



```
1 | RTP/1.0 200 OK\r\n
2 | CSeq: 5\r\n
3 | Range: npt=0.000-\r\n
4 | Session: 66334873; timeout=60
5 | \r\n
```

## 二、多播的RTSP服务器实现过程

### 2.1 创建套接字

```
1 /*
2  * 作者: _3T_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 main()
7 {
8     /* 创建套接字 */
9     serverSockfd = createTcpSocket();
10
11     /* 绑定地址和端口 */
12     bindSocketAddr(serverSockfd, "0.0.0.0", SERVER_PORT);
13
14     /* 开始监听 */
15     listen(serverSockfd, 10);
16
17     ...
18
19     while(1)
20     {
21         ...
22     }
23 }
24
25
26
```

### 2.2 创建线程向多播地址推送RTP包

在进入while循环接收客户端前，我们创建一个线程不断地向多播地址发送RTP包

```
1 main()
2 {
3     ...
4     pthread_create(&threadId, NULL, sendRtpPacket, NULL);
5
6     while(1)
7     {
8         ...
9     }
10 }
```

下面看一看发送函数

```
1 /*
2  * 作者: _3T_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 sendRtpPacket()
7 {
8     ...
9     while(1)
10    {
11        ...
12
13        /* 获取一帧数据 */
14        getFrameFromI264File(fd, frame, 500000);
15
16        /* 向多播地址发送RTP包 */
17        rtpSendI264Frame(sockfd, MULTICAST_IP, MULTICAST_PORT,
18                        rtpPacket, frame-startCode, frameSize);
19
20        ...
21    }
22 }
```

### 2.2 接收客户端连接

进入while循环后，开始接收客户端，然后处理客户端请求

```
1 main()
2 {
3     ...
4
5     while(1)
6     {
7         /* 接收客户端 */
8         acceptClient(serverSockfd, clientIp, &clientPort);
9
10        /* 处理客户端 */
11        doClient(clientSockfd, clientIp, clientPort);
12    }
13 }
```

下面仔细看一看如何处理客户端请求

### 2.3 解析命令

先解析请求方法，然后解析序列号，再根据不同的请求做不同的处理，然后再放回结果给客户端

```
1 /*
2  * 作者: _3T_
3  * 博客: https://blog.csdn.net/weixin_42462202
4  */
5
6 doClient()
7 {
8     ...
9     while(1)
10    {
11        /* 接收请求 */
12        recv(clientSockfd, rBuf, BUF_MAX_SIZE, 0);
13
14        /* 解析请求方法 */
15        sscanf(line, "%s %s %s\r\n", method, url, version)
16
17        /* 解析序列号 */
18        sscanf(line, "CSeq: %d\r\n", &cseq);
19
20        /* 处理请求 */
21        if(!strcmp(method, "OPTIONS"))
22            handleCmd_OPTIONS(sBuf, cseq);
23        else if(!strcmp(method, "DESCRIBE"))
24            handleCmd_DESCRIBE(sBuf, cseq, url);
25        else if(!strcmp(method, "SETUP"))
26            handleCmd_SETUP(sBuf, cseq, localIp);
27        else if(!strcmp(method, "PLAY"))
28            handleCmd_PLAY(sBuf, cseq);
29
30        /* 返回结果给客户端 */
31        send(clientSockfd, sBuf, strlen(sBuf), 0);
32    }
33 }
```

### 2.4 处理请求

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

发送多播的sdp描述文件

```
1 handleCmd_DESCRIBE()
2 {
3     /* 多播sdp */
4     sprintf(sdp, "v=0\r\n"
5             "o=- %ld 1 IN IP4 %s\r\n"
6             "t=0 0\r\n"
```

```

7      "a=control:\r\n"
8      "a=type:broadcast\r\n"
9      "a=rtsp-unicast: reflection\r\n"
10     "m=video 3d RTP/AVP 96\r\n"
11     "c=IN IP4 %s/255\r\n"
12     "a=rtmpmap:96 H264/90000\r\n"
13     "a=control:track0\r\n",
14     time(NULL),
15     localIp,
16     MULTICAST_PORT,
17     MULTICAST_IP);
18
19     sprintf(result, "RTSP/1.0 200 OK\r\nCseq: %d\r\n"
20     "Content-Base: %s\r\n"
21     "Content-type: application/sdp\r\n"
22     "Content-length: %d\r\n\r\n"
23     "%s",
24     cseq,
25     url,
26     strlen(sdp),
27     sdp);
28 }

