DoubleLi
qq: 517712484 wx: ldbgliet

2021年7月

日	一	二	三	四	五	六
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

公告

昵称：DoubleLi
园龄：11年6个月
粉丝：2053
关注：29
+加关注

搜索

找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类 (5113)
android(2)
ASP.NET(30)
ASP.NET MVC(11)
Boost(118)
c#(9)
C++/C(773)
c++11(15)
cmake/autotool(62)
com/ATL/Activex(75)
Css(16)
CxImage(12)
darwin stream server(3)
DataBase(29)
DirectX(16)
Extjs(13)
更多

随笔档案 (3811)
2021年7月(40)
2021年6月(1)
2021年5月(29)
2021年4月(15)
2021年3月(13)
2021年2月(96)
2021年1月(47)
2020年12月(2)
2020年11月(27)
2020年10月(44)
2020年9月(14)
2020年8月(4)
2020年7月(18)
2020年6月(5)
2020年5月(2)
更多

文章分类 (2)
SilverLight(1)
sql server(1)

参考博客
com
morewindows
关注DirectX
chenyujing1234
个人开发历程知识库
opencv教程
大坡3D软件开发
Dean Chen的专栏
Sloan
音视频FFmpeg等
回忆未来-向东
Nginx模块开发与原理剖析
linux驱动
webrtc参考一
ffmpeg参考
更多

最新评论
1. Re:RTSP协议详解
@陶子 用vlc是可以播放的，是直接获取海康设备的rtsp的地址...
--cosine-x
2. Re:RTSP协议详解
@cosine-x 你的rtsp要能够用vlc播放器播放...
--陶子
3. Re:RTSP协议详解
@cosine-x ip和端口对吗...
--陶子
4. Re:RTSP协议详解
@陶子 sk.isConnectable()为false，sk.isReadable()也为false，这个问题请问你是怎么的结局的，我也遇到了这个问题...
--cosine-x
5. Re:Git 进阶 - 获取 fetch
大佬，流程图中的两个push是不是手误？应该是pull？
--临渊久渡

阅读排行榜
1. Nginx之location 匹配规则详解(234821)
2. cmake使用方法详解(167335)
3. MinGW安装和使用(100038)
4. RTMP、RTSP、HTTP视频协议详解（附：直播流地址、播放软件）(96061)
5. C语言字符串操作总结大全(超详细)(90955)

评论排行榜
1. 非IE内核浏览器支持activex插件(37)
2. Nginx之location 匹配规则详解(19)
3. Javascript中定义类(15)
4. C++中的头文件和源文件(9)
5. RTSP协议详解(8)

推荐排行榜
1. C++中的头文件和源文件(25)
2. Nginx之location 匹配规则详解(22)
3. Javascript中定义类(12)
4. JavaScript中typeof知多少？(11)
5. MinGW安装和使用(9)

RTSP协议详解

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。而且允许同时多个串流需求控制（Multicast），除了可以降低服务器端的网络用量，还可以支持多方视频会议（Video onference）。 因为与HTTP1.1的运作方式相似，所以代理服务器《Proxy》的快取功能《Cache》也同样适用于RTSP，并因RTSP具有重新导向功能，可视实际负载情况来转换提供服务的服务器，以避免过大的负载集中于同一服务器而造成延迟。

---

rtsp和http的区别和联系

（1）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 。

（2）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去断掉连接。 ，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的。rtsp协议使用554端口，http使用80端口。

---

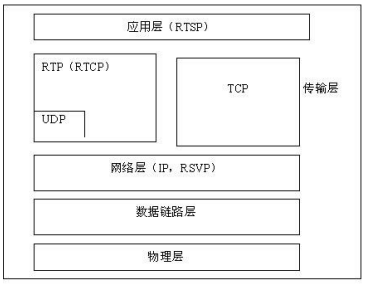
rtsp和sip的区别和联系

SIP（Session Initiation Protocol），是基于IP的一个应用层控制协议。由于SIP是基于纯文本的信令协议，可以管理不同接入网络上的会话等。会话可以是终端设备之间任何类型的通信，如视频会话、既时信息处理或协作会话。该协议不会定义或限制可使用的业务，传输、服务质量、计费、安全性等问题都由基本核心网络和其它协议处理。

（1）联系：sip和rtsp都是应用层的控制协议，负责一次通信过程的建立和控制和结束，不负责中间的传输部分。他们都是基于纯文本的信令协议，穿墙性能良好。支持tcp、udp，支持多方通信。他们都需要服务器支持，都支持会话中重定向。sip和rtsp 都使用sdp协议来传送媒体参数，使用rtp（rtcp）协议来传输媒体流。

