

JT同学

初级4年

暂无认证

105

2万+

91万+

46万+

原创

周排名

总排名

总排名

等级

5440

1861

1016

346

3845

积分

粉丝

获赞

评论

收藏

私信

关注

搜博主文章

热门文章

从零基础写一个RTSP服务器（一）RTSP协议讲解

C++ deque的用法与示例

Linux I2C驱动框架（超详细）

我的开源项目-RtpServer

从零基础写一个RTSP服务器（二）RTSP协议的实现

最新评论

我的开源项目-RtpServer

kaifa3n: 我测试 /h264_rtp_server test.h264 也会这样的问题？请问解决了没？ m...

使用mp4v2封装H.264或mp4最简单元示例

热心兔斯基: 应该是播放器的原因，换个播放器试试

使用mp4v2封装H.264或mp4最简单元示例

al604233436: h264的标准中，好像默认的设置是90KHz，所以我理解timeScale...

使用mp4v2封装H.264或mp4最简单元示例

al604233436: 有遇到，不知道原因呢。

深入学习Linux摄像头（二）v4l2驱动框架

qq_43745897: 请问往缓存区里面来采集图像数据的帧率是多少呢？请问如果缓存区...

您愿意向朋友推荐“博客详情页”吗？

强烈推荐

不推荐

一般般

推荐

强烈推荐

最新文章

深入浅出MySQL事务（二）MVCC的实现原理

深入浅出MySQL事务（一）事务隔离

深入浅出MySQL索引（二）InnoDB存储引擎的索引

2020年 4篇

2019年 95篇

2018年 7篇

从零开始写一个RTSP服务器（九）一个RTP OVER RTSP/TCP的RTSP服务器

JT同学

于 2019-08-13 15:59:33 发布

7872 收藏 42

分类专栏:

从零基础写一个RTSP服务器

文章标签:

rtsp

流媒体

H.264

版权

从零基础写一个RTS...

专栏收录该内容

187 订阅

10 篇文章

订阅专栏

从零开始写一个RTSP服务器系列

★我的开源项目-RtpServer

从零开始写一个RTSP服务器（一）RTSP协议讲解

从零开始写一个RTSP服务器（二）RTSP协议的实现

从零开始写一个RTSP服务器（三）RTP传输H.264

从零开始写一个RTSP服务器（四）一个传输H.264的RTSP服务器

从零开始写一个RTSP服务器（五）RTP传输AAC

从零开始写一个RTSP服务器（六）一个传输AAC的RTSP服务器

从零开始写一个RTSP服务器（七）多播传输RTP包

从零开始写一个RTSP服务器（八）一个多播的RTSP服务器

从零开始写一个RTSP服务器（九）一个RTP OVER RTSP/TCP的RTSP服务器

从零开始写一个RTSP服务器（九）一个RTP OVER RTSP/TCP的RTSP服务器

文章目录

从零开始写一个RTSP服务器（九）一个RTP OVER RTSP/TCP的RTSP服务器

一、RTP OVER RTSP(TCP)的实现

1.1 发送RTP包方式

1.2 如何区分RTSP、RTP、RTCP?

二、RTP OVER RTSP(TCP)的RTSP交互过程

OPTIONS

DESCRIBE

SETUP

PLAY

三、实现过程

3.1 RTP发包

3.2 服务器的实现

3.2.1 创建socket套接字

3.2.2 接收客户端连接

3.2.3 解析请求

3.2.4 处理请求

3.2.5 发送RTP包

3.3 源码

rtsp_server.c

tcp_rtp.h

tcp_rtp.c

四、测试

一、RTP OVER RTSP(TCP)的实现

1.1 发送RTP包方式

对于RTP OVER UDP 的实现，我们使用TCP连接来发送RTSP交互，然后创建新的UDP套接字来发送RTP包

对于RTP OVER RTSP(TCP)来说，我们会复用使用原先发送RTSP的socket来发送RTP包

1.2 如何区分RTSP、RTP、RTCP?