```

#### SETUP

```

1 handleCmd_SETUP()
2 {
3     sprintf(result, "RTSP/1.0 200 OK\r\n"
4     "Cseq: %d\r\n"
5     "Transport: RTP/AVP;multicast;destination=%s;"
6     "source=%s;port=%d-%d;ttl=255\r\n"
7     "Session: 66334873\r\n"
8     "\r\n",
9     cseq,
10    MULTICAST_IP,
11    localIp,
12    MULTICAST_PORT,
13    MULTICAST_PORT+1);
14 }

```

#### 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 }

```

在play回复完成之后，客户端就会去多播地址拉取媒体流，然后播放

#### 源码

源码有三个文件: [multicast\\_rtsp\\_server](#)、[rtsp.h](#)、[rtsp.c](#)

#### multicast\_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 <sys/socket.h>
14 #include <netinet/in.h>
15 #include <arpa/inet.h>
16 #include <time.h>
17 #include <sys/types.h>
18 #include <sys/stat.h>
19 #include <fcntl.h>
20 #include <assert.h>
21 #include <pthread.h>
22
23 #include "rtsp.h"
24
25 #define H264_FILE_NAME "test.h264"
26 #define MULTICAST_IP "239.255.255.11"
27 #define SERVER_PORT 8554
28 #define MULTICAST_PORT 9832
29 #define BUF_MAX_SIZE (1024*1024)
30
31 static int createTcpSocket()
32 {
33     int sockfd;
34     int on = 1;
35
36     sockfd = socket(AF_INET, SOCK_STREAM, 0);
37     if(sockfd < 0)
38         return -1;
39
40     setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));
41
42     return sockfd;
43 }
44
45 static int createUdpSocket()
46 {
47     int sockfd;
48     int on = 1;
49
50     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
51     if(sockfd < 0)
52         return -1;
53
54     setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));
55
56     return sockfd;
57 }
58
59 static int bindSocketAddr(int sockfd, const char* ip, int port)
60 {
61     struct sockaddr_in addr;
62
63     addr.sin_family = AF_INET;
64     addr.sin_port = htons(port);
65     addr.sin_addr.s_addr = inet_addr(ip);
66
67     if(bind(sockfd, (struct sockaddr*)&addr, sizeof(struct sockaddr)) < 0)
68         return -1;
69
70     return 0;
71 }
72
73 static int acceptClient(int sockfd, char* ip, int* port)
74 {
75     int clientfd;
76     socklen_t len = 0;
77     struct sockaddr_in addr;
78
79     memset(&addr, 0, sizeof(addr));
80     len = sizeof(addr);
81
82     clientfd = accept(sockfd, (struct sockaddr*)&addr, &len);
83     if(clientfd < 0)
84         return -1;
85
86     strcpy(ip, inet_ntoa(addr.sin_addr));
87     *port = ntohs(addr.sin_port);
88
89     return clientfd;
90 }
91
92 static inline int startCode3(char* buf)
93 {
94     if(buf[0] == 0 && buf[1] == 0 && buf[2] == 1)
95         return 1;
96     else
97         return 0;
98 }
99
100 static inline int startCode4(char* buf)
101 {
102     if(buf[0] == 0 && buf[1] == 0 && buf[2] == 0 && buf[3] == 1)
103         return 1;
104     else
105         return 0;
106 }

