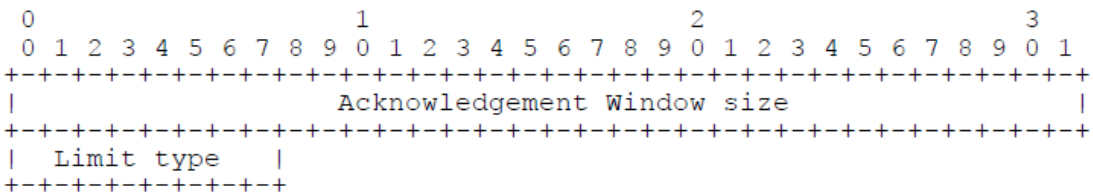


Figure 7 Pay load for 'Set Peer Bandwidth'

3.3 RTMP 命令消息(RTMP commands message)

3.3.1、RTMP 命令消息介绍：

命令消息负载了在客户端和服务端的 AMF 编码命令，当用 AMF0 是此时用命令消息是 Message type=20，而是 AMF3 时 message type 为 17.这些消息发送用来运行操作诸如：连接，创造流，发布，播放，暂停。像 onstatus,result 这样的命令用来通知发送方需求命令的状态，一个命令消息由命令名称，处理 ID 和命令对象，一个客户端或者服务端可以通过命令消息通信的流请求远程进程调用（RPC）。

3.3.2、命令的类型

客户端和务端交换的命令是用 AMF 编码的，发送方发送的命令消息由命令名称，处理 ID 和包含了相对参数的命令对象组成。例如:连接命令包含'app'参数，告诉服务端客户端连接的应用程序的名字。接收到的进程会以相同的 ID 发回应答，回应字符可以是_result,_error,_或者一个方法名称（verifyClient 或者 contactExternalServer）中的一种.

一个_result 或者_error 命令字符串发送回应信号，处理 ID 指示了回应消息中突出的命令，就像 IMAP 和许多其他的协议中的标签（tag）一样。在命令字符串中的方法名称代表了发送方试图在接收端运行一个方法（或动作）。下面的类对象用来发送多样性命令：

NetConnection---一个高层表示在服务端和客户端连接的高层协议。

NetStream---一个表述音视频或者其他流的通道的对象，我们还发送命令像 play, pause 等等用来控制数据的流通。

(1)、NetConnection 命令：Netconnection 在在客户端应用程序和服务端之间的双向连接，另外，他支持异步远程方法调用。下面的命令可以在 Netconnection 中发送 Connect、call、close、creatStream。(下面详细介绍下 connect 的情形，其他几个命令请参见标准文档)

Connect： 客户端向服务端发送连接（connect）命令请求连接一个服务应用实例。以下为命令的结构

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 Arguements	Object	Any optional information

Following is the description of the name-value pairs used in Command Object of the connect command.

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/test app/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 _SORENSON
pageUrl	String	URL of the web page from where the SWF file was loaded.	http://somehost/sample.html
objectEncoding	Number	AMF encoding method.	kAMF3

以下是服务端到客户端命令的结构:

Field Name	Type	Description
Command Name	String	_result or _error; indicates whether the response is result or error.
Transaction ID	Number	Transaction ID is 1 for call 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.

Figure 4 Message flow in the connect command

在命令执行过程中消息的流通过程：

客户端向服务端发送连接命令请求与服务端应用实例的连接。

当接收到连接命令后，服务端给客户端发送协议消息“窗口确认大小“，服务端也连接在连接命令中提及到得应用程序。

服务端向客户端发送协议消息“设置对端输出带宽“。

客户端在收“设置对端输出带宽“后向服务端发送“窗口确认大小”。

服务端向客户端发送（streamBegin）。

服务端发送 result 命令消息向客户端提供链接状态的情报（失败或者成功），这个命令明确了处理 ID（对于连接（connect）命令来说等同于 1），这个消息也明确了一些特性，比如 Flash medial Server 的版本，扩展其他连接的能力，级别的信息，编码，表述，对象编码等等。

Call：Call 方法在接收方运行远程进程，被呼叫的远程进程调用作为 call 命令的一个参数

createStream（创造流）：客户端向服务端发送这个消息来创造一个逻辑的通道以供消息通信，音视频的发布，元数据在使用 createStream 命令创建的流通道上实现