如上面所说，我们复用发送RTSP交互的socket来发送RTP包和RTCP信息，那么对于客户端来说，如何区分这三种数据呢？

我们将这三个分为两类，一类是RTSP，一类是RTP、RTCP

发送RTSP信息的情况没有变化，还是更以前一样的方式

发送RTP、RTCP包，在每个包前面都加上四个字节

字节	描述
第一个字节	‘\$’，标识符，用于与RTSP区分
第二个字节	channel，用于区分RTP和RTCP
第三和第四字节	RTP包的大小

由此我们可知，第一个字节\$用于与RTSP区分，第二个字节用于区分RTP和RTCP

RTP和RTCP的channel是在RTSP的SETUP过程中，客户端发送给服务端的

所以现在RTP的打包方式要在之前的每个RTP包前面加上四个字节，如下所示



二、RTP OVER RTSP(TCP)的RTSP交互过程

OPTIONS

C→S

```
1 OPTIONS rtsp://127.0.0.1:8554/live RTSP/1.0\r\n
2 CSeq: 2\r\n
3 \r\n
```

S→C

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

DESCRIBE

C→S

```
1 DESCRIBE rtsp://127.0.0.1:8554/live RTSP/1.0\r\n
2 CSeq: 3\r\n
3 Accept: application/sdp\r\n
4 \r\n
```

S→C

```
1 RTSP/1.0 200 OK\r\n
2 CSeq: 3\r\n
3 Content-length: 146
4 Content-type: application/sdp
5 \r\n
6 v=0
7 o=- 91565667683 1 IN IP4 127.0.0.1
8 t=0 0
9 a=control:*
10 m=video 0 RTP/AVP 96
11 a=rtmap:96 H264/900000
12 a=framerate:25
13 a=control:track0
```

SETUP

C→S

```
1 SETUP rtsp://127.0.0.1:8554/live/track0 RTSP/1.0\r\n
2 CSeq: 4\r\n
3 Transport: RTP/AVP/TCP;unicast;interleaved=0-2\r\n
4 \r\n
```

RTSP/AVP/TCP 表示使用RTP OVER TCP，interleaved=0-1表示这个会话连接的RTP channel为0，RTCP channel为1

目录

从零开始写一个RTSP服务器（九）一个...

文章目录

一、RTP OVER RTSP(TCP)的实现

1.1 发送RTP包方式

1.2 如何区分RTSP、RTP、RTCP?

二、RTP OVER RTSP(TCP)的RTSP交互...

OPTIONS

DESCRIBE

SETUP

PLAY

三、实现过程

分类专栏

C/C++	1篇
STL源码剖析	18篇
MySQL	4篇
Linux内核	14篇
分布式	
nginx	
从零开始写一个RTSP服...	10篇
live555源码分析与应用	9篇
Linux驱动	19篇

S→C

```
1 RTSP/1.0 200 OK\r\n
2 CSeq: 4\r\n
3 Transport: RTP/AVP/TCP;unicast;interleaved=0-1\r\n
4 Session: 327023c6\r\n
5 \r\n
```

PLAY

C→S

```
1 PLAY rtsp://127.0.0.1:8554/live RTSP/1.0\r\n
2 CSeq: 5\r\n
3 Session: 327023c6\r\n
4 Range: npt=0.000-\r\n
5 \r\n
```

S→C

```
1 RTSP/1.0 200 OK\r\n
2 CSeq: 5\r\n
3 Range: npt=0.000-\r\n
4 Session: 327023c6; timeout=60\r\n
5 \r\n
```

三、实现过程

3.1 RTP发包

经过上面的介绍，我们知道RTP OVER TCP和RTP OVER UDP的RTP发包方式是不同的，RTP OVER TCP需要在整个RTP包前面加上四个字节，为此我修改了RTP发包部分

RTP Packet 结构体

```
1 /*
2  * 作者: _JT_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 struct RtpPacket
7 {
8     char header[4];
9     struct RtpHeader rtpHeader;
10    uint8_t payload[0];
11 };
```

header: 前四个字节

rtpHeader: RTP包头部

payload: RTP包载荷

RTP的发包函数修改

每次发包前都需要添加四个字节的头，并且通过tcp发送

```
1 /*
2  * 作者: _JT_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 rtpSendPacket()
7 {
8     ...
9
10    rtpPacket->header[0] = '$';
11    rtpPacket->header[1] = rtpChannel;
12    rtpPacket->header[2] = (size & 0xFF00) >> 8;
13    rtpPacket->header[3] = size & 0xFF;
14
15    ...
```

3.2 服务器的实现

下面开始介绍RTP OVER TCP服务器的实现过程

3.2.1 创建socket套接字

```
1 /*
2  * 作者: _JT_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 main()
7 {
8     serverSockfd = createTcpSocket();
9     bindSocketAddr(serverSockfd, "0.0.0.0", SERVER_PORT);
10    listen(serverSockfd, 10);
11    ...
12    while(1)
13    {
14        ...
15    }
16 }
```

