WebRtC应用分析

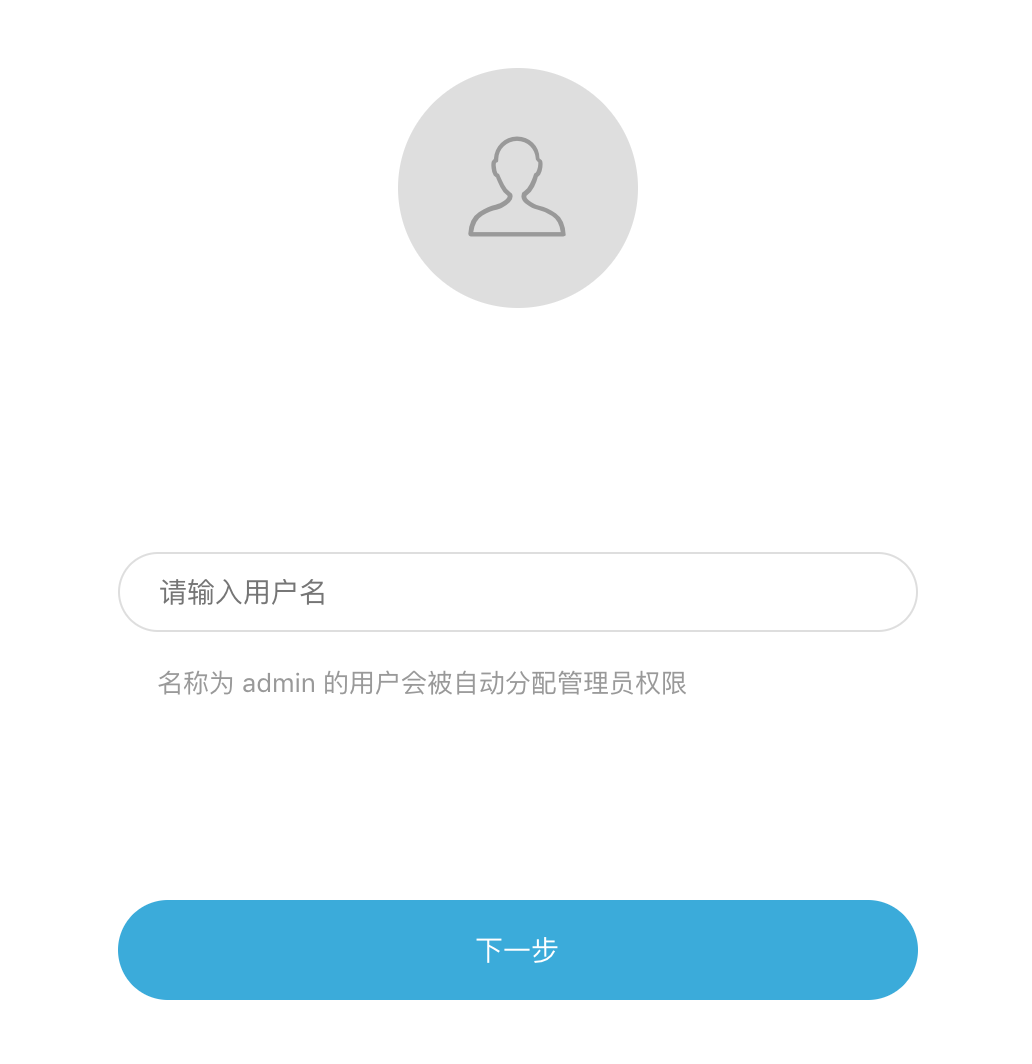
