最详细的音视频流媒体传输协议-rtsp协议详解

零声Github分享官

流媒体传输协议-rtsp协议详解

参阅:RTSP协议详解和分析从零开始写一个RTSP服务器（一）RTSP协议讲解关于RTSP_RTP_RTCP协议的深刻初步介绍

rtsp

RTSP出现以前，最热的大概就是HTTP协议。想象一下，当你需要欣赏网络中的某一段视频，通过HTTP协议访问其URL、开始下载、下载完成后播放。对于早期的视频采集设备、网络带宽或是负责渲染的显示器而言，似乎多给予一点耐心、多重连几次断开的HTTP连接、甚至多校验几次下载后文件的完整性，体验上也还能过得去。毕竟那时候的分辨率、帧率、带宽限制了互联网途径传播媒体文件的大小，信息的分享只能通过各种硬盘、U盘、光盘以存储后文件的形式进行传输。

随着硬件设备技术的发展，采集设备分辨率在提升，显示器支持了更高的帧率，网络带宽也指数增长，这都为更好的观影体验提供了基础支持。随着网络资源的日益丰富，用户时间的稀缺性日益凸显，为了快速观看、判别视讯本身是否符合自身口味，在线实时观看成了一大诉求。而传统的HTTP下载显然不能够匹配该需求，因此在寻求streaming的道路上，RTSP脱颖而出。

RTSP全称实时流协议（Real Time Streaming Protocol），它是一个网络控制协议，设计用于娱乐、会议系统中控制流媒体服务器。RTSP用于在希望通讯的两端建立并控制媒体会话（session），客户端通过发出VCR-style命令如play、record和pause等来实时控制媒体流。可以参考RTSP 2326 中文版

什么是rtsp

> RTSP协议以客户服务器方式工作，，如：暂停/继续、后退、前进等。它是一个多媒体播放控制协议，用来使用户在播放从因特网下载的实时数据时能够进行控制, 因此 RTSP 又称为“因特网录像机遥控协议”。

RTSP(Real-Time Stream Protocol)是一种基于文本的应用层协议，在语法及一些消息参数等方面，RTSP协议与HTTP协议类似。是TCP/IP协议体系中的一个应用层协议, 由哥伦比亚大学, 网景和RealNetworks公司提交的IETF RFC标准.

该协议定义了一对多应用程序如何有效地通过IP网络传送多媒体数据. RTSP在体系结构上位于RTP和RTCP之上，它使用TCP或RTP完成数据传输. RTSP被用于建立的控制媒体流的传输，它为多媒体服务扮演“网络远程控制”的角色。尽管有时可以把RTSP控制信息和媒体数据流交织在一起传送，但一般情况RTSP本身并不用于转送媒体流数据。媒体数据的传送可通过RTP/RTCP等协议来完成。

该协议用于C/S模型, 是一个基于文本的协议, 用于在客户端和服务器端建立和协商实时流会话.

网络体系

RTSP是类似http的应用层协议，一个典型的流媒体框架网络体系可参考下图

重点概念讲解

- RTSP（Real Time Streaming Protocol），RFC2326，实时流传输协议，是TCP/IP协议体系中的一个应用层协议，由哥伦比亚大学、网景和RealNetworks公司提交的IETF RFC标准。该协议定义了一对多应用程序如何有效地通过IP网络传送多媒体数据。RTSP在体系结构上位于RTP和RTCP之上，它使用TCP或UDP完成数据传输。
- Real-time Transport Protocol或简写RTP，它是由IETF的多媒体传输工作小组1996年在RFC 1889中公布的。RTP协议详细说明了在互联网上传递音频和视频的标准数据包格式。它是创建在UDP协议上的。
- Real-time Transport Control Protocol或RTP Control Protocol或简写RTCP）是实时传输协议（RTP）的一个姐妹协议。RTCP由RFC 3550定义（取代作废的RFC 1889）。RTP 使用一个 偶数 UDP port ； 而RTCP 则使用 RTP 的下一个 port，也就是一个奇数 port。RTCP与RTP联合工作，RTP实施实际数据的传输，RTCP则负责将控制包送至电话中的每个人。其主要功能是就RTP正在提供的服务质量做出反馈。

