

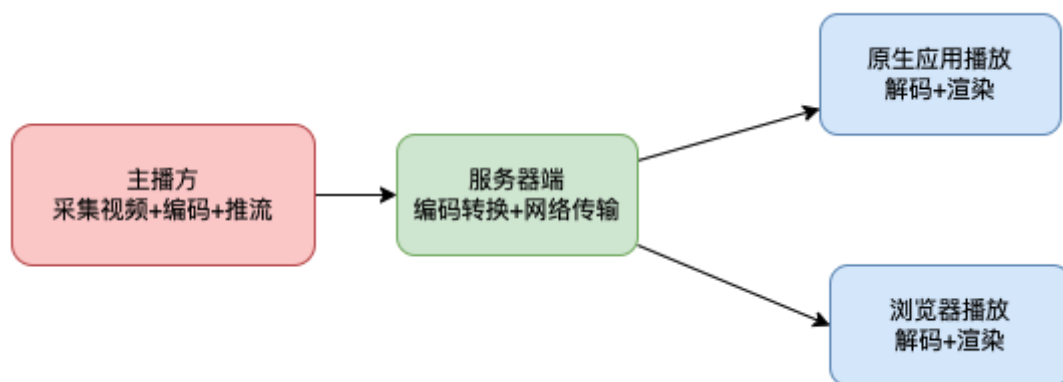
- 一、音视频直播技术发展
- 二、直播中的协议与格式
- 三、基于Nginx搭建点播直播服务器
 - 1、下载nginx和rtmp模块
 - 2、使用源码进行编译
 - 3、测试点播服务
 - 4、测试直播服务
 - 5、案例补充：使用ffmpeg推流

一、音视频直播基础

阶段目标：了解并演练音视频直播的基础技术，为在系统中集成音视频直播做准备。

一、音视频直播技术发展

一个典型的直播模型一般包括三个模块：主播方、服务器端和客户端。



音视频处理的每个流程都需要对底层数据做非常复杂的操作。Java语言并不擅长做这些与操作系统底层直接打交道的应用。音视频处理主要还是C/C++的天下。特别是音视频的开发，虽然比不上操作系统、游戏引擎的开发难度，但是就算在C/C++领域，也算得上是一个大的难点。

但是，幸好，经过很多年的发展，互联网已经开源出非常多灵活易用的产品，可以帮我们降低音视频的技术门槛。

二、直播中的协议与格式

常见的流媒体直播协议有RTSP、RTMP和HLS等多种协议。

RTSP(Real Time Streaming Protocol)是一个基于TCP的流媒体传输协议，用于控制流媒体数据的传输。它定义了如何初始化和控制一个或多个媒体流的传输，如播放、暂停、记录等。使用RTP来传输流媒体，延迟较低。时延通常在几百毫秒之内，是目前直播协议中最低的一种。但是技术实现比较复杂，成本也更高。通常用于安防视频监控、IPTV等场景。

RTMP是由Adobe公司开发的一种视频传输协议。RTMP基于TCP和HTTP协议实现，使用Flv(Flash Video)格式传输。RTMP协议具有实时传输、跨平台、支持多种编码方式和数据格式、安全性高等特点，被广泛用于实时音视频传输。在很长一段时间内，RTMP在PC端拥有非常大的市场。虽然不是国际标准，但是应用广泛，已经成为事实上的一种工业标准。

HLS是由苹果公司推出的视频传输协议。HLS基于HTTP协议传输流媒体数据，因此他的兼容性是最广的。几乎所有现代浏览器都支持HLS协议。但是HLS协议是使用分片MP4文件，也就是说他需要下载整个分片文件才能开始播放，因此他的延迟相比RTSP和RTMP要更高一点。

这三种流媒体协议各有优缺点，适用于不同的应用场景。从市场环境方面来看，经过多年的发展和磨合，很多CDN大厂已经非常完美的支持RTMP和HLS协议了，CDN不会对稳定营利的系统轻易做出变化。所以目前直播还是以RTMP和HLS协议为主。但是实际上，很多大型的直播产品往往会同时支持多种协议。例如抖音，也有自己的RTSP协议支持。

