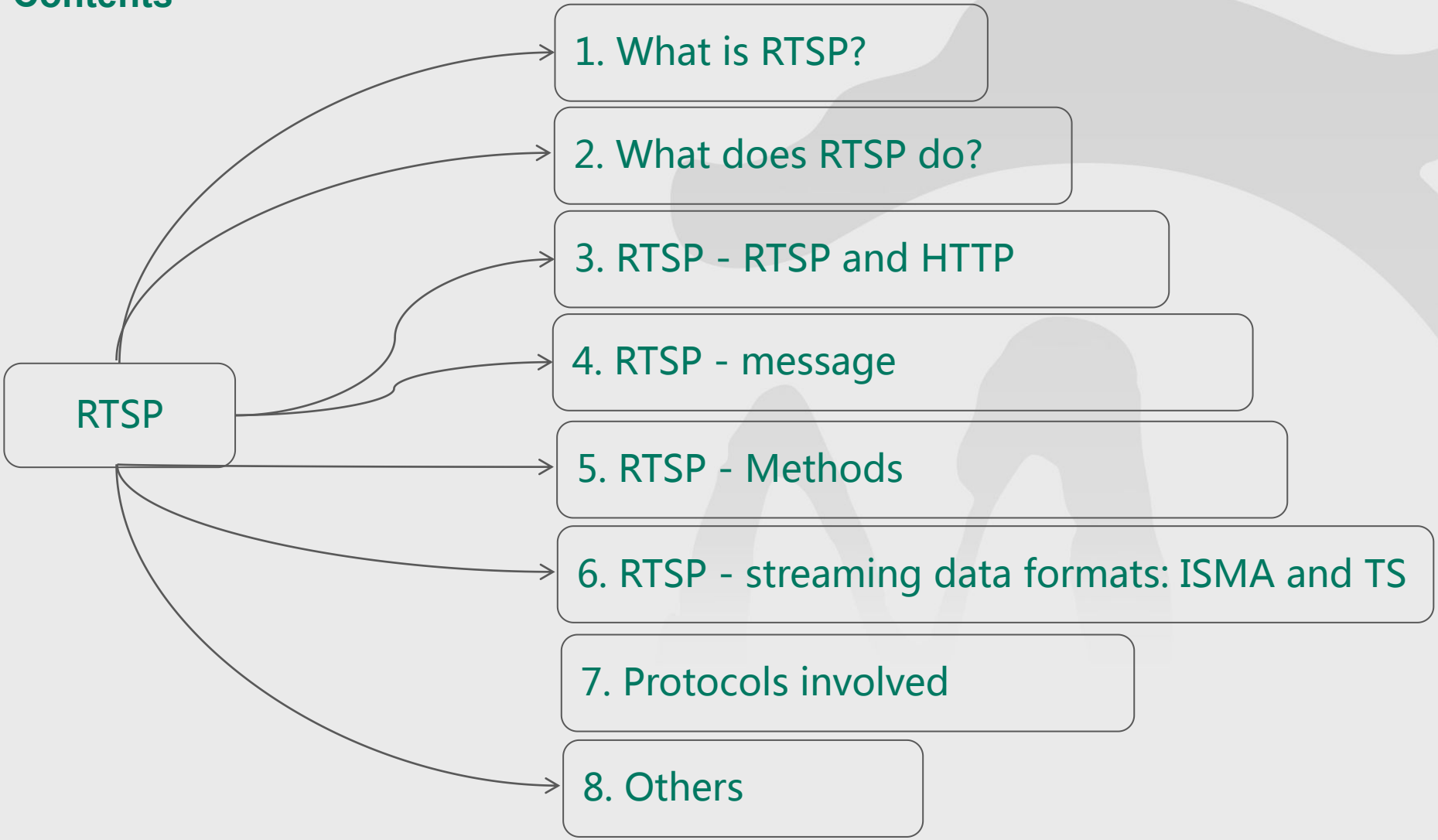
RTSP

Introduction

www.movision.com.cn

常见视频相关通信协议介绍

Contents


RTSP
1. What is RTSP?
2. What does RTSP do?
3. RTSP - RTSP and HTTP
4. RTSP - message
5. RTSP - Methods
6. RTSP - streaming data formats: ISMA and TS
7. Protocols involved
8. Others

Foreword

Before attending this course, you should know:

1. Heard of RTSP, have a basic knowledge of HTTP
2. RTSP is closely related to video surveillance
3. Precondition
   Presumed that you have some knowledge of SDP
4. Targets
   - 协议组相关同仁
   - 视频会议相关研发
   - 视讯产线相关的测试
   - 其他对此感兴趣的同仁

Goal

After attending this course, you will know:

1. 了解RTSP流媒体协议；
2. 了解各种常见的RTP/RTCP及其相关的信令和流程；
3. 掌握常见的RTSP相关的故障的定位分析思路；

**SDP部分内容由于跟SIP重叠，因此暂不在此详细说明*

1. What is RTSP?

RTSP - Real Time Streaming Protocol

RFC2326

Submitted by Columbia U., Netscape and RealNetworks

2. What does RTSP do?

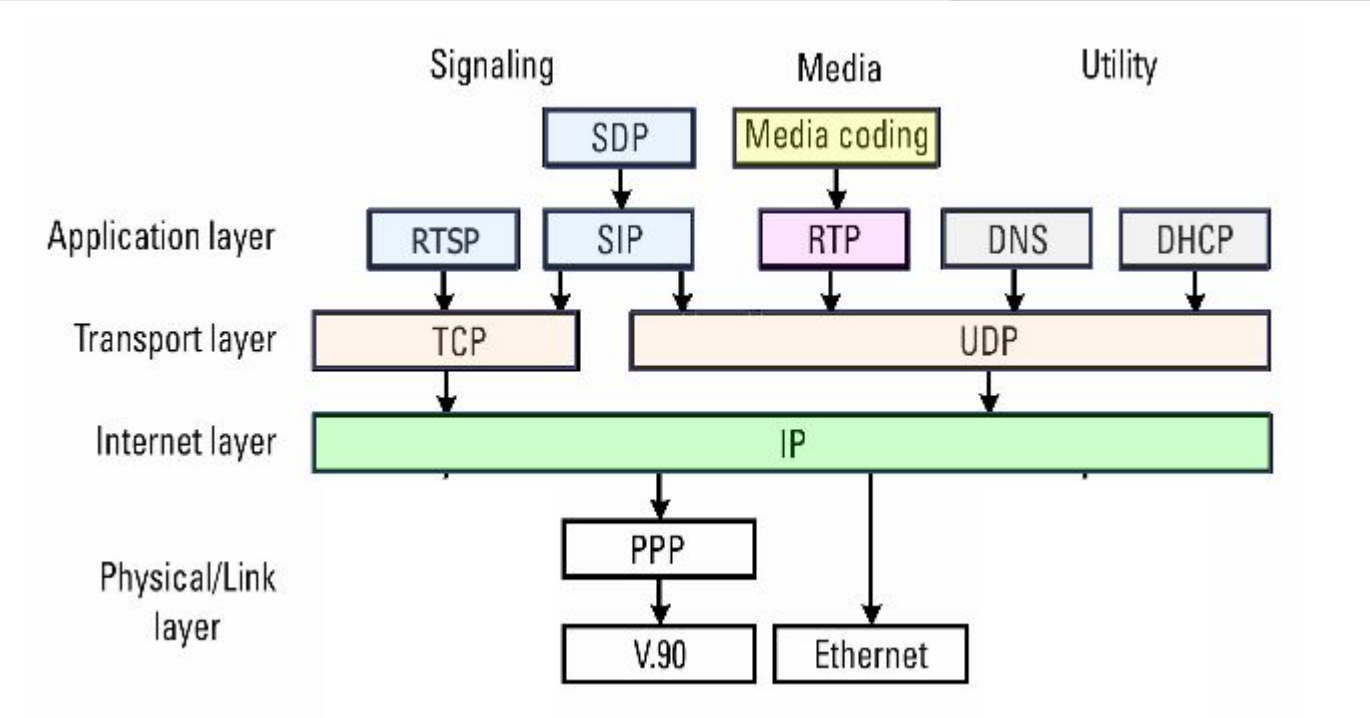
The Real-Time Streaming Protocol (RTSP) establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video.

The protocol supports the following operations:

* Retrieval of media from media server
* Invitation of a media server to a conference
* Addition of media to an existing presentation

RTSP requests may be handled by proxies, tunnels and caches as in HTTP/1.1.