- RTSP协议：负责服务器与客户端之间的请求与响应
- RTP协议：负责传输媒体数据
- RTCP协议：在RTP传输过程中提供传输信息

rtsp承载与rtp和rtcp之上，rtsp并不会发送媒体数据，而是使用rtp协议传输

rtp并没有规定发送方式，可以选择udp发送或者tcp发送

企鹅群994289133领取资料

rtp协议详解

背景知识

流（Streaming）是近年在Internet上出现的新概念，其定义非常广泛，主要是指通过网络传输多媒体数据的技术总称。

流式传输分为两种

- 顺序流式传输 (Progressive Streaming)
- 实时流式传输 (Real time Streaming)

实时流式传输是实时传送，特别适合现场事件。“实时”（real time）是指在一个应用中数据的交付必须与数据的产生保持精确的时间关系，这需要相应的协议支持，这样RTP和RTCP就相应的出现了

rtp协议原理

较简单，负责对流媒体数据进行封包并实现媒体流的实时传输，即它按照RPT数据包格式来封装流媒体数据，并利用与它绑定的协议进行数据包的传输。

RTP在端口号1025到65535之间选择一个未使用的偶数UDP端口号，而在同一次会话中的RTCP则使用下一个基数UDP端口号。RTP默认端口号5004，所以RTCP端口号默认为5005。

从下图可看出RTP被划分在传输层，它建立在UDP上。同UDP协议一样，为了实现其实时传输功能，RTP也有固定的封装形式。RTP用来为端到端的实时传输提供时间信息和流同步，但并不保证服务质量。服务质量由RTCP来提供。

rtp协议封装

详细讲解:

- 填充位（1bit）若p=1则在该报文的尾部填充一个或多个额外的八位组，它们不是有效载荷的一部分。填充可能用于某些具有固定长度的加密算法或者用在底层数据单元中传输多个RTP包
- 扩展（X）：1个比特，置“1”表示RTP报头后紧随一个扩展报头
- 参与源数（CSRC计数（CC））4位，CSRC计数包括紧接在固定头后CSRC标识符个数。
- 标记（M）：1个比特，其具体解释由应用文档来定义。例如，对于视频流，它表示一帧的结束，而对于音频，则表示一次谈话的开始
- 有效载荷类型，7个比特，它指示在用户数据字段中承载数据的载荷类别，记录后面资料使用哪种编码，接收端找出相应的 decoder 解码出来
- 序列号16比特 每发送一个RTP数据包,序列号加一,接收机可以据此检测包损和重建包序列.序列号的初始值是随机的(不可预测),以使即便在源本身不加密时(有时包要通过翻译器,它会这样做),对加密算法泛知的普通文本攻击也会更加困难
- 时间戳，32位，时标反映RTP数据包中第一个八进制数的采样时刻，采样时刻必须从单调、线性增加的时钟导出，以允许同步与抖动计算。时标可以让receiver端知道在正确的时间将资料播放出来。只有系列号而没有时标，并不能完整按照顺序的将data播放出来，因为如果data中间有一段是没有资料的，只有系列号的话会造成错误.
- SSRC，32位，SSRC段标识同步源。此标识不是随机选择的，目的在于使同一RTP包连接中没有两个同步源有相同的SSRC标识。尽管多个源选择同一个标识的概率很低，所有RTP实现都必须探测并解决冲突。
- CSRC列表，0到15项，每项32位。CSRC列表表示包内的对载荷起作用的源。标识数量由CC段给出。如超出15个作用源，也仅标识15个。。