(2)、Netstream 定义了流视频，音频和数据消息可以流通的的通道，一个 NetConnection 对象可以支持许多数据流的 NetStream。以下命令可以在 NetStream 中发送 Play、play2、deletestream、closeStream、ReceiveAudio、receiveVideo、publish、seek、pause。

Play：

客户端向服务端发送这个命令来播放流，一个播放列表也可以使用这个命令被创建。如果你想创造动态的在不同直播或者录制的流中交换的播放列表，调用 play 多次，每次给 reset 传递 false。如果你想立刻播放确定的流，清除其他任何排列播放的流，给 reset 传递 true。

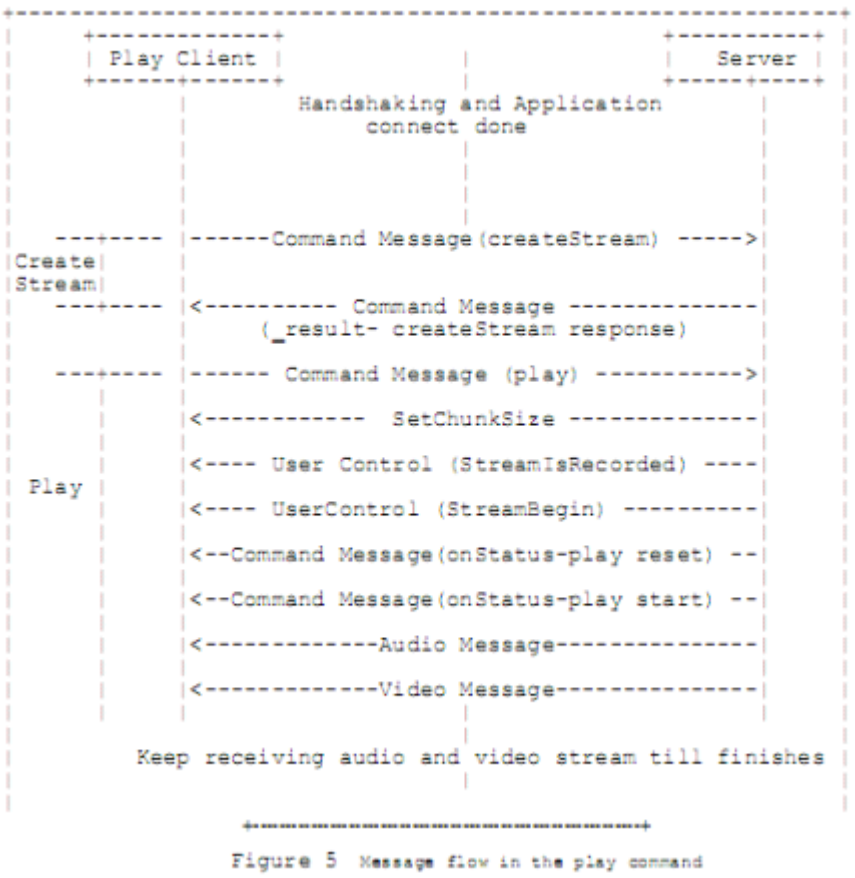


Figure 5 Message flow in the play command

在命令执行时的消息流通过程：

客户端在接收到创造流的 result 后发送播放命令。

当接收到播放命令，服务端发送一个协议消息来设置消息块大小。

服务端发送另一个协议消息（用户控制）来确定“StreamIsRecorded“事件和在消息这个消息中的流 ID，消息在前两个字节中负载有事件类型，最后四个字节中负载有流 ID。

服务端发送另一个协议消息确定事件“StreamBegin“，用来标示对客户端流的开始。

如果由客户端发送的 play 命令成功，服务端发送 OnStatus 命令消息 NetStream.Play.Start 和 NetStream.Play.Reset，在客户端发送的 Play 命令已经设置了 reset 标签服务端只发送 NetStream.Play.Reset。

