目录

[1.coturn 2](#_Toc528249443)

[1.1 coturn介绍 2](#_Toc528249444)

[1.2 coturn安装 2](#_Toc528249445)

[2.websocket信令服务器 3](#_Toc528249446)

[2.1websocket介绍 3](#_Toc528249447)

[2.2websocket作用 4](#_Toc528249448)

[3.webRTC 4](#_Toc528249449)

[3.1webrtc介绍 4](#_Toc528249450)

[3.2 webRTC核心介绍 5](#_Toc528249451)

[3.3多人视频几种方案 5](#_Toc528249452)

[3.2.1 Mesh全网状 6](#_Toc528249453)

[3.2.2 MCU混合 7](#_Toc528249454)

[3.2.3 SFU 8](#_Toc528249455)

[4.webRTC媒体服务器 9](#_Toc528249456)

[4.1媒体服务器介绍 9](#_Toc528249457)

[4.2多人视频为啥要用媒体服务器 9](#_Toc528249458)

[4.3常用的媒体服务器 11](#_Toc528249459)

[4.3.1 Jitsi平台 11](#_Toc528249460)

[4.3.2 Kurento媒体服务器 11](#_Toc528249461)

[4.3.3 Licode 12](#_Toc528249462)

[5.webRTC示例搭建 12](#_Toc528249463)

[5.1 webRTC视频需要的准备工作 12](#_Toc528249464)

[5.2视频连接具体实现 12](#_Toc528249465)

[5.2.1 A用户进入房间 13](#_Toc528249466)

[5.2.2 B进入房间 13](#_Toc528249467)

[5.2.3 C进入房间 15](#_Toc528249468)

[5.3webRTC拓展 16](#_Toc528249469)

**实现视频准备工作：**

* 穿透服务器coturn
* 前后台消息传输Websocket（注可以用ajax异步传输）
* 媒体服务器搭建（煤气化未用，视频人数有限）
* webRTC
* 学习网址：<http://webrtc.org.cn>
* 学习资料：https://blog.csdn.net/Guofengpu/article/details/51481604?utm\_source=blogxgwz0

# 1.coturn

## 1.1 coturn介绍

防火墙打洞服务器(STUN/TURN/ICE Server)，主要穿透防火墙获取外网地址来搭建相互访问的外网ip（具体详细介绍找度娘）

## 1.2 coturn安装

**下载编译安装coturn**

git clone https://github.com/coturn/coturn

cd coturn

./configure

make

sudo make install

具体步骤可以查看<https://blog.csdn.net/bvngh3247/article/details/80742396>

安装完成以后测试地址：<https://webrtc.github.io/samples/src/content/peerconnection/trickle-ice/>

**注：测试安装成功必须提示有relay返回外网ip才算安装成功如图1.1**

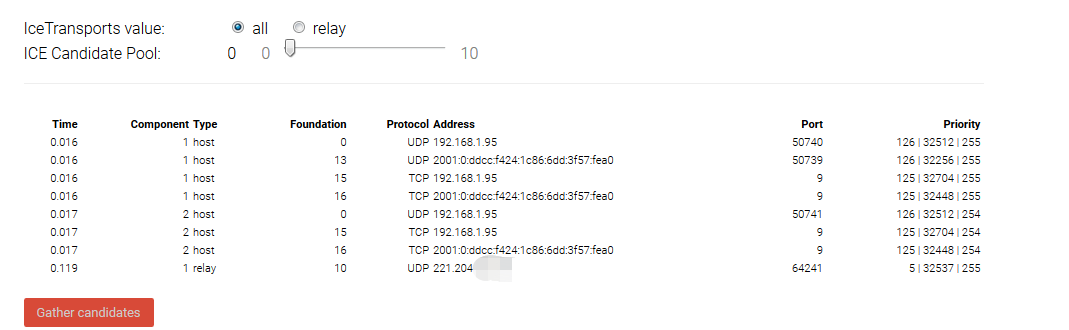


图1.1



# 2.websocket信令服务器

## 2.1websocket介绍

WebSocket协议是基于TCP的一种新的网络协议。它实现了浏览器与服务器全双工(full-duplex)通信——允许服务器主动发送信息给客户端。

WebSocket协议支持客户端与（选择加入该代码的通信的）远程主机之间进行全双工通信。用于此的安全模型是Web浏览器常用的基于原始的安全模式。 协议包括一个开放的握手以及随后的TCP层上的消息帧。 该技术的目标是为基于浏览器的、需要和服务器进行双向通信的应用程序提供一种通信机制。

