JT同学
码龄4年 暂无认证
105 原创 | 2万+ 周排名 | 91万+ 总排名 | 46万+ 访问 | 等级
5440 积分 | 1861 粉丝 | 1016 获赞 | 346 评论 | 3945 收藏
私信 关注

搜博主文章

热门文章
从零开始写一个RTSP服务器（一）RTSP协议讲解 57162
C++ deque的用法与示例 38004
Linux I2C驱动框架（超详细） 35491
我的开源项目-RtspServer 25554
从零开始写一个RTSP服务器（二）RTSP协议的实现 21860

最新评论
我的开源项目-RtspServer
kaifa3k: 我测试 ./h264_rtsp_server test.h264 也会这样的问题？请问解决了吗？m...
使用mp4v2封装H.264成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成mp4最简单示例
ai604233436: h264的标准中，好像默认的设置的90KHZ。我所以我理解timeScale...
使用mp4v2封装H.264成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...

您愿意向朋友推荐“博客详情页”吗？
强烈不推荐 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（二）RTSP协议的实现

原创 JT同学 于 2019-08-10 12:58:58 发布 21863 收藏 145 版权
分类专栏： 从零开始写一个RTSP服务器 文章标签： rtsp rtp 音视频 流媒体

从零开始写一个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-RtspServer

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（二）RTSP协议的实现

文章目录

写在前面

此系列只追求精简，旨在学习RTSP协议的实现过程，不追求复杂完美，所以这里要实现的RTSP服务器为了简单，实现上同一时间只能有一个客户端，下面开始介绍实现过程

在写一个RTSP服务器之前，我们必须知道一个RTSP服务器最简单的包含两部分，一部分是RTSP的交互，一部分是RTP发送，本文先实现RTSP交互过程

一、创建套接字

想一下我们在vlc输入 `rtsp://127.0.0.1:8554` 后发生了什么事？

在这种情况下，vlc其实是一个rtsp客户端，当输入这个url后，vlc知道目的IP为 `127.0.0.1` ，目的端口号为 `8854` ，这时vlc会发起一个tcp连接取连接服务器，连接成功后就开始发送请求，服务端响应

所以我们要写一个rtsp服务器，第一步肯定是创建tcp服务器

首先创建tcp套接字，绑定端口，监听

创建套接字

```
/*
 * 作者: _JT_
 * 博客: https://blog.csdn.net/weixin_42462202
 */
serverSockfd = socket(AF_INET, SOCK_STREAM, 0);
setsockopt(serverSockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));
```

绑定地址和端口号

```
bind(serverSockfd, (struct sockaddr *)&addr, sizeof(struct sockaddr)
```

这个示例绑定的地址是 `INADDR_ANY` ，端口号为 `8554`

开始监听

```
listen(serverSockfd, 10);
```

RTSP服务器传输音视频数据和信息使用的是RTP和RTCP，所以我们还要为RTP和RTCP创建UDP套接字，并绑定号端口

创建套接字

```
serverRtpSockfd = createUdpSocket();
serverRtcpSockfd = createUdpSocket();
```

绑定端口号

```
bindSocketAddr(serverRtpSockfd, "0.0.0.0", SERVER_RTP_PORT);
bindSocketAddr(serverRtcpSockfd, "0.0.0.0", SERVER_RTCP_PORT);
```

当创建好套接字还有绑定号端口后，就可以接收客户端请求了

开始accept等待客户端连接

```
clientfd = accept(serverSockfd, (struct sockaddr *)&addr, &len);
```

二、解析请求

当rtsp客户端连接成功后就会开始发送请求，服务器这是需要接收客户端请求并开始解析，再采取相应得操作

请求的格式为（详细参考上一篇从零开始写一个RTSP服务器（一）不一样的RTSP协议讲解）

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

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

```
SETUP rtsp://127.0.0.1:8554/live/track0 RTSP/1.0\r\n
CSeq: 4\r\n
Transport: RTP/AVP;unicast;client_port=54492-54493\r\n
\r\n
```

```
PLAY rtsp://127.0.0.1:8554/live RTSP/1.0\r\n
CSeq: 5\r\n
Session: 66334873\r\n
Range: npt=0.000-\r\n
\r\n
```