在这之后，服务端发送客户端播放的音视频数据。

Play2：

与 play 命令不一样，play2 可以再不改变播放内容时间线的情况下交换成一个不同的比特率流。服务端为支持的比特率保持多个文件，客户端可以在 play2 中请求。

deleteStream：NetStream 在 NetStream 对象被毁坏后发送删除流命令。

receiveAudio：NetStream 发送接收音频消息来通知服务端是否发送或者不发送音频给客户端。

receiveVideo：NetStream 发送接收视频消息来通知服务端是否发送视频给客户端。

Publish：客户端发送发布命令向服务端发布命名的流，使用这个命名的名称，任何客户可以播放这个流，接收发布的音频、视频和数据消息。

seek：客户端发送搜寻命令搜寻在媒体文件或者播放列表中的偏移量（毫秒为单位）。

pause：客户端发送 pause 命令告诉服务端暂停或者开始播放。

4. 示例：

4.1 AMF0 数据类型

Rtmp 包默认的最大长度为 128 字节,(或通过 chunksize 改变 rtmp 包最大长度), 当 AMF 数据超过 128Byte 的时候就可能有多个 rtmp 包组成,如果需要解码的 rtmp 包太长则被 TCP 协议分割成多个 TCP 包.那么解码的时候需要先将包含 rtmp 包的 tcp 封包合并, 再把合并的数据解码,解码后可得到 amf 格式的数据,将这些 AMF 数据取出来就可以对 AMF 数据解码了.服务器和 Flash 客户端之间的命令都是用 AMF 格式的数据在传送,例如 connect() publish()等命令和服务器 as 脚本里面自己定义的一些方法. 常用的数据类型整理如下:

类型说明(ObjType)	数据	dataSize
CORE_String	0x02	2 字节 (2 字节的数据纪录了 String 的实际长度)
CORE_Object	0x03	0 字节（开始嵌套 0x00000009 表示嵌套结束）
NULL	0x05	0 字节 空字节无意义
CORE_NUMBER	0x00	8 字节
CORE_Map	0x08	4 字节（开始嵌套）
CORE_BOOLEAN	0x01	1 字节

AMF0 数据的嵌套关系如下:

Object={ObjType + ObjValue}

ObjType={CORE_BOOLEAN、CORE_NUMBER、CORE_STRING、CORE_DATE、CORE_ARRAY、CORE_MAP、CORE_OBJECT}

CORE_BOOLEAN={Value(1 Byte)}

CORE_NUMBER={Value(8 Byte)}

CORE_String={StringLen(2 Byte) + StringValue(StringLen Byte)}

CORE_DATE={value(10 Byte)}

CORE_Array={ArrayLen(4 Byte) + Object}

CORE_Map={MapNum(4 Byte) + CORE_Object}

CORE_Object={CORE_String + Object}

4.2 示例 1：

例如完成握手后,Flash 向 FMS 发送的第一个 RTMP 数据,内容如下:

```
03 00 00 00 00 00 BC 14   00 00 00 00 02 00 07 63  -  ........ .......c
6F 6E 6E 65 63 74 00 00   00 00 00 00 00 00 00 08  -  onnect.. ........
00 00 00 00 00 0E 6F 62   6A 65 63 74 45 6E 63 6F  -  ......ob jectEnco
64 69 6E 67 00 00 00 00   00 00 00 00 00 00 03 61  -  ding.... .......a
70 70 02 00 0B 6D 65 64   69 61 73 65 72 76 65 72  -  pp...med iaserver
00 04 66 70 64 61 01 00   00 05 74 63 55 72 6C 02  -  ..fpda.. ..tcUrl.
00 1C 72 74 6D 70 3A 2F   2F 31 32 37 2E 30 2E 30  -  ..rtmp:/ /127.0.0
2E 31 2F 6D 65 64 69 61   73 65 72 76 65 72 00 0B  -  .1/media server..
61 75 64 69 6F 43 6F 64   65 63 73 00 C3 40 83 38  -  audioCod ecs..@.8
00 00 00 00 00 00 0B 76   69 64 65 6F 43 6F 64 65  -  .......v ideoCode
63 73 00 40 53 00 00 00   00 00 00 00 00 00 09 02  -  cs.@S... ........
09 50 55 42 4C 49 53 48   45 52 02 00 0C 73 74 72  -  .PUBLISH ER...str
65 61 6D 52 65 63 6F 64   65                       -  eamRecod e
```

上面一段数据由 2 个 RTMP 包组成,2 个 RTMP 包头分别用蓝色表示,第一个蓝色的是 12 字节的包头,后面一个蓝色的 C3 是一个字节的包头,绿色部分是 AMF 数据,红色的是 AMF 数据类型,整个 RTMP 解码过程如下