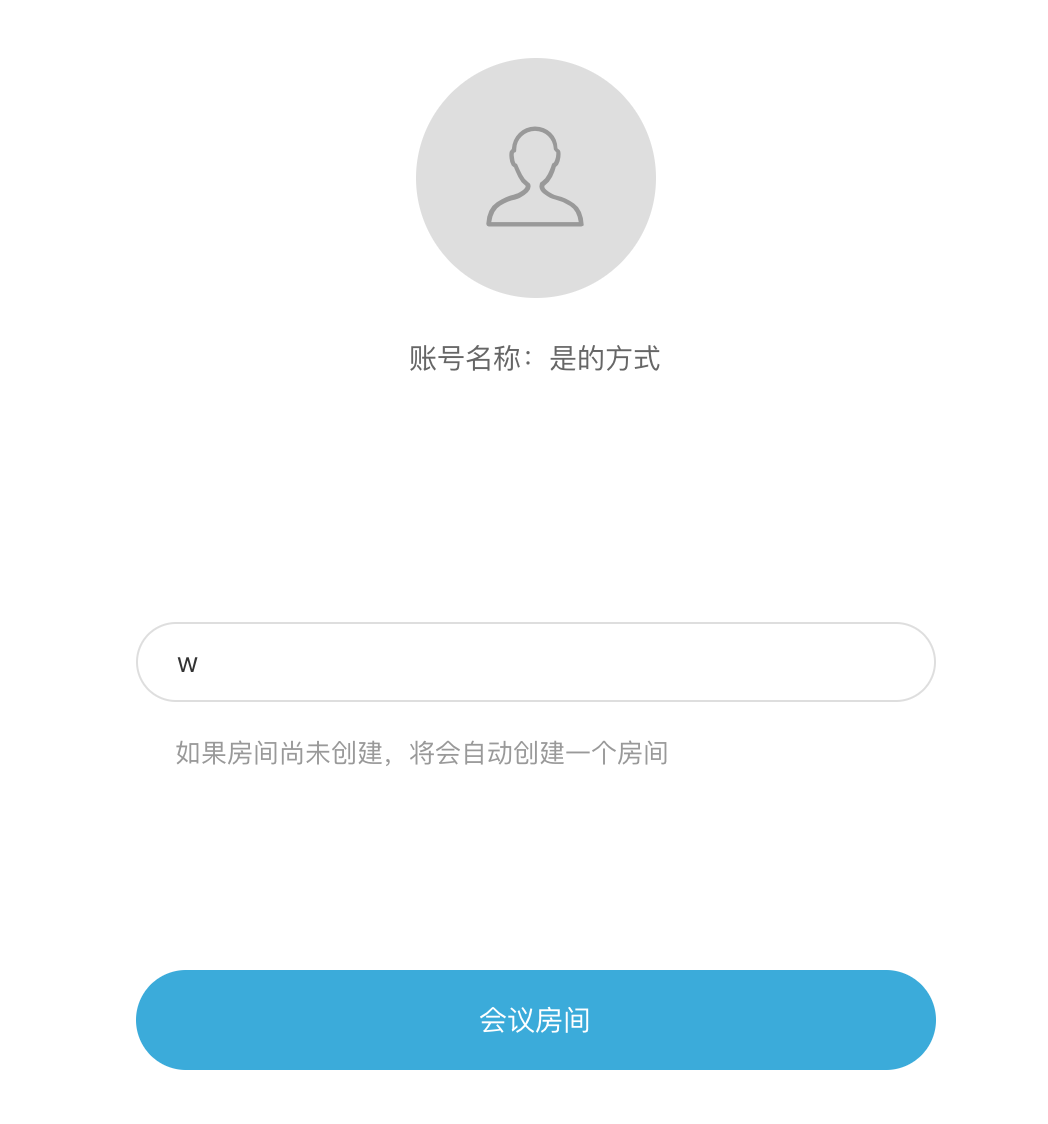
* **WebRTC实时通信**