```

```

    return 0;
}

static char* FindNextStartCode(char* buf, int len)
{
    int i;

    if(len < 3)
        return NULL;

    for(i = 0; i < len-3; ++i)
    {
        if(startCode3(buf) || startCode4(buf))
            return buf;

        ++buf;
    }

    if(startCode3(buf))
        return buf;

    return NULL;
}

static int getFrameFromH264File(int fd, char* frame, int size)
{
    int rSize, frameSize;
    char* nextStartCode;

    if(fd < 0)
        return fd;

    rSize = read(fd, frame, size);
    if(!startCode3(frame) && !startCode4(frame))
        return -1;

    nextStartCode = FindNextStartCode(frame+3, rSize-3);
    if(!nextStartCode)
    {
        lseek(fd, 0, SEEK_SET);
        frameSize = rSize;
    }
    else
    {
        frameSize = (nextStartCode - frame);
        lseek(fd, frameSize-rSize, SEEK_CUR);
    }

    return frameSize;
}

static int rtpSendH264Frame(int socket, const char* ip, int16_t port,
    struct RtpPacket* rtpPacket, uint8_t* frame, uint32_t frameSize)
{
    uint8_t naluType; // nalu第一个字节
    int sendBytes = 0;
    int ret;

    naluType = frame[0];

    if (frameSize <= RTP_MAX_PKT_SIZE) // nalu长度小于最大包长: 单一NALU单元模式
    {
        /*
         * 0 1 2 3 4 5 6 7 8 9
         * +-----+-----+-----+-----+
         * |F|NRI| Type | 0 single NAL unit ... |
         * +-----+-----+-----+-----+
         */
        memcpy(rtpPacket->payload, frame, frameSize);
        ret = rtpSendPacket(socket, ip, port, rtpPacket, frameSize);
        if(ret < 0)
            return -1;

        rtpPacket->rtpHeader.seq++;
        sendBytes += ret;
        if ((naluType & 0x1F) == 7 || (naluType & 0x1F) == 0) // 如果是SPS、PPS就不需要加时间戳
            goto out;
    }
    else // nalu长度大于最大包长: 分片模式
    {
        /*
         * 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
         * +-----+-----+-----+-----+-----+-----+
         * | FU indicator | FU header | FU payload ... |
         * +-----+-----+-----+-----+-----+
         */
        /*
         * FU Indicator
         * 0 1 2 3 4 5 6 7
         * +-----+-----+
         * |F|NRI| Type |
         * +-----+-----+
         */
        /*
         * FU Header
         * 0 1 2 3 4 5 6 7
         * +-----+-----+
         * |S|E|R| Type |
         * +-----+-----+
         */

        int pktNum = frameSize / RTP_MAX_PKT_SIZE; // 有几个完整的包
        int remainPktSize = frameSize % RTP_MAX_PKT_SIZE; // 剩余不完整包的大小
        int i, pos = 1;

        /* 发送完整的包 */
        for (i = 0; i < pktNum; i++)
        {
            rtpPacket->payload[0] = (naluType & 0x60) | 28;
            rtpPacket->payload[1] = naluType & 0x1F;

            if (i == 0) // 第一包数据
                rtpPacket->payload[1] |= 0x80; // start
            else if (remainPktSize == 0 && i == pktNum - 1) // 最后一包数据
                rtpPacket->payload[1] |= 0x40; // end

            memcpy(rtpPacket->payload+2, frame-pos, RTP_MAX_PKT_SIZE);
            ret = rtpSendPacket(socket, ip, port, rtpPacket, RTP_MAX_PKT_SIZE+2);
            if(ret < 0)
                return -1;

            rtpPacket->rtpHeader.seq++;
            sendBytes += ret;
            pos += RTP_MAX_PKT_SIZE;
        }

        /* 发送剩余的数据 */
        if (remainPktSize > 0)
        {
            rtpPacket->payload[0] = (naluType & 0x60) | 28;
            rtpPacket->payload[1] = naluType & 0x1F;
            rtpPacket->payload[1] |= 0x40; //end

            memcpy(rtpPacket->payload+2, frame-pos, remainPktSize+2);
            ret = rtpSendPacket(socket, ip, port, rtpPacket, remainPktSize+2);
            if(ret < 0)
                return -1;

            rtpPacket->rtpHeader.seq++;
            sendBytes += ret;
        }
    }

    out:
    return sendBytes;
}

static char* getlineFromBuf(char* buf, char* line)
{
    while(*buf != '\n')
    {
        *line = *buf;
        line++;
        buf++;
    }

    *line = '\n';
    ++line;
    *line = '\0';
}