3.2.2 接收客户端连接

```
1 main()
2 {
3     ...
4     while(1)
5     {
6         acceptClient(serverSockfd, clientIp, &clientPort);
7         doClient(clientSockfd, clientIp, clientPort);
8     }
9 }
```

接收客户端连接后，执行doClient处理客户端请求

3.2.3 解析请求

接收请求后，解析请求。先解析方法，再解析序列号，如果是SETUP，那么就将RTP通道和RTCP通道解析出来

然后处理不同的请求方法

```
1 /*
2  * 作者: _JT_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 doClient()
7 {
8     while(1)
9     {
10        /* 接收数据 */
11        recv(clientSockfd, rBuf, BUF_MAX_SIZE, 0);
12
13        /* 解析命令 */
14        sscanf(line, "%s %s %s\r\n", method, url, version);
15        ...
16        sscanf(line, "CSeq: %d\r\n", &cseq);
17        ...
18        if(!strcmp(method, "SETUP"))
19            sscanf(line, "Transport: RTP/AVP/TCP;unicast;interleaved=%dhu-%dhu\r\n",
20                   &rtpChannel, &rtcpChannel);
21
22        /* 处理请求 */
23        if(!strcmp(method, "OPTIONS"))
24            handleCmd_OPTIONS(sBuf, cseq)
25        else if(!strcmp(method, "DESCRIBE"))
26            handleCmd_DESCRIBE(sBuf, cseq, url);
27        else if(!strcmp(method, "SETUP"))
28            handleCmd_SETUP(sBuf, cseq, rtpChannel);
29        else if(!strcmp(method, "PLAY"))
30            handleCmd_PLAY(sBuf, cseq);
31
32        send(clientSockfd, sBuf, strlen(sBuf), 0);
33    }
34 }
```

3.2.4 处理请求

OPTIONS

```
OPTIONS
```

```

1 handleCmd_OPTIONS()
2 {
3     sprintf(result, "RTSP/1.0 200 OK\r\n"
4               "CSeq: %d\r\n"
5               "Public: OPTIONS, DESCRIBE, SETUP, PLAY\r\n"
6               "\r\n",
7               cseq);
8 }

```

DESCRIBE

```

1 handleCmd_DESCRIBE()
2 {
3     sprintf(sdp, "v=0\r\n"
4             "o=- 981d 1 IN IP4 %s\r\n"
5             "t=0 0\r\n"
6             "a=control:* \r\n"
7             "m=video 0 RTP/AVP 96 \r\n"
8             "a=rtpmap:96 H264/90000 \r\n"
9             "a=control:track0 \r\n",
10            time(NULL), localIp);
11
12     sprintf(result, "RTSP/1.0 200 OK\r\nCSeq: %d\r\n"
13               "Content-Base: %s\r\n"
14               "Content-type: application/sdp\r\n"
15               "Content-length: %d\r\n\r\n",

```

SETUP

```

1 handleCmd_SETUP()
2 {
3     sprintf(result, "RTSP/1.0 200 OK\r\n"
4               "CSeq: %d\r\n"
5               "Transport: RTP/AVP/TCP;unicast;interleaved=3hhu-3hhu\r\n"
6               "Session: 66334873\r\n"
7               "\r\n",
8               cseq,
9               rtpChannel,
10              rtpChannel+1
11              );
12 }

```

PLAY

```

1 handleCmd_PLAY()
2 {
3     sprintf(result, "RTSP/1.0 200 OK\r\n"
4               "CSeq: %d\r\n"
5               "Range: npt=0.000- \r\n"
6               "Session: 66334873; timeout=60 \r\n\r\n",
7               cseq);
8 }

```

3.2.5 发送RTP包

在发送完PLAY回复之后，开始发送RTP包

```

1 /*
2  * 作者: _JT_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 doClient()
7 {
8     ...
9
10
11     while(1)
12     {
13         ...
14
15         send(clientSockfd, sBuf, strlen(sBuf), 0);