1. **WebRTC实时通信技术介绍**

WebRTC名称源自网页即时通信（英语：Web Real-Time Communication）的缩写，是一个支持网页浏览器进行实时语音对话或视频对话的API。这里有两个关键字：1、网页浏览器；2、实时音视频通信。也就是说它不依赖服务器进行中转，实现浏览器之间点对点的通信，目的是无插件实现web端的实时通信的能力。

WebRTC提供了视频会议的核心技术，包括音视频的采集、编解码、网络传输、显示等功能，并且还支持跨平台：windows，linux，mac，android。

1. **WebRTC应用案例**

1. 实现两个浏览器音视频通信
2. 实现实时聊天通信
3. **WebRTC技术实现**
4. MediaStream：获取音频和视频流
5. RTCPeerConnection：音频和视频数据通信

在JS中，我们通过getUserMedia函数来处理音频和视频，该函数接收三个参数，分别是音视频的约束，成功的回调以及失败的回调。

在底层，浏览器通过音频和视频引擎对捕获的原始音频和视频流加以处理，除了对画质和音质增强之外，还得保证音频和视频的同步。



通过简单定义，getUserMedia获取到到音视频流数据，就可以显示到本地video对象了。

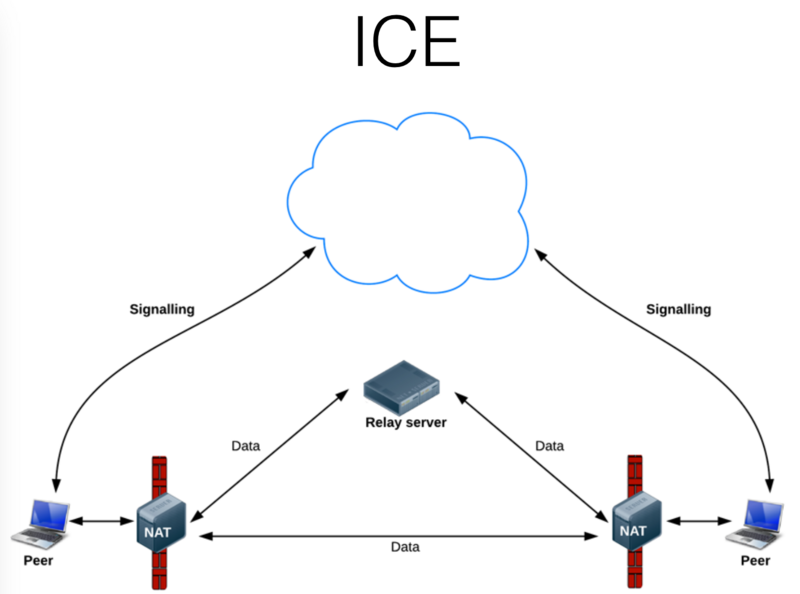
在获取到音频和视频流后，下一步要做的就是将其发送出去。但这个跟client-server模式不同，这是client-client之间的传输，因此，在协议层面就必须解决NAT穿透问题。

那为什么要NAT穿透呢？NAT(netword address transition网络地址转换)，这是因为ip4规则下网络ip地址不够用。为此，在外网与内网之间进行ip地址转换来实现扩容到目的。

一般情况下，我们客户端都是在NAT之后的。比如：

1. 客户端A通过链接路由器上网，它局域网地址：192.168.1.12，而公网地址10.13.23.12；
2. 客户端B通过链接路由器上网，它局域网地址：192.168.1.102，而公网地址100.133.213.127；

如果客户端A / B要进行通信就必须取得本地端的公共IP和端口，由于端与端之间存在多层防火墙和NAT设备阻隔，因此我们需要一种机制来收集两端之间公共线路的IP，而ICE则是干这件事的好帮手。



那ICE干嘛的？

**ICE**就是一个应用框架，而且浏览器封装好的了。在这个框架中， 可以整合其他现存的NAT穿透协议，如STUN、TURN、RSIP等。ICE是一种探索和更新式的解决方案，通过搜集自身和对端尽可能多的网络信息，尝试在这些地址间建立数据通道，并在这一过程中不断更新先前收集到的信息，从而找出和选择能够进行NAT穿透的数据通道。

