RTSP协议

lory17 关注
0.907 2017.03.04 10:46:50 字数 1,777 阅读 23,942

RTSP简介

RTSP(Real Time Streaming Protocol)是由Real Network和Netscape共同提出的如何有效地在IP网络上传输流媒体数据的应用层协议。RTSP对流媒体提供了诸如暂停，快进等控制，而它本身并不传输数据，RTSP的作用相当于流媒体服务器的远程控制。服务器端可以自行选择使用TCP或UDP来传送串流内容，它的语法和运作跟HTTP 1.1类似，但并不特别强调时间同步，所以比较能容忍网络延迟。

RTSP和HTTP RTP(RTCP)的关系

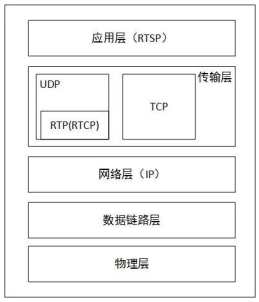
RTSP和HTTP

- 联系：两者都用纯文本来发送消息，且rtsp协议的语法也和HTTP类似。Rtsp一开始这样设计，也是为了能够兼容使用以前写的HTTP协议分析代码 。
- 区别：rtsp是有状态的，不同的是RTSP的命令需要知道现在正处于一个什么状态，也就是说rtsp的命令总是按照顺序来发送，某个命令总在另外一个命令之前要发送。Rtsp不管处于什么状态都不会断掉连接。而http则不保存状态，协议在发送一个命令以后，连接就会断开，且命令之间是没有依赖性。rtsp协议使用554端口，http使用80端口。

RTSP和RTP(RTCP)

- RTP：Realtime Transport Potocol 实时传输协议
  RTP提供时间标志,序列号以及其他能够保证在实时数据传输时处理时间的方法。
- RTCP：Realtime Transport Control Potocol 实时传输控制协议
  RTCP是RTP的控制部分,用来保证服务质量和成员管理。RTP和RTCP是一起使用的。
- RTSP：RealTime Streaming Potocol 实时流协议
  RTSP具体数据传输交给RTP,提供对流的远程控制

RTP是基于 UDP协议的，UDP不用建立连接，效率更高；但允许丢包，这就要求在重新组装媒体的时候多做些工作
RTP只是包裹内容信息，而RTCP是交换控制信息的，Qos是通过RTCP实现的
应用程序对应的是play, seek, pause, stop等命令，RTSP则是处理这些命令，在UDP传输时并使用RTP(RTCP)来完成。如果是TCP连接则不会使用RTP(RTCP)。



RTSP structure

RTSP的client连接server通过SDP(会话描述协议)传递信息，详细请见：RTSP消息

RTSP消息

RTSP的消息有两大类，一是请求消息(request)，一是回应消息(response)，两种消息的格式不同。
请求消息格式：

> 方法 URI RTSP版本 CR LF
> 消息头 CR LF CR LF
> 消息体 CR LF

方法包括：OPTIONS、SETUP、PLAY、TEARDOWN DESCRIBE
URI是接收方(服务端)的地址，例如：rtsp://192.168.22.136:5000/v0
每行后面的CR LF表示回车换行，需要接收端有相应的解析，消息头需要有两个CR LF。

```
DESCRIBE rtsp://192.168.1.211 RTSP/1.0
CSeq: 1
Accept: application/sdp
User-Agent: mangus-fc
```

回应消息格式：

> RTSP版本 状态码 解释 CR LF
> 消息头 CR LF CR LF
> 消息体 CR LF

其中RTSP版本一般都是RTSP/1.0，状态码是一个数值，200表示成功，解释是与状态码对应的文本解释，详细请见：SDP协议介绍。

```
RTSP/1.0 200 OK
CSeq: 1
Server: GrandStream Rtsp Server V100R001
Content-Type: application/sdp
Content-length: 256
Content-Base: rtsp://192.168.1.211/0

v=0
o=StreamingServer 3331435948 1116907222000 IN IP4 192.168.1.211
s=h264.mp4
c=IN IP4 0.0.0.0
t=0 0
a=control:*
m=video 0 RTP/AVP 96
a=control:trackID=0
a=rtpmap:96 H264/90000
m=audio 0 RTP/AVP 97
a=control:trackID=1
a=rtpmap:97 G726-16/8000
```

简单的rtsp交互过程：

C表示rtsp客户端，S表示rtsp服务端

> step1:
> C->S:OPTION request //询问S有哪些方法可用
> S->C:OPTION response //S回应信息中包括提供的所有可用方法