```

3.3 源码

rtsp_server.c

```

1 /*
2  * 作者: _JT_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <stdint.h>
9 #include <string.h>
10 #include <sys/types.h>
11 #include <sys/socket.h>
12 #include <sys/socket.h>
13 #include <netinet/in.h>
14 #include <arpa/inet.h>
15 #include <time.h>
16 #include <sys/types.h>
17 #include <sys/stat.h>
18 #include <fcntl.h>
19 #include <assert.h>
20
21 #include "tcp_rtp.h"
22
23 #define H264_FILE_NAME "test.h264"
24 #define SERVER_PORT 8554
25 #define BUF_MAX_SIZE (1024*1024)
26
27 static int createTcpSocket()
28 {
29     int sockfd;
30     int on = 1;
31
32     sockfd = socket(AF_INET, SOCK_STREAM, 0);
33     if(sockfd < 0)
34         return -1;
35
36     setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));
37
38     return sockfd;
39 }
40
41 static int createUdpSocket()
42 {
43     int sockfd;
44     int on = 1;
45
46     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
47     if(sockfd < 0)
48         return -1;
49
50     setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));
51
52     return sockfd;
53 }
54
55 static int bindSocketAddr(int sockfd, const char* ip, int port)
56 {
57     struct sockaddr_in addr;
58
59     addr.sin_family = AF_INET;
60     addr.sin_port = htons(port);
61     addr.sin_addr.s_addr = inet_addr(ip);
62
63     if(bind(sockfd, (struct sockaddr*)&addr, sizeof(struct sockaddr)) < 0)
64         return -1;
65
66     return 0;
67 }
68
69 static int acceptClient(int sockfd, char* ip, int* port)
70 {
71     int clientfd;
72     socklen_t len = 0;
73     struct sockaddr_in addr;
74
75     memset(&addr, 0, sizeof(addr));
76     len = sizeof(addr);
77
78     clientfd = accept(sockfd, (struct sockaddr*)&addr, &len);
79     if(clientfd < 0)
80         return -1;
81
82     strcpy(ip, inet_ntoa(addr.sin_addr));
83     *port = ntohs(addr.sin_port);
84 }

```

```

85     return clientfd;
86 }
87
88 static inline int startCode3(char* buf)
89 {
90     if(buf[0] == 0 && buf[1] == 0 && buf[2] == 1)
91         return 1;
92     else
93         return 0;
94 }
95
96 static inline int startCode4(char* buf)
97 {
98     if(buf[0] == 0 && buf[1] == 0 && buf[2] == 0 && buf[3] == 1)
99         return 1;
100     else
101         return 0;
102 }
103
104 static char* findNextStartCode(char* buf, int len)
105 {
106     int i;
107
108     if(len < 3)
109         return NULL;
110
111     for(i = 0; i < len-3; ++i)
112     {
113         if(startCode3(buf) || startCode4(buf))
114             return buf;
115
116         ++buf;
117     }
118
119     if(startCode3(buf))
120         return buf;
121
122     return NULL;
123 }
124
125 static int getFrameFromH264File(int fd, char* frame, int size)
126 {
127     int rSize, frameSize;
128     char* nextStartCode;
129
130     if(fd < 0)
131         return fd;
132
133     rSize = read(fd, frame, size);
134     if(!startCode3(frame) && !startCode4(frame))
135         return -1;
136
137     nextStartCode = findNextStartCode(frame+3, rSize-3);
138     if(!nextStartCode)
139     {
140         //lseek(fd, 0, SEEK_SET);
141         //frameSize = rSize;
142         return -1;
143     }
144     else
145     {
146         frameSize = (nextStartCode-frame);
147         lseek(fd, frameSize-rSize, SEEK_CUR);
148     }
149
150     return frameSize;
151 }
152
153 static int rtpSendH264Frame(int socket, int rtpChannel, struct RtpPacket* rtpPacket, uint8_t* frame, uint32_t f
154 {
155     uint8_t naluType; // nalu第一个字节
156     int sendBytes = 0;
157     int ret;
158
159     naluType = frame[0];
160
161     if (frameSize <= RTP_MAX_PKT_SIZE) // nalu长度小于最大包长: 单一NALU单元模式
162     {
163         /*
164          * 0 1 2 3 4 5 6 7 8 9
165          * +-----+
166          * |F|NRI| Type | a single NAL unit ... |
167          * +-----+
168          */
169         memcpy(rtpPacket->payload, frame, frameSize);
170         ret = rtpSendPacket(socket, rtpChannel, rtpPacket, frameSize);
171         if(ret < 0)
172             return -1;
173
174         rtpPacket->rtpheader.seq++;
175         sendBytes += ret;
176         if ((naluType & 0x1F) == 7 || (naluType & 0x1F) == 8) // 如果是SPS、PPS就不需要延时问题
177             goto out;
178     }
179     else // nalu长度小于最大包长: 分片模式
180     {
181         /*
182          * 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
183          * +-----+
184          * | FU indicator | FU header | FU payload ... |
185          * +-----+
186          */
187
188         /*
189          * FU Indicator
190          * 0 1 2 3 4 5 6 7
191          * +-----+
192          * |F|NRI| Type |
193          * +-----+
194          */
195
196         /*
197          * FU Header
198          * 0 1 2 3 4 5 6 7
199          * +-----+
200          * |S|E|R| Type |
201          * +-----+
202          */
203
204         int pktNum = frameSize / RTP_MAX_PKT_SIZE; // 有几个完整的包
205         int remainPktSize = frameSize % RTP_MAX_PKT_SIZE; // 剩余不完整包的大小
206
207         int i, pos = 1;
208
209         /* 发送完整的包 */
210         for (i = 0; i < pktNum; i++)
211         {
212             rtpPacket->payload[0] = (naluType & 0x60) | 28;
213             rtpPacket->payload[1] = naluType & 0x1F;
214
215             if (i == 0) //第一包数据
216                 rtpPacket->payload[1] |= 0x80; // start
217             else if (remainPktSize == 0 && i == pktNum - 1) //最后一包数据
218                 rtpPacket->payload[1] |= 0x40; // end
219
220             memcpy(rtpPacket->payload+2, frame-pos, RTP_MAX_PKT_SIZE);
221             ret = rtpSendPacket(socket, rtpChannel, rtpPacket, RTP_MAX_PKT_SIZE+2);
222             if(ret < 0)
223                 return -1;
224
225             rtpPacket->rtpheader.seq++;
226             sendBytes += ret;
227             pos += RTP_MAX_PKT_SIZE;
228         }
229
230         /* 发送剩余的数据 */
231         if (remainPktSize > 0)
232         {
233             rtpPacket->payload[0] = (naluType & 0x60) | 28;
234             rtpPacket->payload[1] = naluType & 0x1F;
235             rtpPacket->payload[1] |= 0x40; //end
236
237             memcpy(rtpPacket->payload+2, frame-pos, remainPktSize+2);
238             ret = rtpSendPacket(socket, rtpChannel, rtpPacket, remainPktSize+2);
239             if(ret < 0)
240                 return -1;
241
242             rtpPacket->rtpheader.seq++;
243             sendBytes += ret;
244         }
245     }
246
247     out:
248