```

```

268 +buf;
269 return buf;
270 }
271
272 static int handleCmd_OPTIONS(char* result, int cseq)
273 {
274     sprintf(result, "RTSP/1.0 200 OK\r\n"
275               "CSeq: %d\r\n"
276               "Public: OPTIONS, DESCRIBE, SETUP, PLAY\r\n"
277               "\r\n");
278     cseq);
279
280     return 0;
281 }
282
283 static int handleCmd_DESCRIBE(char* result, int cseq, char* url)
284 {
285     char sdp[500];
286     char localIp[100];
287
288     sscanf(url, "rtsp://%[^:]:", localIp);
289
290     sprintf(sdp, "v=0\r\n"
291               "o=- 5010 1 IN IP4 %s\r\n"
292               "t=0 0\r\n"
293               "a=control:* \r\n"
294               "a=type:broadcast \r\n"
295               "a=rtcp-unicast: reflection \r\n"
296               "a=video %d RTP/AVP 96 \r\n"
297               "c=IN IP4 %s/255 \r\n"
298               "a=rtmap:96 H264/90000 \r\n"
299               "a=control:track0 \r\n",
300             time(NULL),
301             localIp,
302             MULTICAST_PORT,
303             MULTICAST_IP);
304
305     sprintf(result, "RTSP/1.0 200 OK\r\nCSeq: %d\r\n"
306               "Content-Base: %s\r\n"
307               "Content-type: application/sdp\r\n"
308               "Content-length: %d\r\n\r\n"
309               "%s",
310             cseq,
311             url,
312             strlen(sdp),
313             sdp);
314
315     return 0;
316 }
317
318 static int handleCmd_SETUP(char* result, int cseq, char* localIp)
319 {
320     sprintf(result, "RTSP/1.0 200 OK\r\n"
321               "CSeq: %d\r\n"
322               "Transport: RTP/AVP;multicast;destination=%s;source=%s;port=%d-%d;ttl=255\r\n"
323               "Session: 6634873\r\n"
324               "\r\n",
325             cseq,
326             MULTICAST_IP,
327             localIp,
328             MULTICAST_PORT,
329             MULTICAST_PORT+1);
330
331     return 0;
332 }
333
334 static int handleCmd_PLAY(char* result, int cseq)
335 {
336     sprintf(result, "RTSP/1.0 200 OK\r\n"
337               "CSeq: %d\r\n"
338               "Range: npt=0.000- \r\n"
339               "Session: 6634873; timeout=60\r\n\r\n",
340             cseq);
341
342     return 0;
343 }
344
345 static void doClient(int clientSockfd, const char* clientIP, int clientPort)
346 {
347     char method[40];
348     char url[100];
349     char localIp[40];
350     char version[40];
351     int cseq;
352     char* bufPtr;
353     char* rBuf = malloc(BUF_MAX_SIZE);
354     char* sBuf = malloc(BUF_MAX_SIZE);
355     char line[400];
356
357     while(1)
358     {
359         int recvlen;
360
361         recvlen = recv(clientSockfd, rBuf, BUF_MAX_SIZE, 0);
362         if(recvlen <= 0)
363             goto out;
364
365         rBuf[recvlen] = '\0';
366         printf("-----C>S>-----\n");
367         printf("%s", rBuf);
368
369         /* 解析方法 */
370         bufPtr = getlineFromBuf(rBuf, line);
371         if(sscanf(line, "%s %s %s\r\n", method, url, version) != 3)
372         {
373             printf("parse err\n");
374             goto out;
375         }
376
377         /* 解析版本号 */
378         bufPtr = getlineFromBuf(bufPtr, line);
379         if(sscanf(line, "CSeq: %d\r\n", &cseq) != 1)
380         {
381             printf("parse err\n");
382             goto out;
383         }
384
385         if(!strcmp(method, "OPTIONS"))
386         {
387             if(handleCmd_OPTIONS(sBuf, cseq))
388             {
389                 printf("failed to handle options\n");
390                 goto out;
391             }
392         }
393         else if(!strcmp(method, "DESCRIBE"))
394         {
395             if(handleCmd_DESCRIBE(sBuf, cseq, url))
396             {
397                 printf("failed to handle describe\n");
398                 goto out;
399             }
400         }
401         else if(!strcmp(method, "SETUP"))
402         {
395             sscanf(url, "rtsp://%[^:]:", localIp);
396             if(handleCmd_SETUP(sBuf, cseq, localIp))
403             {
404                 printf("failed to handle setup\n");
405                 goto out;
406             }
407         }
408         else if(!strcmp(method, "PLAY"))
409         {
410             if(handleCmd_PLAY(sBuf, cseq))
411             {
412                 printf("failed to handle play\n");
413                 goto out;
414             }
415         }
416         else
417         {
418             goto out;
419         }
420     }
421
422     printf("-----S>C>-----\n");
423     printf("%s", sBuf);
424     send(clientSockfd, sBuf, strlen(sBuf), 0);
425 }
426
427 out:
428 printf("finish\n");
429 close(clientSockfd);
430 free(rBuf);
431 free(sBuf);