RTSP: An Application layer protocol



- *Data transfer:*
  - *UDP (RTP/AVP)*
  - *TCP (RTP/AVP/TCP) reuse signaling TCP socket*
  - *HTTP(Refer to ONVIF protocol for more details)*

3. RTSP - RTSP v.s. HTTP

- NEW Methods
  - RTSP introduces a number of new methods and has a different protocol identifier.
- Stateful
  - An RTSP server needs to maintain state, SETUP, PLAY, RECORD, PAUSE, and TEARDOWN, by default in almost all cases, as opposed to the stateless nature of HTTP.
- Requests trigger
  - Both an RTSP server and client can issue requests.
- Data transmit
  - Data is carried out-of-band by a different protocol. (There is an exception to this.)
- Charset
  - RTSP is defined to use ISO 10646 (UTF-8) rather than ISO 8859-1, consistent with current HTML internationalization efforts [3].
- Request-URI
  - The Request-URI always contains the absolute URI. Because of backward compatibility with a historical blunder, HTTP/1.1 [2] carries only the absolute path in the request and puts the host name in a separate header field.

	HTTP	RTSP
Session	Connection based	Session based
Tranport	TCP	TCP or UDP
Data transmit	in-band	out-of-band or in-band
Server states	Stateless	Statefull
Methods	GET, POST	OPTION, DESCRIBE, SETUP, PLAY.
Req trigger	Client	Client or Server
Charset	Latin-1	UTF-8
Chunk	Support	Not support
URI	relativeURI/absoluteURI + host	relativeURI/absoluteURI

4. RTSP - message

- RTSP is a text-based protocol and uses the ISO 10646 character set in UTF-8 encoding (RFC 2279 [21]). (Diff to HTTP, Similar to SIP)
- Lines are terminated by CRLF, but receivers should be prepared to also interpret CR and LF by themselves as line terminators. (Similar to HTTP and SIP)
- The 10646 character set avoids tricky character set switching, but is invisible to the application as long as US-ASCII is being used. (Same with RTCP)

4. RTSP - message type

- RTSP messages consist of requests from client to server and responses from server to client.

  RTSP -message = Request | Response ; RTSP /1.0 messages

- 请求（Request）和回应（Response）消息都使用RFC822中实体传输部分规定（作为消息中的有效载荷）的消息格式。两者的消息都可能包括一起始行，一个或多个标题域（headers）、一行表示标题域结束的空行（即CRLF前没有内容的行），和一个消息主体（message-body,可选）。

  generic-message = start-line

  *message-header

  CRLF

  [message-body]

  start-line = Request-Line | Status-Line

- In the interest of robustness, servers SHOULD ignore any empty line(s) received where a Request-Line is expected. In other words, if the server is reading the protocol stream at the beginning of a message and receives a CRLF first, it should ignore the CRLF.

4. RTSP - message header

- RTSP标题域，包括主标题（General-Header,4.3节）、请求标题、回应标题（及实体标题，都遵照RFC822-3.1
- 节[7]给出的通用格式定义。每个标题域由后紧跟冒号的名字，单空格（SP），字符及域值组成。域名是大小写敏感的。虽然不提倡，标题域还是可以扩展成多行使用，只要这些行以一个以上的SP或HT开头就行。
- RTSP-header = field-name ":" [field-value] CRLF
- field-name = token
- field-value = *(field-content | LWS)
- field-content = <the OCTETs make up the field-value
- and consisting of either *TEXT or combinations
- of token, tspecials, and quoted-string>
- 标题域接收的顺序并不重要，但良好的习惯是，先发送主标题，然后是请求标题或回应标题，最后是实体标题。
- 当且仅当标题域的全部域值都用逗号分隔的列表示时（即，#（值）），多个有相同域名的RTSP标题域才可以表示在一个消息里。而且必须能在不改变消息语法的前提下，将并发的域值加到第一个值后面，之间用逗号分隔，最终能将多个标题域结合成“域名：域值”对。

4. RTSP - message body

- RTSP消息的消息主体（如果有）用来携带请求或回应的主体。仅在使用传输编码(Transfer-Encoding)时消息主体和实体主体才有所不同，这种情况在传输编码标题域中有详细说明。
- message-body = entity-body | <entity-body encoded as per Transfer-Encoding>

- 传输编码必须能解释所有保证传输安全和正确的应用程序的传输编码。传输编码是消息而不是实体的一个属性，因此可以由任一应用程序随着请求/回应链添加或者删除。

- 什么时候允许消息带消息体的规则在请求和回应两种情况下有所不同。
- 在请求中有无消息主体的标志是是否包含内容长度或请求消息标题域中的传输编码标题域。只有当请求方法允许有实体主体的时候才能在请求中包含消息主体。
- 而对于回应消息来说，无论消息中是否存在消息主体都与请求方法和回应状态编码无关。所有回应标题请求方法的消息都不能包含消息主体，尽管有时会因为存在实体标题域而使人产生误解。所有1××（信息），204（无内容），304（未修改）回应都不包含消息主体。而其他回应则都包含主体，尽管其长度有可能长度为零。