```

```

250 }
251 return sendBytes;
252 }
253 static char* getlineFromBuf(char* buf, char* line)
254 {
255     while(*buf != '\n')
256     {
257         *line = *buf;
258         line++;
259         buf++;
260     }
261     *line = '\n';
262     ++line;
263     *line = '\0';
264     ++buf;
265     return buf;
266 }
267
268 static int handleCmd_OPTIONS(char* result, int cseq)
269 {
270     sprintf(result, "RTSP/1.0 200 OK\r\n"
271               "CSeq: %d\r\n"
272               "Public: OPTIONS, DESCRIBE, SETUP, PLAY\r\n"
273               "\r\n",
274             cseq);
275     return 0;
276 }
277
278 static int handleCmd_DESCRIBE(char* result, int cseq, char* url)
279 {
280     char sdp[500];
281     char localIp[100];
282     sscanf(url, "rtsp://%[^:]:", localIp);
283     sprintf(sdp, "v=0\r\n"
284             "o=- 9%ld 1 IN IP4 %s\r\n"
285             "t=0 0\r\n"
286             "a=control:\r\n"
287             "m=video 0 RTP/AVP 96\r\n"
288             "a=rtpmap:96 H264/90000\r\n"
289             "a=control:track0\r\n"
290             "time(NULL), localIp);
291     sprintf(result, "RTSP/1.0 200 OK\r\nCSeq: %d\r\n"
292             "Content-Base: %s\r\n"
293             "Content-type: application/sdp\r\n"
294             "Content-length: %d\r\n\r\n"
295             "%s",
296             cseq,
297             url,
298             strlen(sdp),
299             sdp);
300     return 0;
301 }
302
303 static int handleCmd_SETUP(char* result, int cseq, uint8_t rtpChannel)
304 {
305     sprintf(result, "RTSP/1.0 200 OK\r\n"
306             "CSeq: %d\r\n"
307             "Transport: RTP/AVP/TCP;unicast;interleaved-%dhu-%dhu\r\n"
308             "Session: 66334873\r\n"
309             "\r\n",
310             cseq,
311             rtpChannel,
312             rtpChannel+1);
313     return 0;
314 }
315
316 static int handleCmd_PLAY(char* result, int cseq)
317 {
318     sprintf(result, "RTSP/1.0 200 OK\r\n"
319             "CSeq: %d\r\n"
320             "Range: npt=0.000-∞\r\n"
321             "Session: 66334873; timeout=60\r\n\r\n",
322             cseq);
323     return 0;
324 }
325
326 static void doClient(int clientSockfd, const char* clientIP, int clientPort)
327 {
328     char method[40];
329     char url[100];
330     char version[40];
331     int cseq;
332     char* bufPtr;
333     char* rBuf = malloc(BUF_MAX_SIZE);
334     char* sBuf = malloc(BUF_MAX_SIZE);
335     char line[400];
336     uint8_t rtpChannel;
337     uint8_t rtcpChannel;
338     while(1)
339     {
340         int recvlen;
341         recvlen = recv(clientSockfd, rBuf, BUF_MAX_SIZE, 0);
342         if(recvlen <= 0)
343             goto out;
344         rBuf[recvlen] = '\0';
345         printf("-----C>S-----\n");
346         printf("%s", rBuf);
347         /* 解析方法 */
348         bufPtr = getlineFromBuf(rBuf, line);
349         if(sscanf(line, "%s %s %s\r\n", method, url, version) != 3)
350         {
351             printf("parse err\n");
352             goto out;
353         }
354         /* 解析序列号 */
355         bufPtr = getlineFromBuf(bufPtr, line);
356         if(sscanf(line, "CSeq: %d\r\n", &cseq) != 1)
357         {
358             printf("parse err\n");
359             goto out;
360         }
361         /* 如果是SETUP, 那么就用解析channel */
362         if(!strcmp(method, "SETUP"))
363         {
364             while(1)
365             {
366                 bufPtr = getlineFromBuf(bufPtr, line);
367                 if(!strcmp(line, "Transport: ", strlen("Transport:")))
368                 {
369                     sscanf(line, "Transport: RTP/AVP/TCP;unicast;interleaved-%dhu-%dhu\r\n",
370                             &rtpChannel, &rtcpChannel);
371                     break;
372                 }
373             }
374         }
375         if(!strcmp(method, "OPTIONS"))
376         {
377             if(handleCmd_OPTIONS(sBuf, cseq))
378             {
379                 printf("failed to handle options\n");
380                 goto out;
381             }
382         }
383         else if(!strcmp(method, "DESCRIBE"))
384         {
385             if(handleCmd_DESCRIBE(sBuf, cseq, url))
386             {
387                 printf("failed to handle describe\n");
388                 goto out;
389             }
390         }
391         else if(!strcmp(method, "SETUP"))
392         {
393             if(handleCmd_SETUP(sBuf, cseq, rtpChannel))
394             {
395                 printf("failed to handle setup\n");
396                 goto out;
397             }
398         }
399     }
400 }