rtcp协议详解

RTCP也是用UDP来传送的，但RTCP封装的仅仅是一些控制信息，因而分组很短，所以可以将多个RTCP分组封装在一个UDP包中。RTCP有如下五种分组类型。

rtsp协议详解

RTSP(Real-Time Stream Protocol)协议是一个基于文本的多媒体播放控制协议，属于应用层。RTSP以客户端方式工作，对流媒体提供播放、暂停、后退、前进等操作。该标准由IETF指定，对应的协议是RFC2326。RTSP作为一个应用层协议，提供了一个可供扩展的框架，使得流媒体的受控和点播变得可能，它主要用来控制具有实时特性的数据的发送，但其本身并不用于传送流媒体数据，而必须依赖下层传输协议(如RTP/RTCP)所提供的服务来完成流媒体数据的传送。RTSP负责定义具体的控制信息、操作方法、状态码，以及描述与RTP之间的交互操作。RTSP媒体服务协议框架如下：

所以从上述架构图中可以看出，RTSP和RTP，RTCP配合使用。RTSP传输的一般是TS、MP4格式的流，其传输一般需要2~3个通道，命令和数据通道分离。使用RTSP协议传输流媒体数据需要有专门的媒体播放器和媒体服务器，也就是需要支持RTSP协议的客户端和服务器。

一次基本的RTSP操作过程:

- 首先，客户端连接到流服务器并发送一个RTSP描述命令（DESCRIBE）。
- 流服务器通过一个SDP描述来进行反馈，反馈信息包括流数量、媒体类型等信息。
- 客户端再分析该SDP描述，并为会话中的每一个流发送一个RTSP建立命令(SETUP)，RTSP建立命令告诉服务器客户端用于接收媒体数据的端口。流媒体连接建立完成后，
- 客户端发送一个播放命令(PLAY)，服务器就开始在UDP上传送媒体流（RTP包）到客户端。在播放过程中客户端还可以向服务器发送命令来控制快进、快退和暂停等。
- 最后，客户端可发送一个终止命令(TERADOWN)来结束流媒体会话

sequenceDiagram:

```
客户端->>服务器:DESCRIBE
服务器->>客户端: 200 OK (SDP)
客户端->>服务器:SETUP
服务器->>客户端: 200 OK
客户端->>服务器:PLAY
服务器->>客户端: (RTP包)
```

协议特点

- 可扩展性: 新方法和参数很容易加入RTSP.
- 易解析: RTSP可由标准HTTP或MIME解析器解析.
- 安全: RTSP使用网页安全机制.
- 独立于传输: RTSP可使用不可靠数据报协议(EDP), 可靠数据报协议(RDP); 如要实现应用级可靠, 可使用可靠流协议.
- 多服务器支持: 每个流可放在不同服务器上, 用户端自动与不同服务器建立几个并发控制连接, 媒体同步在传输层执行.
- 记录设备控制: 协议可控制记录和回放设备.
- 流控与会议开始分离: 仅要求会议初始化协议提供, 或可用来创建惟一会议标识号. 特殊情况下, 可用SIP或H.323来邀请服务器入会.
- 适合专业应用: 通过SMPTE时标, RTSP支持帧级精度, 允许远程数字编辑.
- 演示描述中立: 协议没强加特殊演示或元文件, 可传送所用格式类型; 然而, 演示描述至少必须包括一个RTSP URL.
- 代理与防火墙友好: 协议可由应用和传输层防火墙处理. 防火墙需要理解SETUP方法, 为UDP媒体流打开一个“缺口”.
- HTTP友好: 此处, RTSP明智地采用HTTP观念, 使现在结构都可重用. 结构包括Internet内容选择平台(PICS). 由于在大多数情况下控制连续媒体需要服务器状态, RTSP不仅仅向HTFP添加方法.
- 适当的服务器控制: 如用户启动一个流, 必须也可以停止一个流.
- 传输协调: 实际处理连续媒体流前, 用户可协调传输方法.
- 性能协调: 如基本特征无效, 必须有一些清理机制让用户决定哪种方法没生效. 这允许用户提出适合的用户界面.