- 4.4 消息长度
- 当消息包含消息主体时，消息主体的长度由以下规则来决定（按优先级高低顺序排列）：
  - 1. 任何回应消息都不包含消息主体（如1××，204和304回应），并且不管消息中是否存在实体标题域都以消息标题域后的第一行空行表示结束。
  - 2. 如果内容长度标题域存在，它在字节中的值就是消息主体的长度。如果内容标题域不存在，则假设值为零。
  - 3. 服务器关闭连接时。（关闭连接没有用来表明请求主体结束，否则可能导致服务器不能回应。

- 注意，RTSP不支持（至少现在）HTTP/1.1的块传输编码（详见[H3.6]）并且要求有内容长度标题域。
- 尽管表示描述长度动态产生，但由于可获得了表示描述返回长度，使得服务器总是能决定表示描述长度而不需使用块传输编码方式。只要有实体主体就必须有内容长度项，这些规则保证了即使没有给出明确长度也能做出合理的操作。

附: HTTP/1.1的块传输编码

3.6.1 Chunked Transfer Coding

The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator, followed by an OPTIONAL trailer containing entity-header fields. This allows dynamically produced content to be transferred along with the information necessary for the recipient to verify that it has received the full message.

```
Chunked-Body   = *chunk
                 last-chunk
                 trailer
                 CRLF
chunk          = chunk-size [ chunk-extension ] CRLF
                 chunk-data CRLF
chunk-size     = 1*HEX
last-chunk     = 1*("0") [ chunk-extension ] CRLF

chunk-extension= *( ";" chunk-ext-name [ "=" chunk-ext-val ] )
chunk-ext-name = token
chunk-ext-val  = token | quoted-string
chunk-data     = chunk-size(OCTET)
trailer        = *(entity-header CRLF)
```

4. RTSP - Response messages

- **Response status line状态行（基本等同HTTP和SIP）：**
  完整回应消息的第一行就是状态行，它依次由协议版本、数字形式的状态代码、及相应的词语文本组成，各元素间以空格（SP）分隔，除了结尾的CRLF外，不允许出现单独的CR或LF符。
  Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
- **Response status code状态代码（基本等同HTTP和SIP）：**
  状态代码由3位数字组成，表示请求是否被理解或被满足。如：200
  原因分析是用简短的文字来描述状态代码产生的原因。如：OK
  状态代码用来支持自动操作，原因分析是为人类用户准备的。
  客户端不需要检查或显示原因分析。
  状态代码分类：
  - 1xx:：保留，将来使用。
  - 2xx：成功 — 操作被接收、理解、接受（received, understood, accepted）。
  - 3xx：重定向（Redirection）— 要完成请求必须进行进一步操作。
  - 4xx：客户端出错 — 请求有语法错误或无法实现。
  - 5xx：服务器端出错 — 服务器无法实现合法的请求。

4. RTSP - Response message header

- 回应标题域中包括不能放在状态行中的附加回应信息。该域还可以存放与服务器相关的信息，以及在对请求URI所指定资源进行访问的下一步信息。

  response-header = Location ;
  | Proxy-Authenticate ;
- | Public ;
- | Retry-After ;
  | Server ;
  | Vary ;
  | WWW-Authenticate ;

  回应标题域名只有在与协议版本的变化结合起来后，才能进行可靠的扩展。实际上，新的或实验中的标题域只要能被通讯各方识别，其语法就可使用，而无法识别的标题域都将被视为实体域。

5. RTSP - Methods

- 方法定义表示了对请求统一资源标志符（Request-URI）识别的资源所执行的操作。方法名区分大小写。将来可能定义新的方法。方法名可能不以美元符' $ '（十进制数24）开头，但必须具有表征意义。
- 对RTSP方法，和其操作方向及所操作对象（P: 表示, S: 媒体流）的一个概览
- 注意：PAUSE方法是推荐的, 但在构建一个全功能的服务器时可能不支持此方法，这时就不需要它，比如对于live feeds。如果服务器不支持某个特殊方法，它必将返回"501 Not Implemented"，并且客户端应该不再向该服务器请求该方法。