（2）区别：rtsp是专门为流媒体制定的协议，在多个媒体流的时间同步方面比sip强大。rtsp还提供网络负载均衡的功能，减轻服务器压力和网络带宽要求。sip一般用来创建一次音频、视频通话（双向），而rtsp一般用来做视频点播、视频监控等（单向）。当然，从原理上讲，rtsp也可以做双向的视频通话。

RTSP和RTP（rtcp）的关系



rtsp负责建立和控制会话，rtp负责多媒体的传输，rtcp配合rtp做控制和流量统计，他们是合作的关系。

RTSP的消息

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

其中方法包括OPTIONS、SETUP、PLAY、TEARDOWN等待，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，最后一个消息头需要有两个CR LF。

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。

状态码由三位数组成，表示方法执行的结果，定义如下：

1XX：保留，将来使用；

2XX：成功，操作被接收、理解、接受（received,understand,accepted）；

3XX：重定向，要完成操作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：服务器出错，服务器无法实现合法的请求。

RTSP的方法

rtsp中定义的方法有：OPTIONS, DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, SCALE, GET_PARAMETER ，SET_PARAMETER

1.OPTION
目的是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

2.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:trackID=0//trackID = 0表示视频流用的是通道0

3.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)
//uri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interleaved，它的值是每个通道都不同的，trackID = 0的interleaved值有两个0或1，0表示rtp包，1表示rtcp包，接受端根据interleaved的值来区别是哪种

数据包。
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=6B8B4567

4.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

5.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
Connection: Close
以上方法都是交互过程中最为常用的,其它还有一些重要的方法如get/set_parameter,pause,redirect等等

ps:
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）

RTSP客户端的JAVA实现

3.1 接口IEvent.java

接口IEvent.java的代码如下：

javaf代码

```java
package com.amigo.rtsp;

import java.io.IOException;
import java.nio.channels.SelectionKey;

/** *//**
* IEvent.java 网络事件处理器，当Selector可以进行操作时，调用这个接口中的方法。
* 2007-3-22 下午03:35:51
* @author sycheng
* @version 1.0
*/
public interface IEvent {
    /** *//**
    * 当channel得到connect事件时调用这个方法。
    * @param key
    * @throws IOException
    */
    void connect(SelectionKey key) throws IOException;

    /** *//**
    * 当channel可读时调用这个方法。
    * @param key
    * @throws IOException
    */
    void read(SelectionKey key) throws IOException;

    /** *//**
    * 当channel可写时调用这个方法。
    * @throws IOException
    */
    void write() throws IOException;

    /** *//**
    * 当channel发生错误时调用。
    * @param e
    */
    void error(Exception e);
}
```

3.2 RTSP的测试类：RTSPClient.java

RTSP的测试类RTSPClient.java类的代码如下所示:

javaf代码

```java
package com.amigo.rtsp;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.SocketChannel;
import java.util.Iterator;
import java.util.concurrent.atomic.AtomicBoolean;
```

```java

public class RTSPClient extends Thread implements IEvent {

    private static final String VERSION = " RTSP/1.0/r/n";
    private static final String RTSP_OK = "RTSP/1.0 200 OK";

    /** *//** 远程地址 */
    private final InetSocketAddress remoteAddress;

    /** *//** * 本地地址 */
    private final InetSocketAddress localAddress;

    /** *//** * 连接通道 */
    private SocketChannel socketChannel;

    /** *//** 发送缓冲区 */
    private final ByteBuffer sendBuf;

    /** *//** 接收缓冲区 */
    private final ByteBuffer receiveBuf;

    private static final int BUFFER_SIZE = 8192;

    /** *//** 端口选择器 */
    private Selector selector;

    private String address;

    private Status sysStatus;

    private String sessionid;

    /** *//** 线程是否结束的标志 */
    private AtomicBoolean shutdown;

    private int seq=1;

    private boolean isSended;

    private String trackInfo;


    private enum Status {
        init, options, describe, setup, play, pause, teardown
    }

    public RTSPClient(InetSocketAddress remoteAddress,
            InetSocketAddress localAddress, String address) {
        this.remoteAddress = remoteAddress;
        this.localAddress = localAddress;
        this.address = address;

        // 初始化缓冲区
        sendBuf = ByteBuffer.allocateDirect(BUFFER_SIZE);
        receiveBuf = ByteBuffer.allocateDirect(BUFFER_SIZE);
        if (selector == null) {
            // 创建新的Selector
            try {
                selector = Selector.open();
            } catch (final IOException e) {
                e.printStackTrace();
            }
        }

        startup();
        sysStatus = Status.init;
        shutdown=new AtomicBoolean(false);
        isSended=false;
    }

    public void startup() {
        try {
            // 打开通道
            socketChannel = SocketChannel.open();
            // 绑定到本地端口
            socketChannel.socket().setSoTimeout(30000);
            socketChannel.configureBlocking(false);
            socketChannel.socket().bind(localAddress);
            if (socketChannel.connect(remoteAddress)) {
                System.out.println("开始建立连接:" + remoteAddress);
            }
            socketChannel.register(selector, SelectionKey.OP_CONNECT
                    | SelectionKey.OP_READ | SelectionKey.OP_WRITE, this);
            System.out.println("端口打开成功");

        } catch (final IOException e1) {
            e1.printStackTrace();
        }
    }

    public void send(byte[] out) {
        if (out == null || out.length < 1) {
            return;
        }
        synchronized (sendBuf) {
            sendBuf.clear();
            sendBuf.put(out);
            sendBuf.flip();
        }

        // 发送出去
        try {
            write();
            isSended=true;
        } catch (final IOException e) {
            e.printStackTrace();
        }
    }

    public void write() throws IOException {
        if (isConnected()) {
            try {
                socketChannel.write(sendBuf);
            } catch (final IOException e) {
            }
        } else {
            System.out.println("通道为空或者没有连接上");
        }
    }

    public byte[] recieve() {
        if (isConnected()) {
            try {
                int len = 0;
                int readBytes = 0;

                synchronized (receiveBuf) {
                    receiveBuf.clear();
                    try {
                        while ((len = socketChannel.read(receiveBuf)) > 0) {
                            readBytes += len;
                        }
                    } finally {
                        receiveBuf.flip();
                    }
                    if (readBytes > 0) {
                        final byte[] tmp = new byte[readBytes];
                        receiveBuf.get(tmp);
                        return tmp;
                    } else {
                        System.out.println("接收到数据为空,重新启动连接");
                        return null;
                    }
```

```
            }
        } catch (final IOException e) {
            System.out.println("接收消息错误:");
        }
    } else {
        System.out.println("端口没有连接");
    }
    return null;
}

public boolean isConnected() {
    return socketChannel != null && socketChannel.isConnected();
}

private void select() {
    int n = 0;
    try {
        if (selector == null) {
            return;
        }
        n = selector.select(1000);

    } catch (final Exception e) {
        e.printStackTrace();
    }

    // 如果select返回大于0, 处理事件
    if (n > 0) {
        for (final Iterator<SelectionKey> i = selector.selectedKeys()
                .iterator(); i.hasNext();) {
            // 得到下一个Key
            final SelectionKey sk = i.next();
            i.remove();
            // 检查其是否还有效
            if (!sk.isValid()) {
                continue;
            }

            // 处理事件
            final IEvent handler = (IEvent) sk.attachment();
            try {
                if (sk.isConnectable()) {
                    handler.connect(sk);
                } else if (sk.isReadable()) {
                    handler.read(sk);
                } else {
                    // System.err.println("Ooops");
                }
            } catch (final Exception e) {
                handler.error(e);
                sk.cancel();
            }
        }
    }
}

public void shutdown() {
    if (isConnected()) {
        try {
            socketChannel.close();
            System.out.println("端口关闭成功");
        } catch (final IOException e) {
            System.out.println("端口关闭错误:");
        } finally {
            socketChannel = null;
        }
    } else {
        System.out.println("通道为空或者没有连接");
    }
}

@Override
public void run() {
    // 启动主循环流程
    while (!shutdown.get()) {
        try {
            if (isConnected()&&(!isSended)) {
                switch (sysStatus) {
                case init:
                    doOption();
                    break;
                case options:
                    doDescribe();
                    break;
                case describe:
                    doSetup();
                    break;
                case setup:
                    if(sessionid==null&&sessionid.length()>0){
                        System.out.println("setup还没有正常返回");
                    }else{
                        doPlay();
                    }
                    break;
                case play:
                    doPause();
                    break;

                case pause:
                    doTeardown();
                    break;
                default:
                    break;
                }
            }
            // do select
            select();
            try {
                Thread.sleep(1000);
            } catch (final Exception e) {
            }
        } catch (final Exception e) {
            e.printStackTrace();
        }
    }

    shutdown();
}

public void connect(SelectionKey key) throws IOException {
    if (isConnected()) {
        return;
    }
    // 完成SocketChannel的连接
    socketChannel.finishConnect();
    while (!socketChannel.isConnected()) {
        try {
            Thread.sleep(300);
        } catch (final InterruptedException e) {
            e.printStackTrace();
        }
        socketChannel.finishConnect();
    }

}

public void error(Exception e) {
    e.printStackTrace();
}

public void read(SelectionKey key) throws IOException {
    // 接收消息
    final byte[] msg = recieve();
```

```
        if (msg != null) {
            handle(msg);
        } else {
            key.cancel();
        }
    }

    private void handle(byte[] msg) {
        String tmp = new String(msg);
        System.out.println("返回内容:");
        System.out.println(tmp);
        if (tmp.startsWith(RTSP_OK)) {
            switch (sysStatus) {
            case init:
                sysStatus = Status.options;
                break;
            case options:
                sysStatus = Status.describe;
                trackInfo=tmp.substring(tmp.indexOf("trackID"));
                break;
            case describe:
                sessionid = tmp.substring(tmp.indexOf("Session: ") + 9, tmp
                        .indexOf("Date:"));
                if(sessionid!=null&&sessionid.length()>0){
                    sysStatus = Status.setup;
                }
                break;
            case setup:
                sysStatus = Status.play;
                break;
            case play:
                sysStatus = Status.pause;
                break;
            case pause:
                sysStatus = Status.teardown;
                shutdown.set(true);
                break;
            case teardown:
                sysStatus = Status.init;
                break;
            default:
                break;
            }
            isSended=false;
        } else {
            System.out.println("返回错误:" + tmp);
        }

    }

    private void doTeardown() {
        StringBuilder sb = new StringBuilder();
        sb.append("TEARDOWN ");
        sb.append(this.address);
        sb.append("/");
        sb.append(VERSION);
        sb.append("Cseq: ");
        sb.append(seq++);
        sb.append("/r/n");
        sb.append("User-Agent: RealMedia Player HelixDNAClient/10.0.0.11279 (win32)/r/n");
        sb.append("Session: ");
        sb.append(sessionid);
        sb.append("/r/n");
        send(sb.toString().getBytes());
        System.out.println(sb.toString());
    }

    private void doPlay() {
        StringBuilder sb = new StringBuilder();
        sb.append("PLAY ");
        sb.append(this.address);
        sb.append(VERSION);
        sb.append("Session: ");
        sb.append(sessionid);
        sb.append("Cseq: ");
        sb.append(seq++);
        sb.append("/r/n");
        sb.append("/r/n");
        System.out.println(sb.toString());
        send(sb.toString().getBytes());

    }

    private void doSetup() {
        StringBuilder sb = new StringBuilder();
        sb.append("SETUP ");
        sb.append(this.address);
        sb.append("/");
        sb.append(trackInfo);
        sb.append(VERSION);
        sb.append("Cseq: ");
        sb.append(seq++);
        sb.append("/r/n");
        sb.append("Transport: RTP/AVP;UNICAST;client_port=16264-16265;mode=play/r/n");
        sb.append("/r/n");
        System.out.println(sb.toString());
        send(sb.toString().getBytes());
    }

    private void doOption() {
        StringBuilder sb = new StringBuilder();
        sb.append("OPTIONS ");
        sb.append(this.address.substring(0, address.lastIndexOf("/")));
        sb.append(VERSION);
        sb.append("Cseq: ");
        sb.append(seq++);
        sb.append("/r/n");
        sb.append("/r/n");
        System.out.println(sb.toString());
        send(sb.toString().getBytes());
    }

    private void doDescribe() {
        StringBuilder sb = new StringBuilder();
        sb.append("DESCRIBE ");
        sb.append(this.address);
        sb.append(VERSION);
        sb.append("Cseq: ");
        sb.append(seq++);
        sb.append("/r/n");
        sb.append("/r/n");
        System.out.println(sb.toString());
        send(sb.toString().getBytes());
    }

    private void doPause() {
        StringBuilder sb = new StringBuilder();
        sb.append("PAUSE ");
        sb.append(this.address);
        sb.append("/");
        sb.append(VERSION);
        sb.append("Cseq: ");
        sb.append(seq++);
        sb.append("/r/n");
        sb.append("Session: ");
        sb.append(sessionid);
        sb.append("/r/n");
        send(sb.toString().getBytes());
        System.out.println(sb.toString());
    }

    public static void main(String[] args) {
        try {
```

```
            // RTSPClient(InetSocketAddress remoteAddress,
            // InetSocketAddress localAddress, String address)
            RTSPClient client = new RTSPClient(
                    new InetSocketAddress("218.207.101.236", 554),
                    new InetSocketAddress("192.168.2.28", 0),
                    "rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp");
            client.start();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

其中:rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp为我在网上找到的一个rtsp的sdp地址,读者可自行更换,RTSP的默认端口为554.

3.3 运行结果

运行RTSPClient.java,运行结果如下所示:

javaf代码