具体功能实现请查看煤气化项目websocket连接或Websocket学习地址：<https://www.cnblogs.com/jingmoxukong/p/7755643.html>

## 2.2websocket作用

Websocket作为webRTC信令传输工具，主要作为SDP信息交换，达到RTCpeerConnection连接成功

# 3.webRTC

## 3.1webrtc介绍

WebRTC是一个可以使我们在浏览器或移动App中直接进行音频/视频交流的技术，例如Google Hangouts, Facebook Messenger, 和Discord.另外，它还可以进行P2P文件共享，处理大量音频数据，实现在线视频会议等等。

WebRTC详细介绍[请看这里](https://www.cnblogs.com/vipzhou/p/7994927.html)

## 3.2 webRTC[核心介绍](https://blog.csdn.net/fishmai/article/details/69681595)

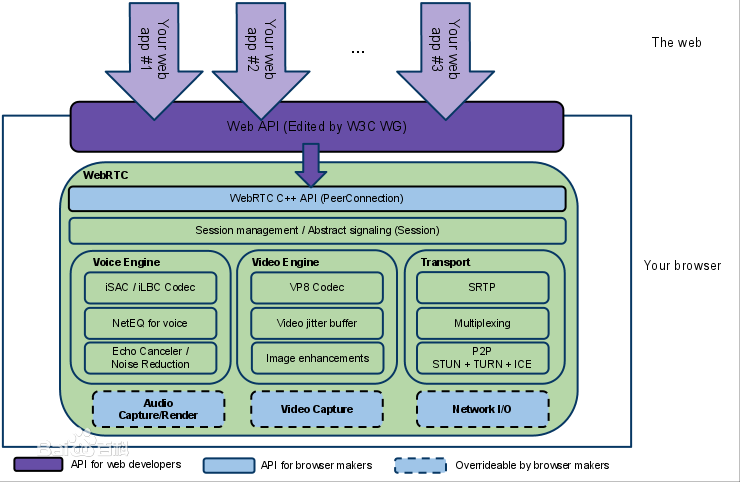


图3.2

必要的API

getUserMedia():开始捕获摄像头数据

RTCPeerConnection: （连接流控制，可以呼叫、中断控制）

RTCDataChannel: （数据通道，可用于文字信息的传输）

## 3.3多人视频几种方案

* MESH
* MCU
* SFU

### 3.2.1 Mesh全网状

Mesh是实现多方视频通话的最简单方式，因为它不需要任何基础结构的改变。所有的工作都是在浏览器完成的。唯一的问题就是它只对小数量的用户有效。如果我们想要支持更多用户，我们需要寻找另一种策略，一种允许我们混合所有这些由Mesh建立的连接的策略，为了避免客户端的大量CPU和带宽消耗。

全网状拓扑结构就是点与点之间连接成一个网状，各点之间两两连接，互相通信（每个用户都要和其他用户进行视频通讯）。

**Mesh特点：**

每一个P2P连接有独立的传输策略控制，通讯质量有一定的保障。但是，这种架构对于客户端系统是一种浪费，一方面需要分配更多的端口，消耗更多的系统资源；另一方面，由于要向其它三个客户端发送本地音视频数据，增加了上行网络带宽的消耗，在同等带宽条件下，支持的多人通话路数就相对有限，视频质量(码率)也比较低。

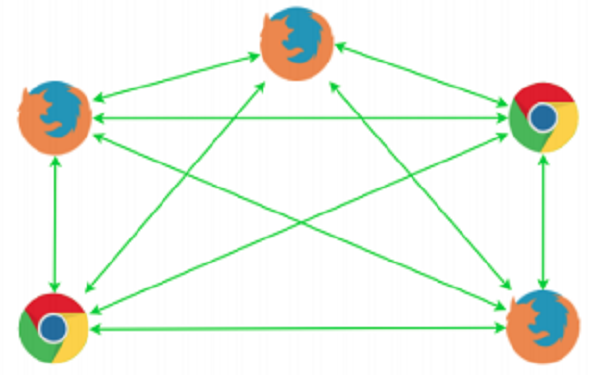


图3.2.1

### 3.2.2 MCU混合

MCU是传统视频会议系统中的核心控制单元，在WebRTC的系统实现中，适合于多人音视频通话场景，MCU可以对接收到的多路流进行转码和混合，并向每个终端输出单路流。