5. RTSP - Methods

Method	Direction	Targets	Implements
OPTIONS	C -> S,S ->C	P,S	required
DESCRIBE	C -> S	P,S	recommended
ANNOUNCE	C -> S,S ->C	P,S	optional
SETUP	C -> S	S	required
PAUSE	C -> S	P,S	recommended
PLAY	C -> S	P,S	required
RECORD	C -> S	P,S	optional
REDIRECT	S ->C	P,S	optional
GET PARAMETER	C -> S,S ->C	P,S	optional
SET PARAMETER	C -> S,S ->C	P,S	optional
TEARDOWN	C -> S	P,S	required

**Direction: C = Client, S = Server*
**Targets: P = Presentation, S = Stream*

5. RTSP - Methods: OPTIONS

- OPTIONS method is used to inquery server's capatibilities or supported features
- OPTIONS request could be sent at any time under any circumstances. It's not related with the current server status.

- Example:
  - C->S:

```
OPTIONS * RTSP/1.0
CSeq: 1
Require: implicit-play
Proxy-Require: gzipped-messages
```

  - S->C:

```
RTSP/1.0 200 OK
CSeq: 1
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

- *TIP: these are necessarily fictional features(这些都是必要的构造特征).*

5. RTSP - Methods: DESCRIBE

- DESCRIBE方法从服务器检索表示的描述或媒体对象，这些资源通过请求统一资源定位符（the request URL）识别。此方法可能结合使用Accept首部域来指定客户端理解的描述格式。服务器端用被请求资源的描述对客户端作出响应。DESCRIBE的答复-响应对（reply-response pair）组成了RTSP的媒体初始化阶段。
- 示例：
- C->S:

  DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
  CSeq: 2
  Accept: application/sdp, application/rtsl, application/mheg

- S->C:

  RTSP/1.0 200 OK
  CSeq: 2
  Date: 23 Jan 1997 15:35:06 GMT
  Content-Type: application/sdp
  Content-Length: 376
  v=0
  o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
  s=SDP Seminar
  i=A Seminar on the session description protocol

u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
m=whiteboard 32416 UDP WB
a=orient:portrait

- DESCRIBE响应必须包含它所描述资源的所有媒体初始化信息。如果媒体客户端从一个数据源获得表示描述，而非通过DESCRIBE，并且该描述包含了一个媒体初始化参数的全集，那么客户端就应该使用这些参数，而不是再通过RTSP请求相同媒体的描述。
- 再有，服务器不应该（SHOULD NOT）使用DESCRIBE响应作为media indirection的方法。

- 需要建立基本的规则，使得客户端有明确的方法了解何时通过DESCRIBE请求媒体初始化信息，何时不请求。强制DESCRIBE响应包含它所描述媒体流集合的所有初始化信息，不鼓励将DESCRIBE用作media indirection的方法，通过这两点避免了使用其他方法可能会引起的循环问题（looping problems）
- 媒体初始化是任何基于RTSP系统的必要条件，但RTSP规范并没有规定它必须通过DESCRIBE方法完成。RTSP客户端可以通过3种方法来接收媒体初始化信息：
- DESCRIBE方法；
- 其它一些协议（HTTP，Email附件，等）；
- 命令行或标准输入（同一个SDP或其它媒体初始化格式的文件一起启动，工作方式类似于浏览器的帮助程序）。
- 为了实际协同工作，强烈建议最精简的服务器也支持DESCRIBE方法，最精简的客户端也支持从标准输入，命令行和/或其它对于客户端操作环境合适的方法来接收媒体初始化文件的能力。

5. RTSP - Methods: ANNOUNCE

- ANNOUNCE方法有两个用途：当客户端向服务器发送时，ANNOUNCE将通过请求URL识别的表示描述或者媒体对象提交给服务器；当服务器向客户端发送时，ANNOUNCE实时更新会话描述。 如果有新的媒体流加到表示中（比如在一个现场表示中），整个表示描述应该重发；而不只是增加组件，如果这样做的话，组件也可以被删除了。
- C->S:

  ANNOUNCE rtsp://server.example.com/fizzle/foo RTSP/1.0
  CSeq: 312