RTSP协议与HTTP协议区别

- RTSP引入了几种新的方法，比如DESCRIBE、PLAY、SETUP 等，并且有不同的协议标识符，RTSP为rtsp 1.0,HTTP为http 1.1；
- HTTP是无状态的协议，而RTSP为每个会话保持状态；
- RTSP协议的客户端和服务器端都可以发送Request请求，而在HTTP协议中，只有客户端能发送Request请求。
- 在RTSP协议中，载荷数据一般是通过带外方式来传送的(除了交织的情况)，及通过RTP协议在不同的通道中来传送载荷数据。而HTTP协议的载荷数据都是通过带内方式传送的，比如请求的网页数据是在回应的消息体中携带的。
- 使用ISO 10646(UTF-8) 而不是ISO 8859-1，以配合当前HTML的国际化；
- RTSP使用URI请求时包含绝对URI。而由于历史原因造成的向后兼容性问题，HTTP/1.1只在请求中包含绝对路径，把主机名放入单独的标题域中；

rtsp报文协议

rtsp数据格式

RTSP协议格式与HTTP协议格式类似

RTSP客户端的请求格式

```
method url vesion\r\n
CSeq: x\r\n
xxx\r\n
...
\r\n
```

- method：方法，表明这次请求的方法，rtsp定义了很多方法，稍后介绍
- url：格式一般为rtsp://ip:port/session，ip表主机ip，port表端口好，如果不写那么就是默认端口，rtsp的默认端口为554，session表明请求哪一个会话
- version：表示rtsp的版本，现在为RTSP/1.0
- CSeq：序列号，每个RTSP请求和响应都对应一个序列号，序列号是递增的

RTSP服务端的响应格式

```
vesion 200 OK\r\n
CSeq: x\r\n
xxx\r\n
...
\r\n
```

- version：表示rtsp的版本，现在为RTSP/1.0
- CSeq：序列号，这个必须与对应请求的序列号相同

RTSP URL的语法结构

一个终端用户是通过在播放器中输入URL地址开始进行观看流媒体业务的第一步，而对于使用RTSP协议的移动流媒体点播而言，URL的一般写法如下：

一个以“rtsp”或是“rtspu”开始的URL链接用于指定当前使用的是RTSP 协议。RTSP URL的语法结构如下：

```
rtsp_url = ("rtsp:" | "rtspu:" | "rtsps:") "//" host ["":"port"] /[abs_path]/content_n
```

- rtsp:使用可信的底层传输协议，例如TCP
- rtspu:使用不可信的底层传输协议，例如UDP
- rtsps:使用可信加密传输协议，例如TCP + TLS
- host：可以是一个有效的域名或是IP地址。
- port：端口号，对于RTSP协议来说，缺省的端口号为554。当我们在确认流媒体服务器提供的端口号为554时，此项可以省略 说明：当HMS服务器使用的端口号为554时，我们在写点播链接时，可以不用写明端口号，但当使用非554端口时，在RTSP URL中一定要指定相应的端口。
- abs_path: 为RTSPServer中的媒体流资源标识
- RTSPURL用来标识RTSPServer的媒体流资源，可以标识单一的媒体流资源，也可以标 识多个媒体流资源的集合。

例如，一个完整的RTSP URL可写为：

```
rtsp://192.168.1.67:554/test
```

又如目前市面上常用的海康网络摄像头的RTSP地址格式为：

```
rtsp://[username]:[password]@[ip]:[port]/[codec]/[channel]/[subtype]/av_stream
```

示例

```
rtsp://admin:12345@192.168.1.67:554/h264/ch1/main/av_stream
```

另一个简单的示例如下：

```
rtsp://media.example.com:554/twister/audiotrack
```

让我们来看一下上面URL的abs path = twister/audiotrack。twister表示一个标识(Presentation)，标识(Presentation)由一个或多个实时流组成。audiotrack表示标识(Presentation)中其中一个实时流的名称。从这个名称可以看出，我们要取的是一个音频流。如果abs path = twister/videotrack，则表示我们要取的是twister的视频流。