这里我们做得最简单，首先解析 `第一行` 得到 `方法` ，对于 `OPTIONS` 、 `DESCRIBE` 、 `PLAY` 、 `TEARDOWN` 我们只解析 `CSeq` 。对于 `SETUP` ，我们讲 `client_port` 解析出来

所以我们要做的第一步就是解析请求中的信息

接收客户端数据

```
recvLen = recv(clientSockfd, rBuf, BUF_MAX_SIZE, 0);
```

这里实现了一个简单得函数 `getLineFromBuf` ，从buf中读取一行(\r\n)

解析第一行请求得到方法

```
sscanf(line, "%s %s %s\r\n", method, url, version);
```

其次解析 `CSeq`

```
sscanf(line, "CSeq: %d\r\n", &cseq)
```

目录
从零开始写一个RTSP服务器（二）RTS...
文章目录
写在前面
一、创建套接字
二、解析请求
三、OPTIONS响应
四、DESCRIBE响应
五、SETUP响应
六、PLAY响应
七、源码
八、测试

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

如果方法是`SETUP`则再解析`client_port`

```
if(!strcmp(method, "SETUP"))
{
 sscanf(line, "Transport: RTP/AVP;unicast;client_port=%d-%d\r\n",
   &clientRtpPort, &clientRtcpPort);
}
```

解析完请求命令后，接下来就是更具不同得方法做不同的响应了，如下

```
/*
 * 作者: _JT_
 * 博客: https://blog.csdn.net/weixin_42462202
 */
if(!strcmp(method, "OPTIONS"))
{
    handleCmd_OPTIONS();
}
else if(!strcmp(method, "DESCRIBE"))
{
    handleCmd_DESCRIBE();
}
else if(!strcmp(method, "SETUP"))
{
    handleCmd_SETUP();
}
else if(!strcmp(method, "PLAY"))
{
    handleCmd_PLAY();
}
else if(!strcmp(method, "TEARDOWN"))
{
    handleCmd_TEARDOWN();
}
```

三、OPTIONS响应

OPTIONS是客户端向服务端请求可用的方法，我们这里就向客户端回复我们当前可用的方法

```
sprintf(sBuf, "RTSP/1.0 200 OK\r\n"
      "CSeq: %d\r\n"
    "Public: OPTIONS, DESCRIBE, SETUP, PLAY\r\n"
       "\r\n",
        cseq);
send(clientSockfd, sBuf, strlen(sBuf));
```

四、DESCRIBE响应

DESCRIBE是客户端向服务器请求媒体信息，这是服务器需要回复sdp描述文件，这个例子中的媒体是H.264

sdp文件生成

```
sprintf(sdp, "v=0\r\n"
    "o=- 9%ld 1 IN IP4 %s\r\n"
     "t=0 0\r\n"
    "a=control:*\r\n"
    "m=video 0 RTP/AVP 96\r\n"
    "a=rtpmap:96 H264/90000\r\n"
    "a=control:track0\r\n",
    time(NULL), localIp);
```

回复

```
sprintf(sBuf, "RTSP/1.0 200 OK\r\n"
   "CSeq: %d\r\n"
   "Content-Base: %s\r\n"
   "Content-type: application/sdp\r\n"
   "Content-length: %d\r\n\r\n"
    "%s",
    cseq,
    url,
   strlen(sdp),
    sdp);

send(clientSockfd, sBuf, strlen(sBuf));
```

五、SETUP响应

SETUP是客户端请求建立会话连接，并发送了客户端的RTP端口和RTCP端口，那么此时服务端需要回复服务端的RTP端口和RTCP端口

回复

```
sprintf(result, "RTSP/1.0 200 OK\r\n"
    "CSeq: %d\r\n"
   "Transport: RTP/AVP;unicast;client_port=%d-%d;server_port=%d-%d\r\n"
    "Session: 66334873\r\n"
     "\r\n",
      cseq,
    clientRtpPort,
    clientRtpPort+1,
    SERVER_RTP_PORT,
    SERVER_RTCP_PORT);

send(clientSockfd, sBuf, strlen(sBuf));
```

其中session id是随便写的，只要保证在多个会话连接时唯一的就行

play响应之后就可以向客户端的RTP端口发送RTP包了

六、PLAY响应

PLAY时客户端向服务器请求播放，这时服务端回复完请求后就开始通过setup过程中创建的udp套接字发送RTP包

回复

```
sprintf(result, "RTSP/1.0 200 OK\r\n"
      "CSeq: %d\r\n"
     "Range: npt=0.000-\r\n"
    "Session: 66334873; timeout=60\r\n\r\n",
        cseq);

send(clientSockfd, sBuf, strlen(sBuf));
```