```

```

432 }
433
434 void* sendRtpPacket(void* arg)
435 {
436     int fd;
437     int frameSize, startCode;
438     char* frame = malloc(500000);
439     struct RtpPacket* rtpPacket = (struct RtpPacket*)malloc(500000);
440     int sockfd = createUdpSocket();
441     assert(sockfd > 0);
442
443     fd = open(H264_FILE_NAME, O_RDONLY);
444     assert(fd > 0);
445
446     rtpHeaderInit(&rtpPacket, 0, 0, 0, RTP_VERSION, RTP_PAYLOAD_TYPE_H264, 0,
447                 0, 0, 0xB8923423);
448
449     while(1)
450     {
451         frameSize = getFrameFromH264File(fd, frame, 500000);
452
453         if(startCode3(frame))
454             startCode = 3;
455         else
456             startCode = 4;
457
458         frameSize -= startCode;
459         rtpSendH264Frame(sockfd, MULTICAST_IP, MULTICAST_PORT,
460                        rtpPacket, frame+startCode, frameSize);
461         rtpPacket->rtpHeader.timestamp += 90000/25;
462
463         usleep(1000*1000/25);
464     }
465
466     free(frame);
467     free(rtpPacket);
468     close(fd);
469
470     return NULL;
471 }
472
473 int main(int argc, char* argv[])
474 {
475     int serverSockfd;
476     int ret;
477     pthread_t threadId;
478
479     serverSockfd = createTcpSocket();
480     if(serverSockfd < 0)
481     {
482         printf("failed to create tcp socket\n");
483         return -1;
484     }
485
486     ret = bindSocketAddr(serverSockfd, "0.0.0.0", SERVER_PORT);
487     if(ret < 0)
488     {
489         printf("failed to bind addr\n");
490         return -1;
491     }
492
493     ret = listen(serverSockfd, 10);
494     if(ret < 0)
495     {
496         printf("failed to listen\n");
497         return -1;
498     }
499
500     printf("rtsp://127.0.0.1:%d\n", SERVER_PORT);
501
502     pthread_create(&threadId, NULL, sendRtpPacket, NULL);
503
504     while(1)
505     {
506         int clientSockfd;
507         char clientIp[40];
508         int clientPort;
509
510         clientSockfd = acceptClient(serverSockfd, clientIp, &clientPort);
511         if(clientSockfd < 0)
512         {
513             printf("failed to accept client\n");
514             return -1;
515         }
516
517         printf("accept client;client ip:%s,client port:%d\n", clientIp, clientPort);
518
519         doClient(clientSockfd, clientIp, clientPort);
520     }
521
522     return 0;
523 }

```

rtp.h

```

1  /*
2   * 作者: _3T_
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