```

```

413     }
414 }
415 else if(!strcmp(method, "PLAY"))
416 {
417     if(handleCmd_PLAY(sBuf, cseq))
418     {
419         printf("failed to handle play\n");
420         goto out;
421     }
422 }
423 else
424 {
425     goto out;
426 }
427 printf("-----S->C-----\n");
428 printf("Sa", sBuf);
429 send(clientSockfd, sBuf, strlen(sBuf), 0);
430
431 /* 开始播放, 发送RTP包 */
432 if(!strcmp(method, "PLAY"))
433 {
434     int frameSize, startCode;
435     char* frame = malloc(500000);
436     struct RtpPacket* rtpPacket = (struct RtpPacket*)malloc(500000);
437     int fd = open(H264_FILE_NAME, O_RDONLY);
438     assert(fd > 0);
439     rtpHeaderInit(rtpPacket, 0, 0, 0, RTP_VERSION, RTP_PAYLOAD_TYPE_H264, 0,
440                 0, 0, 0x88923423);
441
442     printf("start play\n");
443
444     while (1)
445     {
446         frameSize = getFrameFromH264File(fd, frame, 500000);
447         if(frameSize < 0)
448         {
449             break;
450         }
451
452         if(startCode3(frame))
453             startCode = 3;
454         else
455             startCode = 4;
456
457         frameSize -= startCode;
458
459         rtpSendH264Frame(clientSockfd, rtpChannel1, rtpPacket, frame+startCode, frameSize);
460         rtpPacket->rtpHeader.timestamp += 90000/25;
461
462         usleep(1000*1000/25);
463     }
464     free(frame);
465     free(rtpPacket);
466     goto out;
467 }
468
469 }
470
471 out:
472 printf("finish\n");
473 close(clientSockfd);
474 free(rBuf);
475 free(sBuf);
476 }
477
478 int main(int argc, char* argv[])
479 {
480     int serverSockfd;
481     int ret;
482
483     serverSockfd = createTcpSocket();
484     if(serverSockfd < 0)
485     {
486         printf("failed to create tcp socket\n");
487         return -1;
488     }
489
490     ret = bindSocketAddr(serverSockfd, "0.0.0.0", SERVER_PORT);
491     if(ret < 0)
492     {
493         printf("failed to bind addr\n");
494         return -1;
495     }
496
497     ret = listen(serverSockfd, 10);
498     if(ret < 0)
499     {
500         printf("failed to listen\n");
501         return -1;
502     }
503
504     printf("rtsp://127.0.0.1:%d\n", SERVER_PORT);
505
506     while(1)
507     {
508         int clientSockfd;
509         char clientIp[40];
510         int clientPort;
511
512         clientSockfd = acceptClient(serverSockfd, clientIp, &clientPort);
513         if(clientSockfd < 0)
514         {
515             printf("failed to accept client\n");
516             return -1;
517         }
518
519         printf("accept client;client ip:%s,client port:%d\n", clientIp, clientPort);
520
521         doClient(clientSockfd, clientIp, clientPort);
522     }
523
524     return 0;
525 }