开始发送数据

回复之后，就开始向客户端指定的RTP端口发送RTP包，如何发送RTP包，下篇文章再介绍

七、源码

```
/*
 * 作者: _JT_
 * 博客: https://blog.csdn.net/weixin_42462202
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>

#define SERVER_PORT     8554
#define SERVER_RTP_PORT  55532
#define SERVER_RTCP_PORT 55533
#define BUF_MAX_SIZE    (1024*1024)

static int createTcpSocket()
{
    int sockfd;
    int on = 1;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd < 0)
        return -1;

    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));

    return sockfd;
}

```

```c
static int createUdpSocket()
{
    int sockfd;
    int on = 1;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if(sockfd < 0)
        return -1;

    setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (const char*)&on, sizeof(on));

    return sockfd;
}

static int bindSocketAddr(int sockfd, const char* ip, int port)
{
    struct sockaddr_in addr;

    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = inet_addr(ip);

    if(bind(sockfd, (struct sockaddr *)&addr, sizeof(struct sockaddr)) < 0)
        return -1;

    return 0;
}

static int acceptClient(int sockfd, char* ip, int* port)
{
    int clientfd;
    socklen_t len = 0;
    struct sockaddr_in addr;

    memset(&addr, 0, sizeof(addr));
    len = sizeof(addr);

    clientfd = accept(sockfd, (struct sockaddr *)&addr, &len);
    if(clientfd < 0)
        return -1;

    strcpy(ip, inet_ntoa(addr.sin_addr));
    *port = ntohs(addr.sin_port);

    return clientfd;
}

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

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

    ++buf;
    return buf;
}

static int handleCmd_OPTIONS(char* result, int cseq)
{
    sprintf(result, "RTSP/1.0 200 OK\r\n"
                    "CSeq: %d\r\n"
                    "Public: OPTIONS, DESCRIBE, SETUP, PLAY\r\n"
                    "\r\n",
                    cseq);

    return 0;
}

static int handleCmd_DESCRIBE(char* result, int cseq, char* url)
{
    char sdp[500];
    char localIp[100];

    sscanf(url, "rtsp://%[^:]:", localIp);

    sprintf(sdp, "v=0\r\n"
                 "o=- 9%ld 1 IN IP4 %s\r\n"
                 "t=0 0\r\n"
                 "a=control:*\r\n"
                 "m=video 0 RTP/AVP 96\r\n"
                 "a=rtpmap:96 H264/90000\r\n"
                 "a=control:track0\r\n",
                 time(NULL), localIp);

    sprintf(result, "RTSP/1.0 200 OK\r\nCSeq: %d\r\n"
                    "Content-Base: %s\r\n"
                    "Content-type: application/sdp\r\n"
                    "Content-length: %d\r\n\r\n"
                    "%s",
                    cseq,
                    url,
                    strlen(sdp),
                    sdp);

    return 0;
}

static int handleCmd_SETUP(char* result, int cseq, int clientRtpPort)
{
    sprintf(result, "RTSP/1.0 200 OK\r\n"
                    "CSeq: %d\r\n"
                    "Transport: RTP/AVP;unicast;client_port=%d-%d;server_port=%d-%d\r\n"
                    "Session: 66334873\r\n"
                    "\r\n",
                    cseq,
                    clientRtpPort,
                    clientRtpPort+1,
                    SERVER_RTP_PORT,
                    SERVER_RTCP_PORT);

    return 0;
}

static int handleCmd_PLAY(char* result, int cseq)
{
    sprintf(result, "RTSP/1.0 200 OK\r\n"
                    "CSeq: %d\r\n"
                    "Range: npt=0.000-\r\n"
                    "Session: 66334873; timeout=60\r\n\r\n",
                    cseq);

    return 0;
}

static void doClient(int clientSockfd, const char* clientIP, int clientPort,
                     int serverRtpSockfd, int serverRtcpSockfd)
{
    char method[40];
    char url[100];
    char version[40];
    int cseq;
    int clientRtpPort, clientRtcpPort;
    char *bufPtr;
    char* rBuf = malloc(BUF_MAX_SIZE);
    char* sBuf = malloc(BUF_MAX_SIZE);
    char line[400];

    while(1)
    {
        int recvLen;

        recvLen = recv(clientSockfd, rBuf, BUF_MAX_SIZE, 0);
        if(recvLen <= 0)
            goto out;

        rBuf[recvLen] = '\0';
        printf("---------------C->S-------------\n");
        printf("%s", rBuf);

        /* 解析方法 */
        bufPtr = getLineFromBuf(rBuf, line);
        if(sscanf(line, "%s %s %s\r\n", method, url, version) != 3)
        {
            printf("parse err\n");
            goto out;
        }