```

rtp.c

```

1  /*
2   * 作者: _3T_
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 "rtp.h"
13
14 void rtpHeaderInit(struct RtpPacket* rtpPacket, uint8_t csrLen, uint8_t extension,
15                  uint8_t padding, uint8_t version, uint8_t payloadType, uint8_t marker,
16                  uint16_t seq, uint32_t timestamp, uint32_t ssrc)
17 {
18     rtpPacket->rtpHeader.csrLen = csrLen;
19     rtpPacket->rtpHeader.extension = extension;
20     rtpPacket->rtpHeader.padding = padding;
21     rtpPacket->rtpHeader.version = version;
22     rtpPacket->rtpHeader.payloadType = payloadType;
23     rtpPacket->rtpHeader.marker = marker;
24     rtpPacket->rtpHeader.seq = seq;
25     rtpPacket->rtpHeader.timestamp = timestamp;
26     rtpPacket->rtpHeader.ssrc = ssrc;
27 }
28
29 int rtpSendPacket(int socket, const char* ip, int16_t port, struct RtpPacket* rtpPacket, uint32_t dataSize)
30 {
31     struct sockaddr_in addr;
32     int ret;
33
34     addr.sin_family = AF_INET;
35     addr.sin_port = htons(port);
36     addr.sin_addr.s_addr = inet_addr(ip);
37
38     rtpPacket->rtpHeader.seq = htons(rtpPacket->rtpHeader.seq);
39     rtpPacket->rtpHeader.timestamp = htonl(rtpPacket->rtpHeader.timestamp);
40     rtpPacket->rtpHeader.ssrc = htonl(rtpPacket->rtpHeader.ssrc);
41
42     ret = sendto(socket, (void*)rtpPacket, dataSize+RTP_HEADER_SIZE, 0,
43                (struct sockaddr*)&addr, sizeof(addr));
44
45     rtpPacket->rtpHeader.seq = ntohs(rtpPacket->rtpHeader.seq);
46     rtpPacket->rtpHeader.timestamp = ntohl(rtpPacket->rtpHeader.timestamp);
47     rtpPacket->rtpHeader.ssrc = ntohl(rtpPacket->rtpHeader.ssrc);
48
49     return ret;
50 }

```

JT同学

粉丝4年

暂无认证

105

2万+

91万+

46万+

原创

周排名

总排名

访问

等级

5440

1861

1016

346

3945

积分

粉丝

获赞

评论

收藏

关注

私信

搜索

搜索

热门文章

最新评论

您愿意向朋友推荐“博客详情”吗？

最新文章

热门文章

从零开始写一个RTSP服务器 (一) RTSP协议讲解

57178

C++ deque的用法与示例

38014

Linux I2C驱动框架 (超详细)

35491

我的开源项目-RtspServer

25563

从零开始写一个RTSP服务器 (二) RTSP协议的实现

21864

最新评论

我的开源项目-RtspServer

kuaf3k: 我测试\_h264\_rtsp\_server test\_h264 也会这样的问题? 请问解决了没? m...

使用mp4v2封装H.264&mp4最简单示例

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&mp4最简单示例

ai604233436: h264的标准中, 好像默认的设置90KHz, 我所以理解timeScale...

使用mp4v2封装H.264&mp4最简单示例

ai604233436: 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: 请问往缓存区里面采集图像数据的帧率是多少呢? 请问如果缓存区里...

您愿意向朋友推荐“博客详情”吗?

强烈推荐

推荐

一般般

不推荐

强烈不推荐

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

### 三、测试

将multicast\_rtsp\_server.c、rtsp.h、rtsp.c保存下来

编译运行, 程序默认会打开test\_h264的视频文件, 如果你没有视频源的话, 可以从RtspServer的example目录下获取

```
1 # gcc multicast_rtsp_server.c rtsp.c -lpthread
2 # ./a.out
```

运行后会打印一个url

```
1 | rtsp://127.0.0.1:8554
```

在Vc中输入url, 就可以播放了

JT同学

关注

7

16

11

专栏目录

NVRServer: 一个简单的rtsp服务器, 可以使用rtsp:从摄像头提取流, 将视频另存为hls流, 并为rtsp服务器提供实时流和本地... 02-05

NVRServer: 一个简单的rtsp服务器, 可以使用rtsp:从摄像头提取流, 将视频另存为hls流, 并为rtsp服务器提供实时流和本地...

2149

从零开始用C语言写一个简单的rtsp转发服务器, 支持H264,H265,流支持UDP和TCP

jack\_wu199202的博文

2149

最近在做一个rtsp的流媒体转发服务器, 看了ffmpeg, Gstreamer和live555, 对于我的目标都不方便移植, 所以从头写了一个简单的, 网上找了不少资料...

2414

C++实现RTSP/RTMP服务器

cr66的博文

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, 这篇文章我们来讲解如何用c++自己写一个简单的最基本的rtsp服...