  Date: 23 Jan 1997 15:35:06 GMT
  Session: 47112344
  Content-Type: application/sdp
  Content-Length: 332
  v=0
  o=mhandley 2890844526 2890845468 IN IP4 126.16.64.4
  s=SDP Seminar
  i=A Seminar on the session description protocol
  u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
  e=mjh@isi.edu (Mark Handley)
  c=IN IP4 224.2.17.12/127
  t=2873397496 2873404696
  a=recvonly
  m=audio 3456 RTP/AVP 0
  m=video 2232 RTP/AVP 31
- S->C:

  RTSP/1.0 200 OK
  CSeq: 312

5. RTSP - Methods: SETUP

- SETUP请求为URI指定流式媒体的传输机制。客户端能够发出一个SETUP请求为正在播放的媒体流改变传输参数，服务器可能同意这些参数的改变。若是不同意，它必须响应错误"455 Method Not Valid In This State"。 为了尽量绕开防火墙干涉，即使它不会影响参数，客户端也必须指出传输参数，例如，指出服务器向外发布的固定的广播地址。
- 由于SETUP包括了所有传输初始化信息，防火墙和其他中间的网络设备（它们需要这些信息）分让了解析DESCRIBE响应的繁琐任务，这些任务留给了媒体初始化。Transport首部域指定了客户端数据传输时可接受的传输参数；响应包含了由服务器选出的传输参数。
- C->S:

  SETUP rtsp://example.com/foo/bar/baz.rm RTSP/1.0
  CSeq: 302
  Transport: RTP/AVP;unicast;client_port=4588-4589

- S->C:

  RTSP/1.0 200 OK
  CSeq: 302
  Date: 23 Jan 1997 15:35:06 GMT
  Session: 47112344
  Transport: RTP/AVP;unicast; client_port=4588-4589;server_port=6256-6257

- 作为对SETUP请求的响应，服务器产生了会话标志符。如果对服务器的请求中包含了会话标志符，服务器必须将此setup请求捆绑到一个存在的会话，或者返回"459 Aggregate Operation Not Allowed"。

5. RTSP - Methods: PLAY

- PLAY方法告知服务器通过SETUP中指定的机制开始发送数据 。在尚未收到SETUP请求的成功应答之前，客户端不可以发出PLAY请求。PLAY请求将正常播放时间（normal play time）定位到指定范围的起始处，并且传输数据流直到播放范围结束。PLAY请求可能被管道化（pipelined），即放入队列中（queued）；服务器必须将PLAY请求放到队列中有序执行。也就是说，后一个PLAY请求需要等待前一个PLAY请求完成才能得到执行。
- 示例：
- C->S:

  PLAY rtsp://audio.example.com/audio RTSP/1.0
  CSeq: 835
  Session: 12345678
  Range: npt=10-15
- C->S:

  PLAY rtsp://audio.example.com/audio RTSP/1.0
  CSeq: 836
  Session: 12345678
  Range: npt=20-25
- C->S:

  PLAY rtsp://audio.example.com/audio RTSP/1.0
  CSeq: 837
  Session: 12345678
  Range: npt=30-

- 结合PAUSE请求的描述，看更深一层的示例。不含Range首部域的PLAY请求也是合法的。它从媒体流开头开始播放，直到媒体流被暂停。如果媒体流通过PAUSE暂停，媒体流传输将在暂停点（the pause point）重新开始。
- 如果媒体流正在播放，那么这样一个PLAY请求将不起更多的作用，只是客户端可以用此来测试服务器是否存活。
- Range首部域可能包含一个时间参数。该参数以UTC格式指定了播放（palayback）开始的时间。如果在这个指定时间后收到消息，那么播放立即开始。时间参数可能用来帮助同步从不同数据源获取的数据流。
- 对于一个点播（On-demand）媒体流，服务器用播放（play back）的实际范围答复请求。如果在请求中没有指定范围，当前位置将在答复中返回。答复中播放范围的单位与请求中相同。在播放完被要求的范围后，表示将自动暂停，就好像发出了一个PAUSE请求。

下面的示例在play整个表示时从SMPTE时间0:10:20直到剪辑（clip）结束。播放开始于1997年1月23号，15点36分

- C->S:

  PLAY rtsp://audio.example.com/twister.en RTSP/1.0
  CSeq: 833
  Session: 12345678
  Range: smpte=0:10:20-;time=19970123T153600Z
  S->C: RTSP/1.0 200 OK
  CSeq: 833
  Date: 23 Jan 1997 15:35:06 GMT
  Range: smpte=0:10:22-;time=19970123T153600Z

- C->S:

  PLAY rtsp://audio.example.com/meeting.en RTSP/1.0