```

```
        /* 解析序列号 */
        bufPtr = getLineFromBuf(bufPtr, line);
        if(sscanf(line, "CSeq: %d\r\n", &cseq) != 1)
        {
            printf("parse err\n");
            goto out;
        }

        /* 如果是SETUP，那么就再解析client_port */
        if(!strcmp(method, "SETUP"))
        {
            while(1)
            {
                bufPtr = getLineFromBuf(bufPtr, line);
                if(!strncmp(line, "Transport:", strlen("Transport:")))
                {
                    sscanf(line, "Transport: RTP/AVP;unicast;client_port=%d-%d\r\n",
                                &clientRtpPort, &clientRtcpPort);
                    break;
                }
            }
        }

        if(!strcmp(method, "OPTIONS"))
        {
            if(handleCmd_OPTIONS(sBuf, cseq))
            {
                printf("failed to handle options\n");
                goto out;
            }
        }
        else if(!strcmp(method, "DESCRIBE"))
        {
            if(handleCmd_DESCRIBE(sBuf, cseq, url))
            {
                printf("failed to handle describe\n");
                goto out;
            }
        }
        else if(!strcmp(method, "SETUP"))
        {
            if(handleCmd_SETUP(sBuf, cseq, clientRtpPort))
            {
                printf("failed to handle setup\n");
                goto out;
            }
        }
        else if(!strcmp(method, "PLAY"))
        {
            if(handleCmd_PLAY(sBuf, cseq))
            {
                printf("failed to handle play\n");
                goto out;
            }
        }
        else
        {
            goto out;
        }

        printf("---------------S->C--------------\n");
        printf("%s", sBuf);
        send(clientSockfd, sBuf, strlen(sBuf), 0);
    }
out:
    close(clientSockfd);
    free(rBuf);
    free(sBuf);
}

int main(int argc, char* argv[])
{
    int serverSockfd;
    int serverRtpSockfd, serverRtcpSockfd;
    int ret;

    serverSockfd = createTcpSocket();
    if(serverSockfd < 0)
    {
        printf("failed to create tcp socket\n");
        return -1;
    }

    ret = bindSocketAddr(serverSockfd, "0.0.0.0", SERVER_PORT);
    if(ret < 0)
    {
        printf("failed to bind addr\n");
        return -1;
    }

    ret = listen(serverSockfd, 10);
    if(ret < 0)
    {
        printf("failed to listen\n");
        return -1;
    }

    serverRtpSockfd = createUdpSocket();
    serverRtcpSockfd = createUdpSocket();
    if(serverRtpSockfd < 0 || serverRtcpSockfd < 0)
    {
        printf("failed to create udp socket\n");
        return -1;
    }

    if(bindSocketAddr(serverRtpSockfd, "0.0.0.0", SERVER_RTP_PORT) < 0 ||
        bindSocketAddr(serverRtcpSockfd, "0.0.0.0", SERVER_RTCP_PORT) < 0)
    {
        printf("failed to bind addr\n");
        return -1;
    }

    printf("rtsp://127.0.0.1:%d\n", SERVER_PORT);

    while(1)
    {
        int clientSockfd;
        char clientIp[40];
        int clientPort;

        clientSockfd = acceptClient(serverSockfd, clientIp, &clientPort);
        if(clientSockfd < 0)
        {
            printf("failed to accept client\n");
            return -1;
        }

        printf("accept client;client ip:%s,client port:%d\n", clientIp, clientPort);

        doClient(clientSockfd, clientIp, clientPort, serverRtpSockfd, serverRtcpSockfd);
    }

    return 0;
}
```