274

RTSP协议详解

sinet\_360020555的博文

274

参考: 手撕RTSP协议系列 Rtsp基本流程 rtsp协议简介 rtsp, 英文全称 Real Time Streaming Protocol, RFC3236, 实时流传输协议, 是TCP/IP协议体系...

2501

RTSP详解

(1) ...

RTSP的功能: rtsp并不传输数据, 其作用相当于流媒体服务器的远程控制, 传输数据可以通过传输层的tcp和udp协议. rtsp和tcp的区别和联系: ...

4844

rtsp 服务器搭建

ymshqin007的博文

4844

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...

4709

openRTSP 使用

occupy的专栏

4709

from: 1: 编译 (1): ./genMakefiles linux (2): make 2: 服务器环境 服务器端是一个支持RTSP server的H.264的摄像头; 3: 运行 openRTSP ...

2247

linux搭建rtsp服务端,手把手搭建RTSP流媒体服务器

weixin\_32632631的博文

2247

0.引言本文主要讲解如何搭建RTSP流媒体服务器的过程, 使用开源项目ZLMediaKit, 通过这个开源项目, 推RTSP流到服务器, 然后拉流端可以拉取RTS...

751

rtsp服务器: C语言实现rtsp服务器 最新发布

xiaoning132的博文

751

class RtspServer { public: RtspServer(void); ~RtspServer(void); public: int start\_server(unsigned short port); /\*RTSP OVER HTTP\*/ int http\_cmd\_serve...

804

7、多播的RTSP服务器

huabiaoche的博文

804

本文目的: 实现一个多播传输H.264的RTSP服务器 一、多播的RTSP交互过程 我们在前面文章从零开始写一个RTSP服务器 (二) RTSP协议的实现中实...

7263

实时协议—RTSP【详解】

王木木

7263

一、RTSP协议介绍 RTSP (Real Time Streaming Protocol 实时流协议) 是一个有效地在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...

267

TSINGSEE青犀视频云边端H265播放器EasyPlayer-RTSP在C#版本增加OOSD功能说明

TSINGSEE青犀视频官方技术博文

267

EasyPlayer播放器项目是TSINGSEE青犀视频研发团队开发的H265编码视频播放器, 经过多年的技术积累与实践打造, EasyPlayer播放器项目系列无论是...

874

RTSP视频流显示(海康威视) 热门推荐

会飞的蜗牛

874

RTSP视频流显示(海康威视) VLCSDK (C++) ffmpeg+Nginx 本文目的是想要在eml上实时显示海康威视的摄像头数据, 笔者尝试了如下三种方式: ...

65-30

基于C#的RTSP客户端编解码程序

05-30

实现了简单的RTSP的客户端命令功能, 可以连接darwin服务器, 并进行交互, 实现了Options.Describ.Setup.Play.beardown命令...

266

【技术教程】C#控制台调用FFMPEG推MP4视频文件至流媒体开源服务平台EasyDarwin过程

Black

266

之前写过一篇科普文《如何使用RTSP流组件EasyPusher推MP4文件推送到EasyDarwin开源平台》, 在该文中, 我们尝试了通过EasyPusher将MP4文件...

3686

Rtsp-服务器搭建 (Ffmpeg+Node.js+ffmpeg网络视频服务器)

wtnu2000的博文

3686

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插件开发包, 结果很坑, 控件, 要求为x64架构...

6652

rtsp组播搭建和rtsp组播实现

ldwenny6的博文

6652

rtsp组播在一些场景下比单播更合适, 比如电子教室等, 单播每一路都要占相同带宽, 带宽要求比较高, 并且路数多了也容易丢包, 组播只占一路带宽, ...

1506

个人对rtsp协议的理解

Daner13921的博文

1506

由于业务需求, 要实现rtsp协议下的视频流代理功能, 此前对rtsp协议不了解, 搜索了很多关于rtsp的文章, 一点点的弄明白了, 现在项目开发完成了, 打...

"相关推荐"对你有帮助?

非常有帮助

有帮助

一般般

没帮助

非常没帮助

2022 CSDN 皮肤主题: 数字20 设计师: CSDN官方博文 返回首页

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 国家反诈中心 家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理 版权投诉 侵权申诉 营业执照 1999-2022 北京创新乐知网络技术有限公司