接下来，我们会以RTMP和HLS协议来进行点播和直播的实战。

三、基于Nginx搭建点播直播服务器

要实现直播和点播当然离不开服务器支持，我们可以使用开源的NGINX服务器搭建直播和点播服务。

当然，NGINX本身是不支持视频的，需要为NGINX增加相应的RTMP模块进行支持。

1、下载nginx和rtmp模块

```
# nginx
wget http://nginx.org/download/nginx-1.18.0.tar.gz
tar -zxvf nginx-1.18.0.tar.gz
# nginx-http-flv-module
wget https://github.com/winshining/nginx-http-flv-module/archive/master.zip
unzip master.zip
```

2、使用源码进行编译

```
# 安装操作系统的依赖项
sudo yum install gcc make pcre pcre-devel openssl openssl-devel make curl wget unzip vim
# 在nginx中进行配置
./configure --add-module=../nginx-http-flv-module-master
# 编译构建产品
make & make install
```

由于C语言的编译非常依赖服务器的各种类库，所以很容易出现各种各样的问题。

如果一切正常，那么就会在/usr/local/nginx目录下搭建起nginx服务，使用nginx -v，能看到版本输出，就表示nginx服务安装成功了。

3、测试点播服务

点播服务需要将视频文件上传到服务器中。例如在服务器的/usr/local/nginx/vod目录中提前上传一个1.mp4视频文件。

然后进入/usr/local/nginx/conf 目录下修改nginx.conf，添加以下片段

```
rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application vod{
            play /usr/local/nginx/vod;
        }
    }
}
```

修改配置文件后，使用nginx -t指令检查配置文件的正确性。

如果检查没有问题，就可以使用 nginx指令启动nginx服务。正常的话，访问服务器的80端口，就能够看到nginx的页面。

客户端需要使用流媒体播放器查看视频。VLC就是一个免费开源的流媒体播放器。事实上很多商用的播放器都是根据VLC修改得到的。

在VLC中打开一个网络地址： rtmp://192.168.65.210:1935/vod/1.mp4 就可以直接播放服务器上的视频文件。

4、测试直播服务

直播服务需要分三个步骤： 1、搭建直播服务端， 2、主播端推流， 3、客户端拉流

1、搭建直播服务端

在nginx的配置文件nginx.conf中，修改以下片段

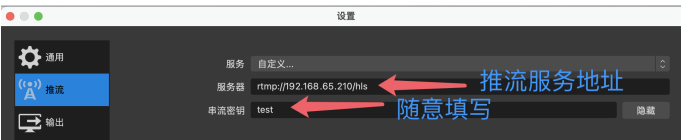
```
rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application hls {
            live on;
            hls on;
            hls_path /usr/local/nginx/html/hls;
        }

        application vod{
            play /usr/local/nginx/vod;
        }
    }
}
```