有的服务器也支持下面的URL形式：

```
rtsp://media.example.com:554/twister
```

该URL表示取标识(Presentation)的视频流和音频流。

RTSP的报文结构

RTSP是一种基于文本的协议，用CRLF作为一行的结束符。使用基于文本协议的好处在于我们可以随时在使用过程中的增加自定义的参数，也可以随便将协议包抓住很直观的进行分析。

RTSP有两类报文：请求报文和响应报文。请求报文是指从客户向服务器发送请求报文，响应报文是指从服务器到客户的回答。由于 RTSP 是面向正文的(text-oriented)，因此在报文中的每一个字段都是一些 ASCII 码串，因而每个字段的长度都是不确定的。RTSP报文由三部分组成，即开始行、首部行和实体主体。在请求报文中，开始行就是请求行.

RTSP请求报文

RTSP请求报文的方法包括：OPTIONS、DESCRIBE、SETUP、TEARDOWN、PLAY、PAUSE、GET_PARAMETER和SET_PARAMETER。

一个请求消息（a request message）即可以由客户端向服务端发起也可以由服务端向客户端发起。请求消息的语法结构如下：

```
Request = Request-Line

  *(  general-header   | request-header | entity-header)

      CRLF

      [message-body]
```

Request Line

请求消息的第一行的语法结构如下：

```
Request-Line    =   Method 空格 Request-URI 空格 RTSP-Version CRLF
```

其中在消息行中出现的第一个单词即是所使用的信令标志。目前已经有的信息标志如下：

```
Method      =   "DESCRIBE"
          |   "ANNOUNCE"
```

```
          |   "GET_PARAMETER"
          |   "OPTIONS"
          |   "PAUSE"
          |   "PLAY"
          |   "RECORD"
          |   "REDIRECT"
          |   "SETUP"
          |   "SET_PARAMETER"
          |   "TEARDOWN"
```

例子：

```
DESCRIBE rtsp://211.94.164.227/3.3gp RTSP/1.0
```

Request Header Fields

在消息头中除了第一行的内容外，还有一些需求提供附加信息。其中有些是一定要的，后续我们会详细介绍经常用到的几个域的含义。

```
Request-header      =   Accept
              |   Accept-Encoding
              |   Accept-Language
              |   Authorization
              |   From
              |   If-Modified-Since
              |   Range
              |   Referer
              |   User-Agent
```

响应消息

响应报文的开始行是状态行，RTSP响应报文的结构如下图所示

响应消息的语法结构如下：

```
Response = Status-Line *( general-header | response-header | entity-header) CRLF [mes
```

Status-Line

响应消息的第一行是状态行（status-line），每个元素之间用空格分开。除了最后的CRLF之外，在此行的中间不得有CR或是LF的出现。它的语法格式如下，

```
Status-Line = RTSP-Version 空格Status-Code 空格Reason-Phrase CRLF
```

状态码（Status-Code） 是一个三位数的整数，用于描述接收方对所收到请求消息的执行结果

Status-Code的第一位数字指定了这个回复消息的种类，一共有5类：

- 1XX: Informational – 请求被接收到，继续处理
- 2XX: Success – 请求被成功的接收，解析并接受
- 3XX: Redirection – 为完成请求需要更多的操作

- 4XX: Client Error – 请求消息中包含语法错误或是不能够被有效执行
- 5XX: Server Error – 服务器响应失败，无法处理正确的有效的请求消息

Response Header Fields

在响应消息的域中存放的是无法放在Status-Line中,而又需要传送给请求者的一些附加信息。

```
Response-header    =   Location
                   |   Proxy-Authenticate
                   |   Public
                   |   Retry-After
                   |   Server
                   |   Vary
                   |   WWW-Authenticate
```