八、测试

编译运行源码，打开vlc，输入`rtsp://127.0.0.1:8554`，点击开始播放，可以看到控制台会打印出交互过程，或是用wireshak抓包

本篇文章到这里结束，至此完成了RTSP协议的交互部分，在PLAY之后并没有开始发送RTP包，所以暂时还看不到视频，究竟如何发送RTP包，请看下一篇文章

★我的开源项目-RtspServer 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…

rtsp协议_rtsp开源服务器之live555 weixin_39955700的博客 92
Live555是一个实现了RTSP协议的开源流媒体框架，Live555包含RTSP服务器端的实现以及RTSP客户端的实现。Live555可以将若干种格式的视频文件或…

RTSP服务器：RTSP协议实现（与vlc交互） weixin_43796767的博客 418
RTSP协议介绍，请参考：RTSP传输流过程分析 协议实现过程： 1）创建TCP连接，与VLC客户端连接。工作流程：创建tcp套接字，绑定端口，监听 2…

从头写rtsp服务器-RTSP协议的解析_achorw4969的博客 10-11
1.首先服务器收到客户端连接请求,生产 一个RtspClientConnection对象,RtspClientConnection定义详见从头写rtsp服务器-模块的划分 intrtsp::v_accept(ne…

从零开始写一个RTSP服务器（一）RTSP协议讲解 热门推荐 JT同学的博客 5万+
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)…

RTSP协议的实现 一叶知秋 603
写在前面 目前正在学习RTSP协议，偶然间发现在这篇文章非常好，故转载学习使用：RTSP协议的实现 本文所有代码均来自此篇文章，在此基础上做了…

一个RtspServer的设计与实现和RTSP2.0简介 weixin_33824363的博客 267
一个RtspServer的设计与实现和RTSP2.0简介 前段时间着手实现了一个RTSP Server，能够正常实现多路RTSP流的直播播放，因项目需要，只做了…

Fast-RTPS, RTPS标准的实现.zip 09-18
Fast-RTPS, RTPS标准的实现 eProsima快速 RTPS system - config - 快速 RTPS 是 RTPS (实时发布订阅服务器) 协议的一个 C 实现，它通过对象管理组…

流媒体开发之开源项目rtsp服务器—何工的eseay rtsp服务器 weixin_33845881的博客 87
http://www.easyrtsp.org/ 转载于:https://www.cnblogs.com/pengkunfan/p/4360772.html

搭建RTSP流媒体服务器的三种方式 fang.lovest.yang的博客 4339
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开源库测试 程序猿的专栏 677
前2年搞过一个C#版RTSPServer项目，原项目中使用EasyIPCamera，这个项目目前已经改名为EasyRTSPServer，提供了一套完整的视频推送方案，这…

从零开始写一个RTSP服务器（五）RTP传输AAC JT同学的博客 8058
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…

音视频学习之rtsp推流学习1（rtspserver开源库example运行及流程梳理） yun6853992的博客 1740
工作需要实现一个rtsp的推流拉流业务流程，对开源项目rtspserver进行学习及理解。 参考系列rtspserver的文章：我的开源项目-RtspServer_JT同学的博…

RTCP协议介绍 无自制，不成功 4000

RTCP概要
实时传输控制协议(R eal-t ime C ontrol P rotocol ， RTCP) 与RTP 共同定义在1996 年提出的RFC 1889 中，是和 RTP …

rtsp服务器: C语言实现rtsp服务器 最新发布 xiaoning132的专栏 751
class RtspServer { public: RtspServer(void); ~RtspServer(void); public: int start_server(unsigned short port); /*RTSP OVER HTTP*/ int http_cmd_serve…

开源RTSP 流媒体服务器 coder 1万+
Columbia University rtspd - Supports serving of .au files via RTSP for Solaris, FreeBSD 3.x, Linux, Windows NT. Darwin Streaming Server - The open-s…

我的开源项目-RtspServer JT同学的博客 2万+
我的开源项目-RtspServer 文章目录我的开源项目-RtspServer一、项目介绍二、功能介绍三、开发环境四、使用方法4.1 传输音视频文件4.2 采集V4L2摄…

开源的rtsp实现 weixin_33924220的博客 437
开源的rtsp实现 ========== -- by BeagleTam 2006-5-4 1:40 最近在做rt…

“相关推荐”对你有帮助么？

非常没帮助 没帮助 一般 有帮助 非常有帮助