```
端口打开成功
OPTIONS rtsp://218.207.101.236:554/mobile/3/67A451E937422331 RTSP/1.0
Cseq: 1


返回内容:
RTSP/1.0 200 OK
Server: PVSS/1.4.8 (Build/20090111; Platform/Win32; Release/StarValley; )
Cseq: 1
Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE, OPTIONS, ANNOUNCE, RECORD


DESCRIBE rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp RTSP/1.0
Cseq: 2


返回内容:
RTSP/1.0 200 OK
Server: PVSS/1.4.8 (Build/20090111; Platform/Win32; Release/StarValley; )
Cseq: 2
Content-length: 421
Date: Mon, 03 Aug 2009 08:50:36 GMT
Expires: Mon, 03 Aug 2009 08:50:36 GMT
Content-Type: application/sdp
x-Accept-Retransmit: our-retransmit
x-Accept-Dynamic-Rate: 1
Content-Base: rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp/

v=0
o=MediaBox 127992 137813 IN IP4 0.0.0.0
s=RTSP Session
i=Starv Box Live Cast
c=IN IP4 218.207.101.236
t=0 0
a=range:npt=now-
a=control:*
m=video 0 RTP/AVP 96
b=AS:20
a=rtpmap:96 MP4V-ES/1000
a=fmtp:96 profile-level-id=8; config=000001b008000001b509000001000000012000844 0fa282c2090a31f; decode_buf=12586
a=range:npt=now-
a=framerate:5
a=framesize:96 176-144
a=cliprect:0,0,144,176
a=control:trackID=1

SETUP rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp/trackID=1
 RTSP/1.0
Cseq: 3
Transport: RTP/AVP;UNICAST;client_port=16264-16265;mode=play


返回内容:
RTSP/1.0 200 OK
Server: PVSS/1.4.8 (Build/20090111; Platform/Win32; Release/StarValley; )
Cseq: 3
Session: 15470472221769
Date: Mon, 03 Aug 2009 08:50:36 GMT
Expires: Mon, 03 Aug 2009 08:50:36 GMT
Transport: RTP/AVP;UNICAST;mode=play;client_port=16264-16265;server_port=20080-20081


PLAY rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp RTSP/1.0
Session: 15470472221769
Cseq: 4


返回内容:
RTSP/1.0 200 OK
Server: PVSS/1.4.8 (Build/20090111; Platform/Win32; Release/StarValley; )
Cseq: 4
Session: 15470472221769
RTP-Info: url=rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp/trackID=1;seq=0;rtptime=0


PAUSE rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp/ RTSP/1.0
Cseq: 5
Session: 15470472221769


返回内容:
RTSP/1.0 200 OK
Server: PVSS/1.4.8 (Build/20090111; Platform/Win32; Release/StarValley; )
Cseq: 5
Session: 15470472221769


TEARDOWN rtsp://218.207.101.236:554/mobile/3/67A451E937422331/8jH5QPU5GWS07Ugn.sdp/ RTSP/1.0
Cseq: 6
User-Agent: RealMedia Player HelixDNAClient/10.0.0.11279 (win32)
Session: 15470472221769


返回内容:
RTSP/1.0 200 OK
Server: PVSS/1.4.8 (Build/20090111; Platform/Win32; Release/StarValley; )
Cseq: 6
Session: 15470472221769
Connection: Close


端口关闭成功
```

对照运行结果,读者可以熟悉RTSP的常用命令.

分类: RTSP/RTP/RTMP/HLS, 音视频、流媒体

DoubleLi
关注 - 29
粉丝 - 2053
+加关注

« 上一篇： RTP 和 RTSP的区别
» 下一篇： rtsp和sdp协议简介

posted on 2017-03-15 11:16 DoubleLi 阅读(64123) 评论(8) 编辑 收藏 举报

编辑推荐：
· 我给鸿星尔克写了一个720°看鞋展厅
· 带团队后的日常（三）
· 你为什么不想向上汇报？
· 传统.NET 4.x应用容器化体验（4）
· CSS 世界中的方位与顺序

最新新闻：
· 马斯克教苹果做事
· 10月28日上市！微软又晒《帝国时代4》：玩法、画质大提升
· 小米降噪耳机Pro全球首发骁龙畅听技术有何用？官方科普
· Intel誓言重回世界第一！第二天 台积电2nm正式获批
· 《仙剑奇侠传七》上架腾讯WeGame：最高直降30元
» 更多新闻...