RTSP的主要方法

注：P---演示，C---客户端,S---服务器， S（对象栏）---流

RTSP重要头字段参数

- Accept: 用于指定客户端可以接受的媒体描述信息类型。比如: Accept: application/rtsl, application/sdp;level=2
- Bandwidth: 用于描述客户端可用的带宽值。
- CSeq：指定了RTSP请求回应对的序列号，在每个请求或回应中都必须包括这个头字段。对每个包含一个给定序列号的请求消息，都会有一个相同序列号的回应消息。
- Rang：用于指定一个时间范围，可以使用SMPTE、NTP或clock时间单元。
- Session: Session头字段标识了一个RTSP会话。Session ID 是由服务器在SETUP的回应中选择的，客户端一当得到Session ID后，在以后的对Session 的操作请求消息中都要包含Session ID.
- Transport: Transport头字段包含客户端可以接受的转输选项列表，包括传输协议，地址端口，TTL等。服务器端也通过这个头字段返回实际选择的具体选项。如: Transport: RTP/AVP;multicast;ttl=127;mode="PLAY", RTP/AVP;unicast;client_port=3456-3457;mode="PLAY"

简单的RTSP消息交互过程

> C表示RTSP客户端,S表示RTSP服务端

第一步：查询服务器端可用方法

```
C->S OPTION request //询问S有哪些方法可用

S->C OPTION response //S回应信息的public头字段中包括提供的所有可用方法
```

第二步：得到媒体描述信息

```
C->S DESCRIBE request //要求得到S提供的媒体描述信息

S->C DESCRIBE response //S回应媒体描述信息，一般是sdp信息
```

第三步：建立RTSP会话

```
C->S SETUP request //通过Transport头字段列出可接受的传输选项，请求S建立会话

S->C SETUP response //S建立会话，通过Transport头字段返回选择的具体转输选项，并返回建立的Sessi
```

第四步：请求开始传送数据

C->S PLAY request //C请求S开始发送数据

S->C PLAY response //S回应该请求的信息

第五步：数据传送播放中

S->C 发送流媒体数据 // 通过RTP协议传送数据

第六步：关闭会话，退出

C->S EARDOWN request //C请求关闭会话

S->C TEARDOWN response //S回应该请求

上述的过程只是标准的、友好的rtsp流程，但实际的需求中并不一定按此过程。其中第三和第四步是必需的！第一步，只要服务器和客户端约定好有哪些方法可用，则option请求可以不要。第二步，如果我们有其他途径得到媒体初始化描述信息（比如http请求等等），则我们也不需要通过rtsp中的describe请求来完成。

RTSP的请求响应示例

> 其中C是客户端，S是服务端。

OPTIONS

- C–>S

```
OPTIONS rtsp://192.168.31.115:8554/live RTSP/1.0\r\n
CSeq: 2\r\n
\r\n
```

客户端向服务器请求可用方法

- S–>C

```
RTSP/1.0 200 OK\r\n
CSeq: 2\r\n
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY\r\n
\r\n
```

服务端回复客户端，当前可用方法OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY

DESCRIBE

- C–>S

```
DESCRIBE rtsp://192.168.31.115:8554/live RTSP/1.0\r\n
CSeq: 3\r\n
Accept: application/sdp\r\n
\r\n
```

客户端向服务器请求媒体描述文件，格式为sdp

- S–>C

```
RTSP/1.0 200 OK\r\n
CSeq: 3\r\n
Content-length: 146\r\n
Content-type: application/sdp\r\n
\r\n

v=0\r\n
o=- 91565340853 1 in IP4 192.168.31.115\r\n
t=0 0\r\n
a=contol:*\r\n
m=video 0 RTP/AVP 96\r\n
a=rtpmap:96 H264/90000\r\n
a=framerate:25\r\n
a=control:track0\r\n
```

服务器回复了sdp文件，这个文件告诉客户端当前服务器有哪些音视频流，有什么属性，具体稍后再讲解

这里只需要直到客户端可以根据这些信息得知有哪些音视频流可以发送

SETUP

1. C–>S

```
SETUP rtsp://192.168.31.115:8554/live/track0 RTSP/1.0\r\n
CSeq: 4\r\n
Transport: RTP/AVP;unicast;client_port=54492-54493\r\n
\r\n
```