2、主播端推流

主播端可以使用一些通用工具进行视频推流。以主流的推流软件OBS为例，服务器地址填写为hls端口地址，串流密钥可以随意填写。



接下来，像你使用B站、抖音、Youtube等直播平台一样，编辑场景，推流直播就可以了。

从状态栏可以跟踪直播的情况。



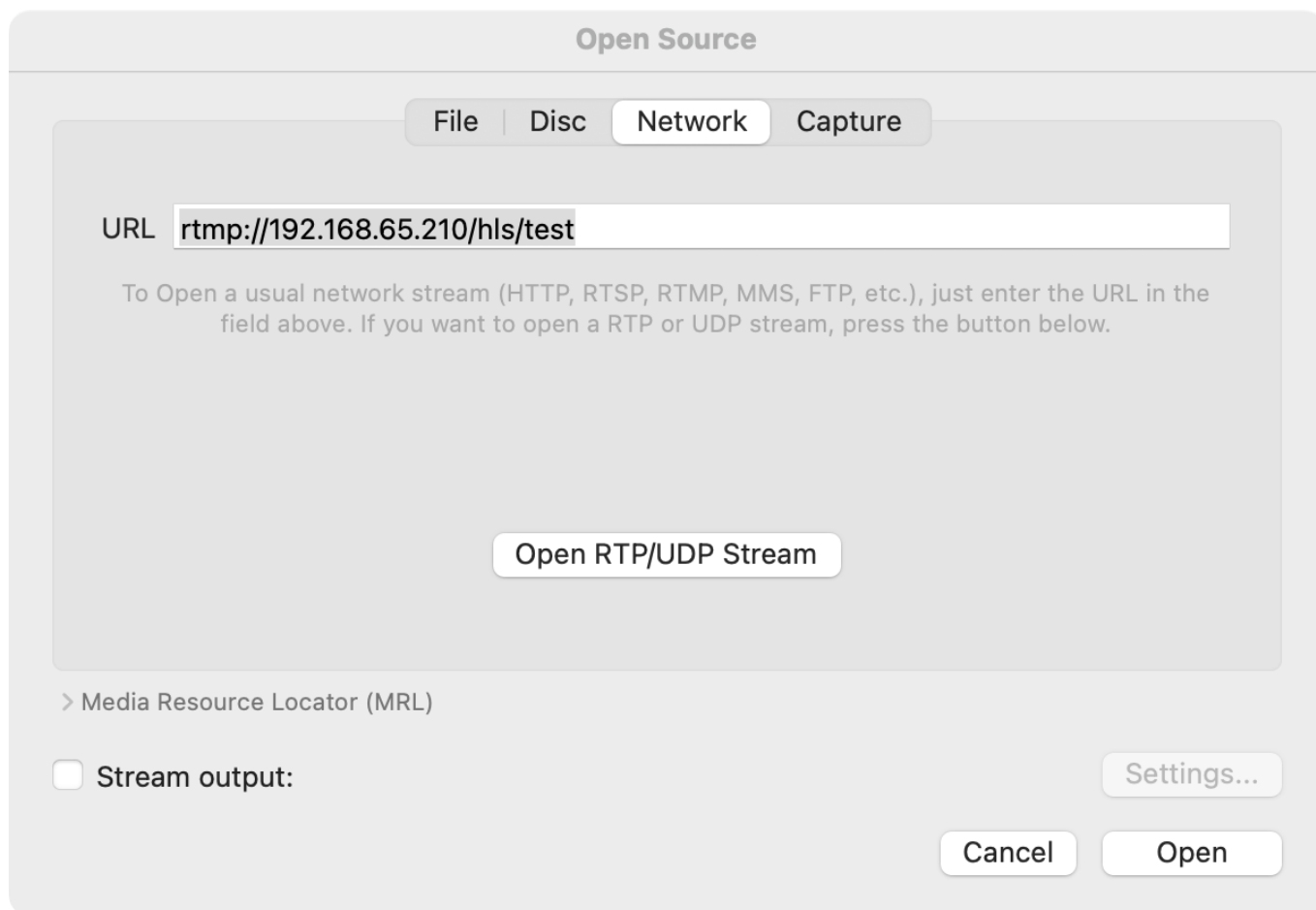
3、客户端拉流

音视频信号推流到服务器后，客户端也需要对信号进行解码才能观看。通常，这需要一些客户端来做信号转换。这里我们使用VLC来尝试一下。

VLC是一款自由、开源的跨平台流媒体播放器，可以播放大多数多媒体文件。同时，他也允许企业基于VLC进行定制开发。

官网地址：<https://www.videolan.org/> 很多平台都可以下载这个播放器。从官网上，你也大致能够了解到，开发一个流媒体播放器有多复杂。

在VLC中，只需要选择打开地址 rtmp://192.168.65.210/hls/test 就可以查看到我们自己的直播信号。



实际上，VLC中还可以通过HTTP协议进行拉流。拉流地址：<http://192.168.65.210/hls/test.m3u8> (这里用的是http的80端口)

作为对比，可以配置另外一个不带hls协议的直播入口

```
rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application live1 {
            live on;
        }
    }
}
```

这个入口不带hls协议，就无法保存视频切片，同样，也无法通过http访问

4、浏览器集成客户端

使用客户端播放还是太过复杂，不太适合浏览器访问的场景。有没有办法在浏览器上实现流媒体视频播放呢？当然有，而且有多种方案。当然，根据推流端的数据源不同，客户端的拉流方案也会不同。这里只针对之前演示的HLS方案。