> step2:
> C->S:DESCRIBE request //要求得到S提供的媒体初始化描述信息
> S->C:DESCRIBE response //S回应媒体初始化描述信息，主要是sdp

> step3:
> C->S:SETUP request //设置会话的属性，以及传输模式，提醒S建立会话
> S->C:SETUP response //S建立会话，返回会话标识符，以及会话相关信息

> step4:

热门故事
闺蜜的一条朋友圈，结束了我和老公5年的婚姻
娘家拆迁分了两套新房，婆婆让我把房本写上小叔子的名？
前任一哭，现任必输
老婆偷偷拿30万给小舅子买豪车，被我一招"制服"

推荐阅读
TCP/IP各个协议层的网络延迟检测方法总结 阅读 261
HTTP首部有哪些字段？ 阅读 364
前端面试09-1：介绍知道的 http 返回的状态码 阅读 115
Python-snmplib：544问题和认证问题解决 阅读 55
2022最新版🔗Fiddler抓包教程(1) 初识Fiddler+HTTP基础知识 (围... 总) 阅读 218

C->S:PLAY request //C请求播放
S->C:PLAY response //S回应该请求的信息

S->C:发送流媒体数据

step5:
C->S:TEARDOWN request //C请求关闭会话
S->C:TEARDOWN response //S回应该请求

RTSP中常用方法

OPTION

得到服务器提供的可用方法

```
OPTIONS rtsp://192.168.20.136:5000/xxx666 RTSP/1.0
CSeq: 1 //每个消息都有序号来标记，第一个包通常是option请求消息
User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)
```

服务器的回应信息包括提供的一些方法,例如:

```
RTSP/1.0 200 OK
Server: UServer 0.9.7_rc1
Cseq: 1 //每个回应消息的cseq数值和请求消息的cseq相对应
Public: OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, SCALE,GET_PARAMETER //服务器提供的可用的方法
```

DESCRIBE

C向S发起DESCRIBE请求,为了得到会话描述信息(SDP):

```
DESCRIBE rtsp://192.168.20.136:5000/xxx666 RTSP/1.0
CSeq: 2
token:
Accept: application/sdp
User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)
```

服务器回应一些对此会话的描述信息(sdp):

```
RTSP/1.0 200 OK
Server: UServer 0.9.7_rc1
Cseq: 2
x-prev-url: rtsp://192.168.20.136:5000
x-next-url: rtsp://192.168.20.136:5000
x-Accept-Retransmit: our-retransmit
x-Accept-Dynamic-Rate: 1
Cache-Control: must-revalidate
Last-Modified: Fri, 10 Nov 2006 12:34:38 GMT
Date: Fri, 10 Nov 2006 12:34:38 GMT
Expires: Fri, 10 Nov 2006 12:34:38 GMT
Content-Base: rtsp://192.168.20.136:5000/xxx666/
Content-Length: 344
Content-Type: application/sdp

v=0 //以下都是sdp信息
o=OnewaveUServerNG 1451516402 1025358037 IN IP4 192.168.20.136
s=/xxx666
u=http:///
e=admin@
c=IN IP4 0.0.0.0
t=0 0
a=isma-compliance:1,1.0,1

a=range:npt=0-
m=video 0 RTP/AVP 96 //m表示媒体描述，下面是对会话中视频通道的媒体描述
a=rtpmap:96 MP4V-ES/90000
a=fmtp:96 profile-level-id=245;config=000001B0F5000001B509000001000000012000C888B0E0E0FA62D089028307 a=control:trackI
```

SETUP

客户端提醒服务器建立会话,并确定传输模式:

```
SETUP rtsp://192.168.20.136:5000/xxx666/trackID=0 RTSP/1.0
CSeq: 3
Transport: RTP/AVP/TCP;unicast;interleaved=0-1
User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)
 //url中 带有trackID=0，表示对该通道进行设置，Transport参数设置了传输模式，包的结构。接下来的数据包头部第二个字节位置就是 interleave
```

服务器回应信息:

```
RTSP/1.0 200 OK
Server: UServer 0.9.7_rc1
Cseq: 3
Session: 6310936469860791894 //服务器回应的会话标识符
Cache-Control: no-cache
Transport: RTP/AVP/TCP;unicast;interleaved=0-1;ssrc=68B84567
```