```

tcp_rtp.h

```

1  /*
2   * 作者: _JT_
3   * 博客: https://blog.csdn.net/weixin_42462202
4   */
5
6  #ifndef _RTP_H_
7  #define _RTP_H_
8  #include <stdint.h>
9
10 #define RTP_VERSION      2
11
12 #define RTP_PAYLOAD_TYPE_H264    96
13 #define RTP_PAYLOAD_TYPE_AAC    97
14
15 #define RTP_HEADER_SIZE    12

```

tcp_rtp.c

```

1  /*
2   * 作者: _JT_
3   * 博客: https://blog.csdn.net/weixin_42462202
4   */
5
6  #include <sys/types.h>
7  #include <sys/socket.h>
8  #include <arpa/inet.h>
9  #include <netinet/in.h>
10 #include <arpa/inet.h>
11
12 #include "tcp_rtp.h"
13
14 void rtpHeaderInit(struct RtpPacket* rtpPacket, uint8_t csrcLen, uint8_t extension,
15                  uint8_t padding, uint8_t ver, uint8_t payloadType, uint8_t marker)

```

四、测试

将rtsp_server.c、tcp_rtp.h、tcp_rtp.c保存下来

编译运行，程序默认打开test.h264，如果你没有视频源的话，可以从RtspServer的example目录下获取

```

1  gcc rtsp_server.c tcp_rtp.c
2  ./a.out

```

运行后得到一个url

```

1  rtsp://127.0.0.1:8554

```

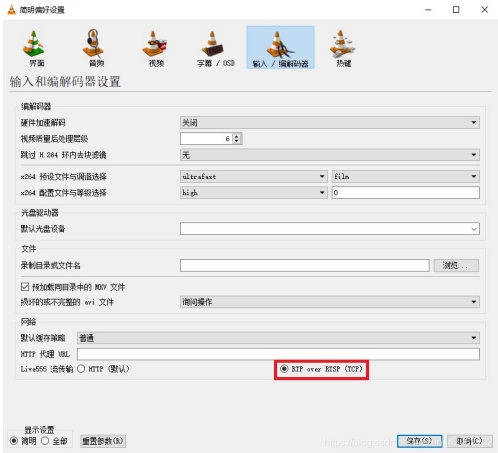
如何启动RTP over TCP?

需要设置Vc的模式

打开工具 >> 首选项 >> 输入/编解码器 >> live555 流传输 >> RTP over RTSP(TCP)

然后选择RTP over RTSP(TCP)