[2008-06-18 16:59:20] DecodeInvoke:
[2008-06-18 16:59:20] InvokeName:String:connect
[2008-06-18 16:59:20] InvokeID:Double:0
[2008-06-18 16:59:20] Map:MapNum:0
[2008-06-18 16:59:20] Params:{
[2008-06-18 16:59:20] Key:String:objectEncoding
[2008-06-18 16:59:20] Value:Double:0
[2008-06-18 16:59:20] Key:String:app
[2008-06-18 16:59:20] Value:String:mediaserver
[2008-06-18 16:59:20] Key:String:fpda
[2008-06-18 16:59:20] Value:Bool:0
[2008-06-18 16:59:20] Key:String:tcUrl
[2008-06-18 16:59:20] Value:String:rtmp://127.0.0.1/mediaserver
[2008-06-18 16:59:20] Key:String:audioCodecs
[2008-06-18 16:59:20] Value:Double:615
[2008-06-18 16:59:20] Key:String:videoCodecs
[2008-06-18 16:59:20] Value:Double:76
[2008-06-18 16:59:20] }End Params
[2008-06-18 16:59:20] InvokeParams:String:PUBLISHER
[2008-06-18 16:59:20] InvokeParams:String:streamRecode

03 表示 12 字节头，channelid=3

000000 表示 Timmer=0

000102 表示 AMFSize=18

14 表示 AMFType=Invoke 方法调用

00 00 00 00 表示 StreamID = 0

02 表示 String

0007 表示 String 长度 7

63 6F 6E 6E 65 63 74 是 String 的 Ascall 值"connect"

00 表示 Double

示例 2:：从 connet 到 publish 数据包

1,C->S,连接

0080: 03 00 00 00 00 01 ……..

0090: 9B 14 00 00 00 00 02 00 07 63 6F 6E 6E 65 63 74 .....….connect

00A0: 00 3F F0 00 00 00 00 00 00 03 00 03 61 70 70 02 .?……..app.

00B0: 00 05 35 32 6E 74 75 00 08 66 6C 61 73 68 56 65 ..52ntu..flashVe

00C0: 72 02 00 0D 57 49 4E 20 39 2C 30 2C 31 32 34 2C r...WIN 9,0,124,

00D0: 30 00 06 73 77 66 55 72 6C 02 00 53 66 69 6C 65 0..swfUrl..Sfile

00E0: 3A 2F 2F 2F 43 3A 2F 44 6F 63 75 6D 65 6E 74 73 :///C:/Documents

00F0: 25 32 30 61 6E 64 25 32 30 53 65 74 74 69 6E 67 %20and%20Setting

0100: 73 2F 67 6F 6E 67 78 69 61 6F 68 75 2F E6 A1 8C s/gongxiaohu/...

0110: E9 9D A2 2F 41 53 C3 33 2F 74 65 73 74 2F 62 69 .../AS.3/test/bi

0120: 6E 2D 64 65 62 75 67 2F 74 65 73 74 2E 73 77 66 n-debug/test.swf

0130: 00 05 74 63 55 72 6C 02 00 19 72 74 6D 70 3A 2F ..tcUrl...rtmp:/

0140: 2F 36 31 2E 31 35 35 2E 38 2E 32 32 30 2F 35 32 /61.155.8.220/52

0150: 6E 74 75 00 04 66 70 61 64 01 00 00 0C 63 61 70 ntu..fpad....cap

0160: 61 62 69 6C 69 74 69 65 73 00 40 2E 00 00 00 00 abilities.@.....

0170: 00 00 00 0B 61 75 64 69 6F 43 6F 64 65 63 73 00 ....audioCodecs.

0180: 40 99 9C 00 00 00 00 00 00 0B 76 69 64 65 6F 43 @.……..videoC

0190: 6F 64 65 63 73 00 40 C3 6F 80 00 00 00 00 00 00 [url=]odecs.@.o[/url]..…..

01A0: 0D 76 69 64 65 6F 46 75 6E 63 74 69 6F 6E 00 3F .videoFunction.?