  CSeq: 835
  Session: 12345678
  Range: clock=19961108T142300Z-19961108T143520Z
  S->C: RTSP/1.0 200 OK
  CSeq: 835
  Date: 23 Jan 1997 15:35:06 GMT

For playing back a recording of a live presentation, it may be desirable to use clock units

5. RTSP - Methods: PAUSE

- PAUSE请求引起媒体流传输的暂时中断。如果请求URL中指定了具体的媒体流，那么只有该媒体流的播放和记录被暂停（halt）。比如，指定暂停音频，播放将会无声。如果请求URL指定了一个表示或者媒体流已成组，那么在该表示或组中的所有当前活动流的传输将被暂停。在重启播放或记录后，必须维护不同媒体轨迹（track）的同步。尽管服务器可能在暂停后，在timeout的时间内关闭会话，释放资源，但是任何资源都必须保存，其中timeout参数位于SETUP消息的会话头中。
- 示例：

  C->S: PAUSE rtsp://example.com/fizzle/foo RTSP/1.0
  CSeq: 834
  Session: 12345678
  S->C: RTSP/1.0 200 OK
  CSeq: 834
  Date: 23 Jan 1997 15:35:06 GMT
- PAUSE请求中可能包含一个Range首部域用来指定何时媒体流或表示暂停，我们称这个时刻为暂停点（pause point）。该首部域必须包含一个精确的值，而不是一个时间范围。媒体流的正常播放时间设置成暂停点。当服务器遇到在任何当前挂起（pending）的PLAY请求中指定的时间点后，暂停请求生效。如果Range首部域指定了一个时间超出了任何一个当前挂起的PLAY请求，将返回错误"457 Invalid Range" 。如果一个媒体单元（比如一个音频或视频禎）正好在一个暂停点开始，那么表示将不会被播放或记录。如果Range首部域缺失，那么在收到暂停消息后媒体流传输立即中断，并且暂停点设置成当前正常播放时间。

- 利用PAUSE请求可忽视所有排队的PLAY请求，但必须维护媒体流中的暂停点。不带Range首部域的后继PLAY请求从暂停点重启播放。
- 比如，如果服务器有两个挂起的播放请求，播放范围（range）分别是10到15和20到29，这时收到一个暂停请求，暂停点是NPT21，那么它将会开始播放第二个范围，并且在NPT21处停止。如果服务器正在服务第一个请求播放到NPT13位置，收到暂停请求，暂停点NPT12，那么它将立即停止。如果请求在NPT16暂停，那么服务器在完成第一个播放请求后停止，放弃了第二个播放请求。
- 再如，服务器收到播放请求，播放范围从10到15和13到20（即之间有重叠），PAUSE暂停点是NPT14，则当服务器播放第一段范围时，PAUSE请求将生效，而第二个PLAY请求会被忽略重叠部分，就好像服务器在开始播放第二段前收到PAUSE请求。不管PAUSE请求何时到达，它总是设置NPT到14。
- 如果服务器已经在Range首部域指定的时间外发送了数据，后继的PLAY仍会在暂停点及时重启，因为它认为客户端会丢弃在暂停点后收到的数据。这就确保了连续、无隙的暂停/播放循环。

5. RTSP - Methods: TEARDOWN

- TEARDOWN请求终止了给定URI的媒体流传输，并释放了与该媒体流相关的资源。如果该URI是对此表示的表示URI，那么任何与此会话相关的任何RTSP会话标志符将不再有效。除非所有传输参数由会话描述符定义，否则SETUP请求必须在会话能被再次播放之前发出。
- 示例：
- C->S:

  TEARDOWN rtsp://example.com/fizzle/foo RTSP/1.0

  CSeq: 892

  Session: 12345678
- S->C:

  RTSP/1.0 200 OK

  CSeq: 892

5. RTSP - Methods: GET_PARAMETER

- GET PARAMETER请求检索URI指定的表示或媒体流的参数值。答复和响应的内容留给了实现。不带实体主体的GET PARAMETER可用来测试客户端或服务器是否存活（"Ping"）。
- 示例：
- S->C:

  GET_PARAMETER rtsp://example.com/fizzle/foo RTSP/1.0
  CSeq: 431
  Content-Type: text/parameters
  Session: 12345678
  Content-Length: 15
  packets_received
  jitter

- C->S:

  RTSP/1.0 200 OK
  CSeq: 431
  Content-Length: 46
  Content-Type: text/parameters
  packets_received: 10
  jitter: 0.3838

- "text/parameters"段只是参数类型的一个例子。对此方法有意的进行了松散的定义，对于答复和响应的内容将在更深一层的实验中给出定义。

5. RTSP - Methods: SET_PARAMETER

- 此方法给URI指定的表示或媒体流设置参数值。
- 为了客户端能检查某个特定的请求为何失败，请求应该只附带一个参数。当请求附带多个参数时，服务器只有在这些参数全都设置正确时才作出响应。服务器必须允许某个参数被重复设置成相同的值，但可能不允许改变参数值。
- 注意：必须只能使用SETUP命令来给媒体流设置传输参数。 限制只有SETUP能设置传输参数有利于防火墙设计。
- 示例：
- C->S:

  SET_PARAMETER rtsp://example.com/fizzle/foo RTSP/1.0