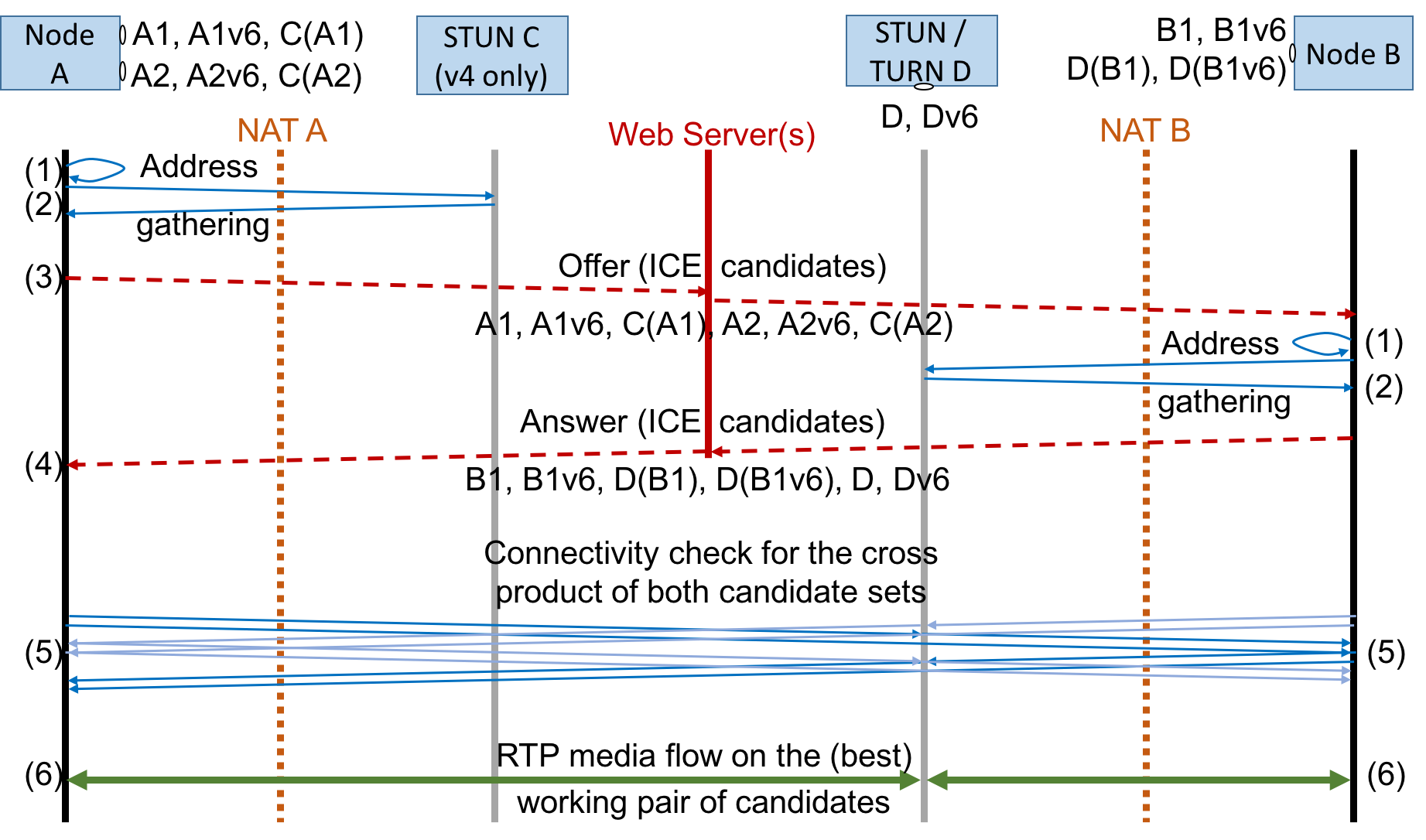
1. ICE代理向操作系统查询本地IP地址
2. 如果配置了STUN服务器，ICE代理会查询外部STUN服务器，以取得本地端的公共IP和端口
3. 如果配置了TURN服务器，ICE则会将TURN服务器作为一个候选项，当端到端的连接失败，数据将通过指定的中间设备转发。