01B0: F0 00 00 00 00 00 00 00 07 70 61 67 65 55 72 6C ..……..pageUrl

01C0: 02 00 54 66 69 6C 65 3A 2F 2F 2F 43 3A 2F 44 6F ..Tfile:///C:/Do

01D0: 63 75 6D 65 6E 74 73 25 32 30 61 6E 64 25 32 30 cuments%20and%20

01E0: 53 65 74 74 69 6E 67 73 2F 67 6F 6E 67 78 69 61 Settings/gongxia

01F0: 6F 68 75 2F E6 A1 8C E9 9D A2 2F 41 53 33 2F 74 ohu/...../AS3/t

0200: 65 73 74 2F 62 69 6E 2D 64 65 62 75 67 2F 74 65 est/bin-debug/te

0210: 73 74 2E 68 74 6D 6C 00 C3 0E 6F 62 6A 65 63 74 st.html...object

0220: 45 6E 63 6F 64 69 6E 67 00 00 00 00 00 00 00 00 Encoding.....

0230: 00 00 00 09

头:[12|3],[时间戳 0],[长度 n],[类型 invoke],[媒体频道 0]

内容:connect invokeid object,一系列参数

2、S->C,BW(BandWidth??)

0030: 02 00 00 00 00 00 04 05 00 00 ……….

0040: 00 00 00 13 12 D0 02 00 00 00 00 00 05 06 00 00 ……..

0050: 00 00 00 13 12 D0 02 02 00 00 00 00 00 0E 04 00 ……..

0060: 00 00 00 00 08 00 00 00 00 00 00 00 01 07 FB 41 ……..A

0070: 50 P

头[12|2],[时间戳 0],[长度 4],[类型 Server down],[媒体频道 0]

内容:00 13 12 D0

头[12|2],[时间戳 0],[长度 5],[类型 Client up],[媒体频道 0]

内容:00 13 12 D0 02

头[12|2],[时间戳 0],[长度 14],[类型 Server down],[媒体频道 0]

内容:00 08 00 00 00 00 00 00 01 07 FB 41 50

3、C->S,BW(BandWidth??)

0030: 02 8D 35 45 00 00 04 05 00 00 ……5E……

0040: 00 00 00 13 12 D0 ……

头:[12|2],[时间戳 8D 35 45],[长度 4],[类型 Server down],[媒体频道 0]

内容:00 13 12 D0

4、S->C,回应连接

0040: 03 00 00 00 00 00 73 14 ……..s.

0050: 00 00 00 00 02 00 07 5F 72 65 73 75 6C 74 00 3F ……._result.?

0060: F0 00 00 00 00 00 00 05 03 00 05 6C 65 76 65 6C ……..level

0070: 02 00 06 73 74 61 74 75 73 00 04 63 6F 64 65 02 ...status..code.

0080: 00 1D 4E 65 74 43 6F 6E 6E 65 63 74 69 6F 6E 2E ..NetConnection.

0090: 43 6F 6E 6E 65 63 74 2E 53 75 63 63 65 73 73 00 Connect.Success.

00A0: 0B 64 65 73 63 72 69 70 74 69 6F 6E 02 00 15 43 .de[url=][url=]script[/url][/url]ion...C

00B0: 6F 6E 6E 65 63 74 69 6F 6E 20 73 75 63 63 65 65 onnection succee

00C0: 64 65 64 2E 00 00 09 ded....

头:[12|3],[时间戳 0],[长度 n],[类型 invoke],[媒体频道 0]

内容:_result invokeid null object,一系列参数

5、C->S,建立流

0030: 03 00 08 15 00 00 19 14 00 00 ...s……..

0040: 00 00 02 00 0C 63 72 65 61 74 65 53 74 72 65 61 .....createStrea

0050: 6D 00 40 00 00 00 00 00 00 00 05 m.@……..

头:[12|3],[时间戳 00 08 15],[长度 19],[类型 invoke],[媒体频道 0]

内容:createStream invokeid null

6、S->C,回应建立流

0030: 03 00 00 00 00 00 1D 14 00 00 ……….

0040: 00 00 02 00 07 5F 72 65 73 75 6C 74 00 40 00 00 ....._result.@..

0050: 00 00 00 00 00 05 00 3F F0 00 00 00 00 00 00 .....?.....

头:[12|3],[时间戳],[长度 1D],[类型 invoke],[媒体频道 0]

内容:_result invokeid null flvchannel->(double)3F F0

7、C->S,发布流

0040: 08 00 08 16 00 00 22 14 ……..".