PLAY

客户端发送播放请求:

```
PLAY rtsp://192.168.20.136:5000/xxx666 RTSP/1.0
CSeq: 4
Session: 6310936469860791894
Range: npt=0.000- //设置播放时间的范围
User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)
```

服务器回应信息:

```
RTSP/1.0 200 OK
Server: UServer 0.9.7_rc1
Cseq: 4
Session: 6310936469860791894
Range: npt=0.000000-
RTP-Info: url=trackID=0;seq=17040;rtptime=1467265309
 //seq和rtptime都是rtp包中的信息
```

TEARDOWN

客户端发起关闭请求:

```
TEARDOWN rtsp://192.168.20.136:5000/xxx666 RTSP/1.0
CSeq: 5
Session: 6310936469860791894
User-Agent: VLC media player (LIVE555 Streaming Media v2005.11.10)
```

服务器回应:

```
RTSP/1.0 200 OK
Server: UServer 0.9.7_rc1
Cseq: 5
Session: 6310936469860791894
```

SDP协议

sdp的格式:

```
v=<version>
o=<username> <session id> <version> <network type> <address type> <address>
s=<session name>
i=<session description>
u=<URI>
e=<email address>
p=<phone number>
c=<network type> <address type> <connection address>
b=<modifier>:<bandwidth-value>
t=<start time> <stop time>
r=<repeat interval> <active duration> <list of offsets from start-time>
z=<adjustment time> <offset> <adjustment time> <offset> ....
```

```
k=<method>
k=<method>:<encryption key>
a=<attribute>
a=<attribute>:<value>
m=<media> <port> <transport> <fmt list>
```

v =（协议版本）
o =（所有者/创建者和会话标识符）
s =（会话名称）
i = *（会话信息）
u = *（URI 描述）
e = *（Email 地址）
p = *（电话号码）
c = *（连接信息）
b = *（带宽信息）
z = *（时间区域调整）
k = *（加密密钥）
a = *（0 个或多个会话属性行）

- 时间描述：
  t =（会话活动时间）
  r = *（0或多次重复次数）

- 媒体描述：
  m =（媒体名称和传输地址）
  i = *（媒体标题）
  c = *（连接信息 — 如果包含在会话层则该字段可选）
  b = *（带宽信息）
  k = *（加密密钥）
  a = *（0 个或多个媒体属性行）

SDP一会话描述协议一描述SAP、SIP和RTSR会话的协议,是一种文件描述协议,是由服务器生成的描述媒体文件编码信息以及所在服务器的链接等的信息。在多媒体会话 中sDP传送有关媒体流的信息,使会话描述的参人方加人会话。SDP主要用于Internet网中,但也可以在其它网络环境下使用。SDP十分通用,可描述其它网络环境中的会话,但主要用 于Internet中。在Internet环境下,SDP有两个主要目的:一是表明会话存在,二是传送足够信息给接收方,以便能加人、参加该会话。SDP所传达的信息包括:会话名称和目的,会话 活动时间,组成会话媒体种类,接收这些媒体的控制信息(如地址、端口、格式、带宽和会议管理人员资料等)。

总结:在RTSP交互过程中，只要在客户端发出Describe请求的时候，服务端回应的时候会有SDP消息发出，用SDP来描述会话情况和内容，方便客户端能够加入该会话。