点击保存



输入url

运行效果



可算把这个系列写完了，还真是累，当然希望对于学习RTSP协议的人有所帮助，这也是我的初衷

- rtsp服务器 (C语言实现) 10-31
- rtsp-服务器, C语言实现, 编译运行没有问题, 是学习流媒体很不错的资料, 吐血上传.
- Rtsp-服务搭建 (Ffmpeg+Node.js+ffmpeg网络视频服务器) wtru2000的专栏 3681
- 当前有几个海康监控, 想接入MES系统, 去他们官方网站下载了两个web的sdk包, 分别为控件开发包和无插件开发包. 结果很坑, 控件, 要求为ie浏览器...
- RTSP 服务器C语言 09-10
- RTSP是实时流媒体传输协议. 服务器和客户端之间通过RTSP协议实现握手和认证过程, 通过RTP协议传输视频数据. 本资源通过C语言实现了RTSP服...
- RTSP协议介绍以及C语言实现具有发送H.264视频功能的RTSP服务器 破武当年少, 真美好时光 7352
- RTP封装H.264视频规范以及C语言实现 以前上学时做过嵌入式开发H.264的流媒体项目, 现在又突然想起来, 不想学过了就忘了浪费了, 所以又自己实...
- 搭建RTSP流媒体服务器的三种方式 最新发布 fang lovest yang的博客 4330
- 主要用于测试目的, 系统是windows, 使用的是docker desktop 3.5.1.1. rtsp-simple-server 官网: <https://github.com/aler9/rtsp-simple-server> (1) 下载...
- RtspServer实现及使用 LiaoJunXiong的博客 175+
- 编译环境: Ubuntu18.04 64位 交叉编译工具: arm-Hisiv500-linux-gcc 最近最近在h3519实现RtspServer, 以便于推流. 这里记录一下工作过程, 目前还...
- RTSP协议的一些分析 (三) ——简单的rtsp服务器的实现 yangguoyu0223的博客 612
- RTSP服务器有两个部分组成. 一个是RTSP的交互, 一个是RTP数据的传输, 本文主要实现RTSP服务的交互过程.
- rtsp 服务器搭建 yinshipin007的博客 4838
- rtsp 服务器搭建: 今天我们搭建这个 rtsp 服务器的名称叫做: ZLMediaKit, 它是一个基于 C++11 的高性能运营级流媒体服务框架, 类似我之前给大家搭...
- 从零开始写一个RTSP服务器 NBA_1的博客 101
- https://blog.csdn.net/weixin_42462202/article/details/98986535
- 从零开始写一个RTSP服务器 (二) RTSP协议的实现 Jf同学的博文 275+
- 从零开始写一个RTSP服务器系列 从零开始写一个RTSP服务器 (一) 不一样的RTSP协议讲解 从零开始写一个RTSP服务器 (二) RTSP协议的实现 从零...
- NGINX-RTMP直播服务器搭建-OBS录制推流-VLC视频播放 瓶上傅来终觉浅, 绝知此事要躬行 275+
- 网上关于视频直播的资料还是挺多的, 看了一些文章, 自己也动手实践了下, 主要有三个步骤: (1) NginxRTMP服务器搭建(2)视频录制推流 (3) 拉...
- ffmpeg命令行拉TCP的RTSP流的方法及使用测试记录——RTP over RTSP(TCP) weixiao的博客 7932
- 1、RTSP流流 (1080P * 1280帧码) : ./demo rdevivideo0 1920 1080 0 注: ./demo 为本地自写的推流工具, 在流媒体服务器上出现如下图记录, 则能...
- C++实现RTSP/RTMP服务器 cr88的博客 2414
- C++实现RTSP/RTMP服务器 前面介绍了rtsp, rtp, h264相关的知识, 记不清的可以回顾一下. 这篇我们来讲解如何使用c++自己写一个简单的基本的rtsp服...
- RTSP服务器 相见不如怀念 6726
- 一: 总体了解RTSP(Real-Time Stream Protocol) 是一种基于文本的应用层协议, 直白的讲客户端与服务端建立连接并服务端接收流, 服务器上的流可...
- 从零开始写一个RTSP服务器 (四) 一个传输H.264的RTSP服务器 Jf同学的博文 175+
- 从零开始写一个RTSP服务器系列 从零开始写一个RTSP服务器 (一) 不一样的RTSP协议讲解 从零开始写一个RTSP服务器 (二) RTSP协议的实现 从零...
- RTSP流媒体服务器的搭建与测试 06-16
- 本文主要介绍了live555搭建RTSP流媒体服务器, 并采用VLC软件进行相应测试
- 最小RTSP服务器, C语言代码 07-27
- 最小RTSP服务器, C语言代码, 每行都有我的注释, 适合新手.
- 从零开始写一个发送h264的rtsp服务器(上) jychen105的专栏 175+
- 从零开始写一个发送h264的rtsp服务器(上)一、什么是RTSP通常所说的rtsp协议其实包含三个协议: rtsp协议, rtp协议, rtcp协议各协议运作流程概要: 第...
- 从零开始写一个RTSP服务器系列 (一) RTSP协议讲解 入门指南 Jf同学的博文 675+
- 从零开始写一个RTSP服务器系列 从零开始写一个RTSP服务器 (一) 不一样的RTSP协议讲解 从零开始写一个RTSP服务器 (二) RTP传输H.264(待写) ...

“相关推荐”对你有帮助?

非常有帮助 有帮助 一般 没帮助

©2022 CSDN 皮肤主题: 数字20 设计师: CSDN官方博客 返回顶部