0050: 01 00 00 00 02 00 07 70 75 62 6C 69 73 68 00 00 .....publish..

0060: 00 00 00 00 00 00 00 05 02 00 04 6D 79 74 68 02 ……..myth.

0070: 00 04 6C 69 76 65 ..live

头:[12|8],[时间戳 00 08 16],[长度 22],[类型 invoke],[媒体频道 01 00 00 00]

内容:publish invokeid(0) null myth live,注，最后分别是发布号和发布类型

8、S->C,回应发布流

0040: 04 00 00 00 00 00 82 14 ……….

0050: 01 00 00 00 02 00 08 6F 6E 53 74 61 74 75 73 00 .....onStatus.

0060: 00 00 00 00 00 00 00 00 05 03 00 05 6C 65 76 65 ……..leve

0070: 6C 02 00 06 73 74 61 74 75 73 00 04 63 6F 64 65 l...status..code

0080: 02 00 17 4E 65 74 53 74 72 65 61 6D 2E 50 75 62 ...NetStream.Pub

0090: 6C 69 73 68 2E 53 74 61 72 74 00 0B 64 65 73 63 lish.Start..desc

00A0: 72 69 70 74 69 6F 6E 02 00 16 6D 79 74 68 20 69 ription...myth i

00B0: 73 20 6E 6F 77 20 70 75 62 6C 69 73 68 65 64 2E s now published.

00C0: 00 08 63 6C 69 65 6E 74 69 64 00 41 9F ED 05 40 ..clientid.A...@

00D0: 00 00 00 00 C4 00 09 …..

头:[12|4],[时间戳 0],[长度 82],[类型 invoke],[媒体频道 01 00 00 00]

内容 nStatus invokeid(0) null object->一系列参数 clientid=00 41 9F ED 05 40 00 00

9、C->S,发数据包,video data

0030: 06 00 08 16 00 03 35 09 01 00 ……..5...

0040: 00 00 12 00 00 84 00 58 42 00 A7 CF CF CF CF FE .....XB.....

0050: FF 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 FF BF CF 9F .>……..

0060: 9F 9F 9F FD FE 7C FC FC FC FF EF EC F9 F9 F9 F9 …..|……..

0070: FF E6 2E 79 FE 91 23 E7 E7 E7 E7 FF B6 AF 5E FF ...y..#……^.

0080: 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 FF BF CF 9F 9F >…….

0090: 9F 9F FD FE 7C FC FC FC FF EF F3 E7 E7 E7 E7 FF ….|…….

00A0: 7F 9F 3F 3F 3F 3F FB FC F9 F9 F9 F9 FF DF E7 CF ..????……..

00B0: CF CF CF FE FF 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 …..>…….

00C0: FF BF C6 CF 9F 9F 9F 9F FD FE 7C FC FC FC FF EF ……..|….

00D0: F3 E7 E7 E7 E7 FF 7F 9F 3F 3F 3F 3F FB FC F9 F9 ……..????....

00E0: F9 F9 FF DF E7 CF CF CF CF FE FF 3E 7E 7E 7E 7F ……..>….

00F0: F7 F9 F3 F3 F3 F3 FF BF CF 9F 9F 9F 9F FD FE 7C ……..|

0100: FC FC FC FF EF F3 E7 E7 E7 E7 FF 7F 9F 3F 3F 3F ……..???

0110: 3F FB FC F9 F9 F9 F9 FF DF E7 CF CF CF CF FE FF ?……..

0120: 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 FF BF CF 9F 9F >…….

0130: 9F 9F FD FE 7C FC FC FC FF EF F3 E7 E7 E7 E7 FF ….|…….

0140: 7F 9F 3F C6 3F 3F 3F FB FC F9 F9 F9 F9 FF DF E7 ..?.???……..

0150: CF CF CF CF FE FF 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 …..>…….

0160: F3 FF BF CF 9F 9F 9F 9F FD FE 7C FC FC FC FF EF ……..|….

0170: F3 E7 E7 E7 E7 FF 7F 9F 3F 3F 3F 3F FB FC F9 F9 ……..????....

0180: F9 F9 FF DF E7 CF CF CF CF FE FF 3E 7E 7E 7E 7F ……..>….