RTSP基于libcurl代码实现

```
/*
 * Copyright (c) 2011, Jim Hollinger
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *   * Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *   * Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *   * Neither the name of Jim Hollinger nor the names of its contributors
 *     may be used to endorse or promote products derived from this
 *     software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */
/* <DESC>
 * A basic RTSP transfer
 * </DESC>
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <curl/curl.h>

#if defined (WIN32)
#include <conio.h>  /* _getch() */
#else
#include <termios.h>
#include <unistd.h>

#define VERSION_STR  "V1.0"

/* error handling macros */
#define my_curl_easy_setopt(A, B, C)                        \
  res = curl_easy_setopt((A), (B), (C));                    \
  if(!res)                                                  \
    fprintf(stderr, "curl_easy_setopt(%s, %s, %s) failed: %d\n", \
            #A, #B, #C, res);

#define my_curl_easy_perform(A)                             \
  res = curl_easy_perform(A);                               \
  if(!res)                                                  \
    fprintf(stderr, "curl_easy_perform(%s) failed: %d\n", #A, res);

static int _getch(void)
{
  struct termios oldt, newt;
  int ch;
  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
  newt.c_lflag &= ~( ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  ch = getchar();
  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  return ch;
}
#endif

/* send RTSP OPTIONS request */
static void rtsp_options(CURL *curl, const char *uri)
{
  CURLcode res = CURLE_OK;
  printf("\nRTSP: OPTIONS %s\n", uri);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_STREAM_URI, uri);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_REQUEST, (long)CURL_RTSPREQ_OPTIONS);
  my_curl_easy_perform(curl);
}

/* send RTSP DESCRIBE request and write sdp response to a file */
static void rtsp_describe(CURL *curl, const char *uri,
                          const char *sdp_filename)
{
  CURLcode res = CURLE_OK;
  FILE *sdp_fp = fopen(sdp_filename, "wb");
  printf("\nRTSP: DESCRIBE %s\n", uri);
  if(sdp_fp == NULL) {
    fprintf(stderr, "Could not open '%s' for writing\n", sdp_filename);
    sdp_fp = stdout;
  }
  else {
    printf("Writing SDP to '%s'\n", sdp_filename);
  }
  my_curl_easy_setopt(curl, CURLOPT_WRITEDATA, sdp_fp);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_REQUEST, (long)CURL_RTSPREQ_DESCRIBE);
  my_curl_easy_perform(curl);
  my_curl_easy_setopt(curl, CURLOPT_WRITEDATA, stdout);
  if(sdp_fp != stdout) {
    fclose(sdp_fp);
  }
}

/* send RTSP SETUP request */
static void rtsp_setup(CURL *curl, const char *uri, const char *transport)
{
  CURLcode res = CURLE_OK;
  printf("\nRTSP: SETUP %s\n", uri);
  printf("      TRANSPORT %s\n", transport);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_STREAM_URI, uri);
```

```
  my_curl_easy_setopt(curl, CURLOPT_RTSP_TRANSPORT, transport);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_REQUEST, (long)CURL_RTSPREQ_SETUP);
  my_curl_easy_perform(curl);
}

/* send RTSP PLAY request */
static void rtsp_play(CURL *curl, const char *uri, const char *range)
{
  CURLcode res = CURLE_OK;
  printf("\nRTSP: PLAY %s\n", uri);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_STREAM_URI, uri);
  my_curl_easy_setopt(curl, CURLOPT_RANGE, range);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_REQUEST, (long)CURL_RTSPREQ_PLAY);
  my_curl_easy_perform(curl);
}

/* send RTSP TEARDOWN request */
static void rtsp_teardown(CURL *curl, const char *uri)
{
  CURLcode res = CURLE_OK;
  printf("\nRTSP: TEARDOWN %s\n", uri);
  my_curl_easy_setopt(curl, CURLOPT_RTSP_REQUEST, (long)CURL_RTSPREQ_TEARDOWN);
  my_curl_easy_perform(curl);
}

/* convert url into an sdp filename */
static void get_sdp_filename(const char *url, char *sdp_filename,
                             size_t namelen)
{
  const char *s = strrchr(url, '/');
  strcpy(sdp_filename, "video.sdp");
  if(s != NULL) {
    s++;
    if(s[0] != '\0') {
      snprintf(sdp_filename, namelen, "%s.sdp", s);
    }
  }
}

/* scan sdp file for media control attribute */
static void get_media_control_attribute(const char *sdp_filename,
                                        char *control)
{
  int max_len = 256;
  char *s = malloc(max_len);
  FILE *sdp_fp = fopen(sdp_filename, "rb");
  control[0] = '\0';
  if(sdp_fp != NULL) {
    while(fgets(s, max_len - 2, sdp_fp) != NULL) {
      sscanf(s, " a = control: %s", control);
    }
    fclose(sdp_fp);
  }
  free(s);
}

/* main app */
int main(int argc, char * const argv[])
{
#if 1
  const char *transport = "RTP/AVP;unicast;client_port=1234-1235";  /* UDP */
#else
  /* TCP */
  const char *transport = "RTP/AVP/TCP;unicast;client_port=1234-1235";
#endif
  const char *range = "0.000-";
  int rc = EXIT_SUCCESS;
  char *base_name = NULL;

  printf("\nRTSP request %s\n", VERSION_STR);
  printf("    Project web site: http://code.google.com/p/rtsprequest/\n");
  printf("    Requires curl V7.20 or greater\n\n");

  /* check command line */
  if((argc != 2) && (argc != 3)) {
    base_name = strrchr(argv[0], '/');
    if(base_name == NULL) {
      base_name = strrchr(argv[0], '\\');
    }
    if(base_name == NULL) {
      base_name = argv[0];
    }
    else {
      base_name++;
    }
    printf("Usage:   %s url [transport]\n", base_name);
    printf("         url of video server\n");
    printf("         transport (optional) specifier for media stream"
           " protocol\n");
    printf("         default transport: %s\n", transport);
    printf("Example: %s rtsp://192.168.0.2/media/video1\n\n", base_name);
    rc = EXIT_FAILURE;
  }
  else {
    const char *url = argv[1];
    char *uri = malloc(strlen(url) + 32);
    char *sdp_filename = malloc(strlen(url) + 32);
    char *control = malloc(strlen(url) + 32);
    CURLcode res;
    get_sdp_filename(url, sdp_filename, strlen(url) + 32);
    if(argc == 3) {
      transport = argv[2];
    }

    /* initialize curl */
    res = curl_global_init(CURL_GLOBAL_ALL);
    if(res == CURLE_OK) {
      curl_version_info_data *data = curl_version_info(CURLVERSION_NOW);
      CURL *curl;
      fprintf(stderr, "    curl V%s loaded\n", data->version);

      /* initialize this curl session */
      curl = curl_easy_init();
      if(curl != NULL) {
        my_curl_easy_setopt(curl, CURLOPT_VERBOSE, 0L);
        my_curl_easy_setopt(curl, CURLOPT_NOPROGRESS, 1L);
        my_curl_easy_setopt(curl, CURLOPT_HEADERDATA, stdout);
        my_curl_easy_setopt(curl, CURLOPT_URL, url);

        /* request server options */
        snprintf(uri, strlen(url) + 32, "%s", url);
        rtsp_options(curl, uri);

        /* request session description and write response to sdp file */
        rtsp_describe(curl, uri, sdp_filename);

        /* get media control attribute from sdp file */
        get_media_control_attribute(sdp_filename, control);

        /* setup media stream */
        snprintf(uri, strlen(url) + 32, "%s/%s", url, control);
        rtsp_setup(curl, uri, transport);

        /* start playing media stream */
        snprintf(uri, strlen(url) + 32, "%s/", url);
        rtsp_play(curl, uri, range);
        printf("Playing video, press any key to stop ...");
        _getch();
        printf("\n");

        /* teardown session */
        rtsp_teardown(curl, uri);

        /* cleanup */
        curl_easy_cleanup(curl);
        curl = NULL;
      }
      else {
        fprintf(stderr, "curl_easy_init() failed\n");
      }
      curl_global_cleanup();
    }
    else {
      fprintf(stderr, "curl_global_init(%s) failed: %d\n",
              "CURL_GLOBAL_ALL", res);
    }
    free(control);
    free(sdp_filename);
    free(uri);
  }

  return rc;
}
```