客户端发送建立请求，请求建立连接会话，准备接收音视频数据

解析一下Transport: RTP/AVP;unicast;client_port=54492-54493\r\n

- RTP/AVP：表示RTP通过UDP发送，如果是RTP/AVP/TCP则表示RTP通过TCP发送
- unicast：表示单播，如果是multicast则表示多播
- client_port=54492-54493：由于这里希望采用的是RTP OVER UDP，所以客户端发送了两个用于传输数据的端口，客户端已经将这两个端口绑定到两个udp套接字上，54492表示是RTP端口，54493表示RTCP端口(RTP端口为某个偶数，RTCP端口为RTP端口+1)

2.S–>C

```
RTSP/1.0 200 OK\r\n
CSeq: 4\r\n
Transport: RTP/AVP;unicast;client_port=54492-54493;server_port=56400-56401\r\n
Session: 66334873\r\n
\r\n
```

- 服务端接收到请求之后，得知客户端要求采用RTP OVER UDP发送数据，单播，客户端用于传输RTP数据的端口为54492，RTCP的端口为54493
- 服务器也有两个udp套接字，绑定好两个端口，一个用于传输RTP，一个用于传输RTCP，这里的端口号为56400-56401
- 之后客户端会使用54492-54493这两端口和服务器通过udp传输数据，服务器会使用56400-56401这两端口和这个客户端传输数据

PLAY

- C–>S

```
PLAY rtsp://192.168.31.115:8554/live RTSP/1.0\r\n
CSeq: 5\r\n
Session: 66334873\r\n
Range: npt=0.000-\r\n
\r\n
```

客户端请求播放媒体

- S–>C

```
RTSP/1.0 200 OK\r\n
```

```
CSeq: 5\r\n
Range: npt=0.000-\r\n
Session: 66334873; timeout=60\r\n
\r\n
```

服务器回复之后，会开始使用RTP通过udp向客户端的54492端口发送数据

TEARDOWN

- C–>S

```
TEARDOWN rtsp://192.168.31.115:8554/live RTSP/1.0\r\n
CSeq: 6\r\n
Session: 66334873\r\n
\r\n
```

- S–>C

```
RTSP/1.0 200 OK\r\n
CSeq: 6\r\n
\r\n
```

编辑于 2022-03-10 16:41

流媒体 RTSP 网络传输协议

写下你的评论...

5 条评论
默认
最新

尘封
博主了解interleaved模式吗？可以写一下吗？
2022-11-11
赞

最是浮沫化光阴、
博主请问下RTSP协议下的scale方法怎么用阿？
2022-08-23
赞

小霸王打天下
请问下，我们公司安装了海康的摄像头，通过RTSP协议可以读取到视频流实时数据，然后进行图像分析；现在的希望是每秒钟抽取两帧进行图像分析，减小系统的负担，请问如何操作
2022-07-12
赞

zuokanyunqi
博主 请问您的rtsp协议如何读取到实时数据流
2022-08-14
赞

王俊
楼主写的浅显易懂，棒棒的。
2022-04-28
赞

推荐阅读

音视频流媒体服务器开发：低 音视频开发技术之同步原理详

延迟流媒体协议SRT、...

低广播延迟已经成为任何关于建设源端站和 CDN 的招标和竞争中的必要特性。以前这种标准只适用于体育广播，但现在运营商要求每个领域的广播设备供应商提供低延迟，比如：广播新闻、音乐会、...

Linux百里

音视频通信协议--RTSP

Fenng... 发表于Media...

解及实现

1.音视频同步简单介绍对于一个播放器，一般来说，其基本构成均可划分为以下几部分：数据接收（网络/本地）->解复用->音视频解码->音视频同步->音视频输出。基本框架如下图所示...

linux 发表于linux...

音视频协议--NACK系列一

Fenng... 发表于Media...