0190: F7 F9 F3 F3 F3 F3 FF BF CF 9F 9F 9F 9F FD FE 7C ……..|

01A0: FC FC FC FF EF F3 E7 E7 E7 E7 FF 7F 9F 3F 3F 3F ……..???

01B0: 3F FB FC F9 F9 F9 F9 FF DF E7 CF CF CF CF FE FF ?……..

01C0: 3E 7E 7E 7E C6 7F F7 F9 F3 F3 F3 F3 FF BF CF 9F >…….

01D0: 9F 9F 9F FD FE 7C FC FC FC FF EF EC BC FC FC FC …..|……..

01E0: CA A7 4E BF 0F 82 11 78 97 62 89 C6 7F B3 24 5D ..N....x.b....$]

01F0: EF F6 8D 5F CF 25 5A 8D 39 BE AB 21 FE 5A AE 7B ..._.%Z.9..!.Z.{

0200: 87 B9 BF F0 09 7C 1F 89 40 18 10 95 28 A5 FF 12 …..|..@...(...

0210: C4 9E 01 A9 61 72 A5 FA CA 03 1F FD 7B 2F 3F 3F ....ar……{/??

0220: 3F 38 04 3F F8 7C AC BE D1 E7 94 B2 C0 B5 FC FA ?8.?.|……..

0230: 27 3F 3F 3B F9 FF FF EB EB 9F 99 49 9F E7 CF CF '??;.....I....

0240: CF CF FE FF 3E C6 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 ....>.~~~.........

0250: FF BF B3 E7 E7 E7 E7 7F FF F5 83 CF E7 A7 CF CF ..~~~~~~~~~~~~~~

0260: CF CF FE FE CF 9F 9F 9F 9D FD 52 A1 EF 20 8F 5A ..~~~~~~R.. .Z

0270: 8D 1F 7F B5 ED B9 F1 FC EE E6 F7 F0 2E E7 7D 1E ..~~~~~~~~.}.

0280: 0F 84 A1 1F F3 EA C1 4D 9C 96 2F 4D 7F 9B 69 CE .....M../M..i.

0290: FE 7D 7F 3F 3B F2 B0 86 5E AE 59 CF 78 0C 7B 8E .}.?;...^.Y.x.{.

02A0: 7F 3D 3E 7E 7E 7E 7F F7 F6 1F 3B B9 B9 F9 EB DF .=>~~~~;~~~~

02B0: 00 30 49 2E 1F 6A 9C 95 7A EF FA E6 2E 77 F3 F3 .0I..j..z....w..

02C0: F3 BF 89 2A C7 AA C6 EF F7 4F BF BA 7C FC FC FC ...*.....O..|...

02D0: FF EF F3 E7 E7 E7 E7 FF 7F 9F 3F 3F 3F 3F FB FB ..~~~~~~~~????..

02E0: 3E 7E 7E 7E 77 F5 65 CA 08 7F E3 E7 CF CF CF CF >~~~w.e~~~~~~~

02F0: FE FF 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 FF BF CF ..>~~~~~~~~~~

0300: 9F 9F 9F 9F FD FE 7C FC FC FC FF EF F3 E7 E7 E7 ~~~~|~~~~~~~

0310: E7 FF 7F 9F 3F 3F 3F 3F FB FC F9 F9 F9 F9 FF DF ....????~~~~~~~

0320: E7 CF CF CF CF FE FF 3E 7E 7E 7E 7F F7 F6 7C FC ~~~~~~>~~~~~|.

0330: FC FC EF FE 03 55 23 FF CC CF 9F 9F 9F 9F FD FE .....U#~~~~~~~~

0340: 7C FC FC FC FF EF F3 C6 E7 E7 E7 E7 FF 7F 9F 3F |~~~~~~~~~~~~~?

0350: 3F 3F 3F FB FC F9 F9 F9 F9 FF DF E7 CF CF CF CF ???~~~~~~~~~~~~

0360: FE FF 3E 7E 7E 7E 7F F7 F9 F3 F3 F3 F3 FF BF CF ..>~~~~~~~~~~

0370: 9F 9F 9F 9F FD FE 7C FC FC FC FF EF E0 ~~~~|~~~~~~~~~~~~~~

头:[12|6],[时间戳 00 08 16],[长度 n],[类型 videodata],[媒体频道 01 00 00 00]