到此，只要把配置信息初始化RTCPeerConnection传入就可以生产一个链接对象。并IEC框架对公网ip地址对收集。如果是本地测试，默认是对等对ip 地址和端口，这可以设置iceServer为null和{}

那两个端点之间是如何实现“全双工”长链接通信？如果实现点对点交互的呢？大概有6个关键的方法：

1. createOffer：发起一个offer
2. createAnswer：接收offer，后发起answer
3. setLocalDescription：把本地会话描述保存
4. setRemoteDescription：把远程会话描述保存
5. onicecandidata：获取到一个远程链接回调
6. onaddstream：获取了数据流回调



(1)(2)端点A/B初始化，并设置setLocalDescription。

(3)发起offer，指令服务器把offer转发给端点B

(4) (5)收到offer，并设置setRemoteDescription，触发onieccandidate事件，并把端点B到网络配置发送给端点A；端点B获取到端点A的answer，设置setRemoteDescription，触发onieccandidate事件，并把端点A到网络配置发送给端点B

(6)链接成功。

1. **实时聊天通信、指令系统**

实时聊天通信是通过websocket在客户端与服务端通过tcp协议建立一个‘全双工’长链接形式实现数据提交与推送。选择websocket这种广播的方式实现数据实时交互的目的：1、数据量不大，服务器有足够的并发处理能力；2、方便数据的保存；

**实现方式：**

1. 服务器创建ws服务对象
2. 客户端创建websocket对象
3. 通过ws://127.0.0.1:3000建立链接
4. 通过send发送数据、onmessage接收数据

另外，websocket除了实现聊天通信功能外，它还是协助webRTC建立链接的一个必不可少的系统，称为指令系统。

我们之前说过，两个端点要建立链接之前，必须要进行一点的交互。正如两个不认识的人要成为朋友，胸口一个“勇”字是不能被接受的，因此要出现一个中间人来事先传达双发意愿，而指令系统就可以看成这个中间人。比如：createOffer后，创建的offer要发送到另外到端点，可以通过ws.send到方法来发送指令；createAnswer后，创建的answer也可以通过ws.send要发送回offer端点。

另外，指令系统除了实现两点之间链接外，还在后续必要到交互提供转播到载体，比如音视频切换、开关、链接到断开等。

* **微信实时音视频**

微信实时音视频是腾讯云的音视频技术，结合腾讯浏览服务 TBS WebRTC 能力与腾讯实时音视频 TRTC SDK ，为客户提供跨终端、多平台互通、低成本、高品质、可定制的实时音视频通信服务的一款产品。