  CSeq: 421
  Content-length: 20
  Content-type: text/parameters
  barparam: barstuff

- S->C:

  RTSP/1.0 451 Invalid Parameter
  CSeq: 421
  Content-length: 10
  Content-type: text/parameters
  barparam

"text/parameters"段只是参数类型的一个例子。对此方法有意的进行了松散的定义。

5. RTSP - Methods: REDIRECT

- REDIRECT请求告知客户端连接到另一个服务器位置。它包含首部域Location，该域指出了客户端应该发出请求的URL。它可能包含参数Range，在重定向生效时，该域指明了媒体流的范围。如果客户端希望继续发送或接收其URI指定的媒体，它必须发出一个TEARDOWN请求来关闭当前会话，并向委派的主机发送SETUP以建立新的会话。
- 本例中，在给定的播放时间将URI请求重定向到新的服务器：
- S->C:

  REDIRECT rtsp://example.com/fizzle/foo RTSP/1.0

  CSeq: 732

  Location: rtsp://bigserver.com:8001

  Range: clock=19960213T143205Z-

5. RTSP - Methods: RECORD

- 此方法根据表示描述开始记录媒体数据。时间戳（timestamp）表现了起始和结束时间（UTC）。
- 如果没有给定时间范围，就使用表示描述中提供的开始和结束时间。如果会话已经开启，立即开始记录。由服务器来决定是否存储记录的数据到请求URI下或者其它URI下。如果服务器没有使用请求URI，那么响应代码应该是201（创建），并且包含一个实体，该实体描述了请求的状态，并通过Location首部域指向新资源。
- 允许记录现场表示（live presentations）的媒体服务器必须支持时钟范围格式（the clock range format），smpte格式对此无用。
- 在本示例中，媒体服务器被邀请到指定的会议
- C->S:

  RECORD rtsp://example.com/meeting/audio.en RTSP/1.0

  CSeq: 954

  Session: 12345678

  Conference: 128.16.64.19/32492374

6. RTSP - streaming data formats: ISMA and TS

- ISMA
  - 1) RFC standard
  - 2) Seperate, independent tracks for audio and video
  - 3) Advantages: Expansibility, extentable for subtitle, NLS streams, or multiple videos.
  - 4) Disadvantage: Need to do something like lip sync.
- TS
  - 1) China Gov（*The famous "光腚"总局*） standard
  - 2) Mux audio and video data to one track.
  - 3) Advantage: Easy to mux and transmit
  - 4) Disadvantage: not extensible for the streaming data

7. Protocols involved

- Commands/Signaling messages
  - HTTP, Web Service, WSDL, SOAP, XML, etc.
- Streaming
  - RTSP, SDP, RTP, RTCP, etc.
- Others
  - UPNP, DNS, etc.

8. Others

- Properties of RTSP
  - Extendable:New methods and parameters can be easily added to RTSP.
  - Easy to parse(Can be parsed by standard HTTP or MIME parsers).
  - Secure
  - Transport-independent
  - Multi-server capable
  - Control of recording devices
  - Separation of stream control and conference initiation
  - Suitable for professional applications
  - Presentation description neutral
  - Proxy and firewall friendly
  - HTTP-friendly
  - Appropriate server control(Play/Pause)
  - Transport negotiation
  - Capability negotiation

Demo:

EasyIPCam for Android

Download:

http://rg4.net/p/easyipcam

百度手机助手搜：EasyIPCam

THANKS

苏州市科达科技股份有限公司上海分公司
上海虹梅路2071号远中产业园1号楼4楼
Tel: 8621-54072200-7851
Mobile: +8618601760035
Email: weiguohua@kedacom.com, jacky@rg4.net（非工作时间）
Website: www.kedacom.com