实际上，我们在Nginx中配置的hls，全称是HTTP Live Streaming。HLS是苹果公司实现的基于HTTP的流媒体传输协议。他可以支持流媒体的直播和点播。既然是基于HTTP协议的，当然就天然支持浏览器。

HLS的基本实现原理是，将多媒体文件或直播流进行切片，形成一系列以ts为后缀的切片文件，以及以m3u8为后缀的索引文件。

在之前配置nginx的直播服务时，我们指定了一个hls_path属性，指向本地的/usr/local/nginx/html/hls文件夹。实际上，当时推流开始后，就可以在这个文件夹下看到不断更定的ts文件和m3u8索引文件。基于这个m3u8索引文件，就可以读取对应的ts文件，转换成视频数据播放。

play.html案例中，引入了第三方的video.js视线拉流播放。

```
<!DOCTYPE html>
<html lang="zh-CN">

<head>
<meta charset="UTF-8">
<meta http-equiv="Access-Control-Allow-Origin" content="*" />
<title>demo</title>
<link href="https://vjs.zencdn.net/7.0.3/video-js.css" rel="stylesheet">
</head>
<body>
<video id="myVideo" class="video-js vjs-default-skin vjs-big-play-centered" controls preload="auto" width="720" height="540" data-setup='{}'>
<source id="source" type="application/x-mpegURL" src="http://192.168.65.210/hls/test.m3u8">
</video>
</body>
<script src="https://vjs.zencdn.net/7.0.3/video.js"></script>
</html>
```

video.js是一个强大的网页嵌入式HTML5视频播放器组件库，可以轻松的解决复杂网页视频渲染。官网地址； <https://videojs.com/>

经测试，这个案例在chrome浏览器中，会报跨域错误。所以，视频播放功能不建议在HTML中单独使用，建议通过一些容器如VUE集成播放。

下一步，我们会选择在前端项目中集成video.js，实现基于网页的客户端拉流。而主播端，可以基于OBS进行推流。

5、案例补充：使用ffmpeg推流

1、什么是ffmpeg?

ffmpeg官网<https://ffmpeg.org/>，是一个可以录制，转换流媒体音视频信号的跨平台的解决方案。这是一个非常强大的音视频处理工具，不光可以做格式转换，也可以用来推流。实际上像早期的格式工厂，OBS等都可以基于ffmpeg实现。并且，ffmpeg是一个音视频处理的综合解决方案。不只可以推流，也提供了ffplay播放器以及ffprobe查看多媒体文件信息的一系列工具。

2、使用ffmpeg推流本地视频文件

将ffmpeg下载到本地后，在本地执行一下指令：

```
./ffmpeg -re -i /Users/roykingw/Movies/1.mp4 -c copy -f flv
rtmp://192.168.65.210:1935/hls/test
```

就可以将本地的一个视频文件1.mp4以流的形式往服务器上进行推送。推送上去的视频依然可以用VLC和浏览器拉流播放

3、使用ffmpeg推流摄像头

step1: 查找当前服务器上的设备

#mac下用这个指令查

```
ffmpeg -f avfoundation -list_devices true -i ""
```

```
[AVFoundation indev @ 0x7fe3adf05400] AVFoundation video devices:
[AVFoundation indev @ 0x7fe3adf05400] [0] FaceTime高清摄像头
[AVFoundation indev @ 0x7fe3adf05400] [1] Capture screen 0
[AVFoundation indev @ 0x7fe3adf05400] [2] Capture screen 1
[AVFoundation indev @ 0x7fe3adf05400] AVFoundation audio devices:
[AVFoundation indev @ 0x7fe3adf05400] [0] 7.1ch Surround Audio Device
[AVFoundation indev @ 0x7fe3adf05400] [1] MacBook Pro麦克风
[AVFoundation indev @ 0x7fe3adf05400] [2] TRTC Audio Device
```

windows下都是用下面这个格式查。

```
ffmpeg -list_devices true -f dshow -i dummy
```

#mac下的推流格式

```
ffmpeg -f avfoundation -framerate 30 -i "0" -f flv rtmp://192.168.156.128:1935/live/test
```

-i 0 就表示捕捉摄像头。同理, -i 1 就表示抓取主屏幕, -i 2就表示抓副屏幕

windows下应该要把格式换成dshow