## 产品架构

## 三大终端（小程序、app、浏览器）

## 小程序通过RTMP协议，利用封装的<live-push><live-play>标签进行推拉流。

## app利用私有协议，比如rtmp\flv等进行推拉流

## 浏览器则是利用webRTC技术，与服务器建立链接，进行推拉流。

## 实时音视频系统，通过光纤通信建立的cdn网络分发数据。

## 旁路直播，基于 UDP 传输协议的 TRTC 服务，可以通过协议转换将音视频流对接到标准的直播 CDN 系统上。

## 点播系统，就是一个视频存储设备

## IM这就是一个指令系统。

## 开通服务

## 开通腾讯云实时音视频 TRTC 服务主要有以下三个步骤：

## 第一步：注册腾讯云账号。

## 第二步：创建实时音视频应用。

## 第三步：激活实时音视频服务。

## 参考 <https://cloud.tencent.com/document/product/647/17195>

## 音视频场景模式

## 直播推流：摄像头推流、录屏推流。

## 直播播放：RTMP协议、http-flv、HLS(m3u8)

## 点播播放：MP4、HLS(m3u8)、DRM加密

## 美颜滤镜：基础美颜、基础滤镜

## 直播连麦：连麦互动、跨房PK

## 视频通话：双人通话、视频会议

## 短视频：剪裁拼接、抖音特效、视频上传

## AI特效：大眼瘦脸、动态贴纸、绿屏抠图等

## 基本功能(小程序)

暂只针对国内主体的社交、教育、金融和汽车等类目的小程序开放，需要先通过类目审核，再在小程序管理后台，「开发」-「接口设置」中自助开通该组件权限。

## <live-pusher>

## 视频推流标签，实时音视频录制。需要用户授权 scope.camera、scope.record

## <live-player>

## 视频拉流标签，实时音视频播放。

## <webrtc-room>

## <webrtc-room> 标签是基于 <live-pusher> 和 <live-player> 实现的用于 WebRTC 互通的自定义组件。

## 首先，它不仅仅简单实现视频数据的录制与播放。它还是通过IM指令系统，实现必要的创建视频房间、会议人员加入与删除等基本逻辑功能。

## 其次，webrtc-room适合网络条件比较好的场景下，主播与听众是多对多的关系，是一个种“全双工”的通讯模式。听众接收主播的视频数据，听众的视频数据也同时上传给主播，相互之间可以进行实时互动。

## <live-room>

## 它也是要给直播模式的标签，与<webrtc-room>最大差别是主播与听众是一对多的关系，听众可以接收主播数据，而主播接收不到听众数据，相互之间不能实时互动。因此，适合在网络条件欠佳的场景下使用。

* **流媒体服务器**

## 流媒体服务器的主要功能是以流式协议（RTP/RTSP、RTMP等）将视频文件传输到客户端，供用户在线观看；也可从视频采集、压缩软件接收实时视频流，再以流式协议直播给客户端。

## 常见协议

参考<https://yq.aliyun.com/articles/683032?spm=a2c4e.11155472.0.0.468a7d2fT3DJa1>

## RTP: Real-time Transport Protocol,实时传输协议，一般用于多媒体数据的传输。

## RTCP: RTP Control Protocol，实时传输控制协议，同RTP一起用于数据传输的监视，控制功能。

## RTSP: Real Time Streaming Protocol,实时流协议，用于多媒体数据流的控制，如播放，暂停等。

## RTP/RTCP相对于底层传输层，和RTSP，SIP等上层协议一起可以实现视频会议，视频直播等应用。

## RTMP协议是Real Time Message Protocol(实时信息传输协议)的缩写，它是由Adobe公司提出的一种应用层的协议，用来解决多媒体数据传输流的多路复用（Multiplexing）和分包（packetizing）的问题。

## HLS (HTTP Live Streaming)是Apple的动态码率自适应技术。主要用于PC和Apple终端的音视频服务。包括一个m3u(8)的索引文件，TS媒体分片文件和key加密串文件。

## 

## 基本功能

## 主播用设备进行音视频采集→前处理→压缩编码→推流到服务器→后处理→服务器流分发→解码→播放器流播放。

## 服务器在流媒体传输过程中所发挥出的作用是“推流和分发”，分发，是重中之重。

## 分发：CDN是内容分发网络，通常会有两种方式，一种是接入第三方CDN，一种是自建流媒体服务器，一套优秀的直播源码应该支持在多家CDN中进行选择，并同时支持接入自建流媒体服务器以实现不同用户的选择。

## 后处理：包括各种美颜算法。

压缩编码：H.264 、H.265

1. H.264新一代的视频压缩标准,能在较低的带宽下传输高质量的图像,有着良好的网络适应性
2. H.265标准围绕着现有的视频编码标准H.264，保留原来的某些技术，同时对一些相关的技术加以改进。
3. 在相同的图象质量下，相比于H.264，通过H.265编码的视频大小将减少大约39-44%
4. 在码率减少51-74%的情况下，H.265编码视频的质量还能与H.264编码视频近似甚至更好，其本质上说是比预期的信噪比（PSNR）要好