参考资料

- csdn相关博客
- 百度百科

lory17
总资产1 共写了1777字 获得11个赞 共4个粉丝
关注

写下你的评论...

全部评论 0 只看作者

按时间倒序 按时间正序

推荐阅读 更多精彩内容 >

RTSP协议 分析
第一部分:RTSP协议 一、RTSP协议概述 RTSP(Real-TimeStream Protocol)是一种...
小鱼儿喜欢花无缺 阅读 2,274 评论 1 赞 6

Live555源码解析(2) - RTSP协议概述
上一篇Live555源码解析(1) - Main 寻根问祖，留其筋骨将main()面数脉络做了整体分析，通常来讲本...
SniperPan 阅读 3,716 评论 0 赞 14

RTSP Spec中文版(1-11)
RFC 2326RTSP Spec中文版(1-11)RTSP Spec中文版(12-16)RTSP Spec中文版...
SniperPan 阅读 3,954 评论 3 赞 9

流媒体协议
RTP 参考文档 RFC3550/RFC3551 Real-time Transport Protocol)是用于...
大草原之夜 阅读 919 评论 0 赞 9

RTSP协议总结
参考原文链接:http://blog.csdn.net/maopig/article/details/666525...
戴同学不呆 阅读 3,316 评论 0 赞 5

海底的眼泪
记得最后一个电话 记得耳边的余温 你嗅闻发出的甜蜜 我做了许多巧克力 一段恋情的结束 是一段酸涩的开始 因巧克力而...
江小昨 阅读 101 评论 9 赞 6

望远
沪江城中流 七星湖水幽 滇越传灵气 青山何来愁
良安书室 阅读 177 评论 13 赞 4

写下你的评论... 评论0 赞11