## 流媒体服务器的传输方式有哪些

流媒体服务器所提供的流式传输方式主要有两种，即顺序流式传输和实时流式传输两种方式。

1. 顺序流式传输：即顺序下载，在下载文件的同时，用户可以观看在线媒体。如果使用普通的HTTP服务器，将音视频数据通过从头到尾的方式进行发送，即为顺序流媒体传输。
2. 实时流式传输：总是实时传送，非常适合现场事件。比如视频为现场直播或者是使用专用的流媒体服务器，可以应用像RTSP等专用的实时协议。实时流式传输必须要匹配链接带宽，也意味着图像质量会因为网络速度的降低而变差。

## 基于Nginx搭建rtmp+his自建流媒体服务器

## 准备需要的模块 nginx-rtmp-module、openssl、git

## 安装git

## $ yum -y install git

## nginx-rtmp-module

## $ git clone https://github.com/arut/nginx-rtmp-module.git

## yum 安装 openssl

## $ yum -y install openssl openssl-devel

## 安装Nginx服务器

## $ wget http://nginx.org/download/nginx-1.10.3.tar.gz

## $ tar -zxvf nginx-1.10.3.tar.gz

## $ cd nginx-1.10.3

## $ make && make install

## nginx和rtmp模块安装完成，此时在浏览器中打开ip地址或域名:端口号（默认80），将显示下图

## 

## 安装FFmpeg

## yum install -y ffmpeg

## 配置HLS

## mac:Users:lx:Desktop:屏幕快照 2019-07-02 17.57.24.png

## 在这之前，先简单认识一下FFmpeg，它是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序

## 通过这个一个方式，把视频上传到服务器。

## ffmpeg -re -i /Users/lx/Documents/webRTC/chrome.mp4 -vcodec copy -f flv rtmp://localhost:1935/hls/movie

## 

## 直播源码中最常见的有两种，RTMP和HLS。通过nginx搭建rtmp协议流媒体服务器总，展示视频如何通过rtmp协议上传一个视频到服务端，并以ts后缀名存储。

## 使用WebRTC和Kurento媒体服务器,来建立视频会议App

## Kurnto目前只支持linux服务器

## WebRTC是一项开源技术,它能够通过JavaScript API实现网页浏览器间的实时通信。

## 它被认为是一种P2P技术,能实现浏览器间的实时通信,但是有些功能,如组通信,媒体流录制,媒体广播或媒体转码是很难实现的

## WebRTC媒体服务器是一种多媒体中继(它位于两个通信端的中间)。媒体服务器能处理媒体流,并有各种功能,包括组通信,混合,转码,录制等

Kurento媒体服务器提供即时可用的组通信,混合,转码,录制和播放

## Kurento API, 客户端和协议

## Kurento媒体服务器的能力是通过Kurento API呈现给应用开发者的。这些API是通过叫做Kurento Client库实现的。

## Kurento提供了两种类型的客户端库: Java和JavaScript。如果你希望使用其它的语言,可以通过Kurento 协议来直接操作 Kurento。

## 这个协议允许控制 Kurento媒体服务器,它是基于标准的网络协议如[WebSocket]

## Kurento客户端的三种应用场景

## 在兼容WebRTC的浏览器上直接使用 Kurento JavaScript Client

## 在Java EE Application Server上使用Kurento Java Client

## 在Node.js 服务器上使用Kurento JavaScript Client

## 使用Kurento创建应用程序

## 例如,如果你想创建一个录制WebRTC流到文件的应用程序,你需要两个媒体元素:WebRtcEndpoint 和 RecorderEndpoint。

## 当一个客户端连接到这个应用程序时,你需要实例化这两个媒体元素,并使用 WebRtcEndpoint (它用来接收WebRTC流)来接收流后输送给RecorderEndpoint (它用来将媒体流录制到文件)。

## 最后,你需要连接它们,这样就能实现将前一个媒体元素接收到的流输送给后一年媒体元素

**参考** <https://blog.csdn.net/liuweihui521/article/details/79885324>