对于多方WebRTC一个不错的选择是MCU。MCU表示多点控制单元，又被称为混合，实现多方WebRTC交流的另一种策略。伴随着MCU，想法由使用peer建立连接变为只需要连接到中心服务器，中心服务器反过来发送信息到其它peers，并且对其它peers也是这样。

**MCU特点：**

MCU将接收到的多路流进行转码和混合，并向每个终端输出单路流的做法，节省了终端用户的下行带宽，并且还能够对不同网络条件的用户，订制不同码率的输出视频流，让多人场景有更好的用户体验。典型的应用场景是多人音视频通话。

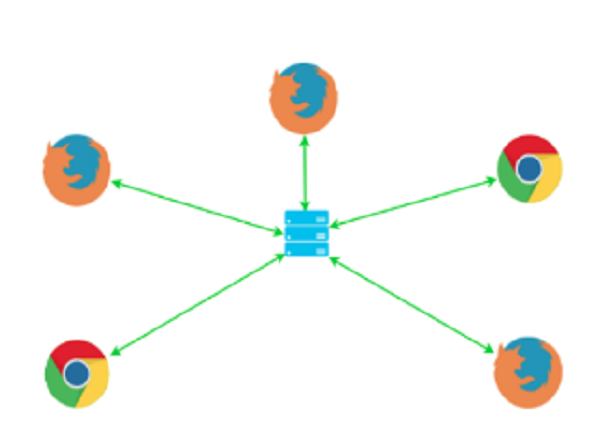


图3.2.2

### 3.2.3 SFU

SFU从发布客户端复制音视频流的信息，然后分发到多个订阅客户端。典型的应用场景是1对多的直播服务。

在SFU结构中，每个peer向媒体服务器发送自己的流，媒体服务器反过来将流引到其它peers。当然，这增加了客户端的花费，客户端现在必须对媒体流编解码。另外，当用户数量增加时，下载带宽也会增加。然而，媒体服务器具有中心控制点，它可以控制正在被引导路线的媒体，为了最小化这两个问题的影响。这就是SFU选择性部分起到作用的地方。尽管现在服务器不必进行过多的像编解码之类的处理了，其它任务可以被媒体完成。

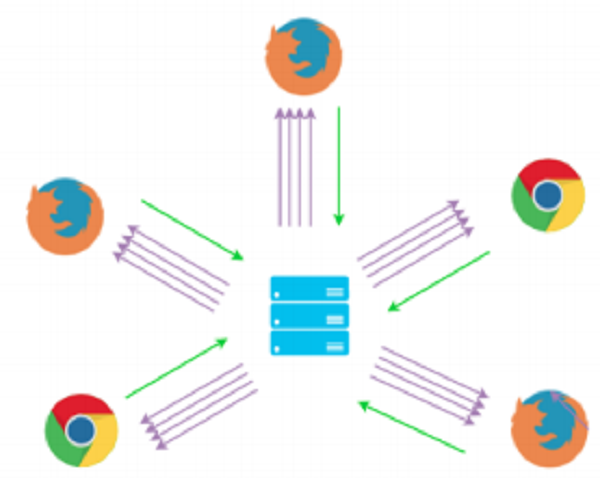


图3.2.3

# 4.webRTC媒体服务器

## 4.1媒体服务器介绍

## 4.2多人视频为啥要用媒体服务器

尽管可以使用对等通信（图3.2.1所的网状结构）与多个参与方保持视频通话，但随着参与者数量的增加，这件事情的可行性会越来越低。因为这要求每个对等端都将他/她的音视频流发送给每个对话参与者，同时还要接受从每个参与者处传来的音视频流。

现实情况中，即便是处在最优的网络环境下，参与人数超过了5人之后，网状结构的视频通话就很难良好地运行了。这就是媒体服务器派上用场的地方，因为它可以有助于减少客户端需要发送的数据流数量，甚至可以减少客户端需要接收的数据流数量，这取决于媒体服务器的功能。

当一个媒体服务器作为这种媒体中继时，通常称为SFU（Single Forwarding Unit，单向转发单元），主要目的是在客户端之间转发媒体流。还有一个MCU（Multipoint Conferencing Unit，多点会议单元）的概念，它用于寻址一个媒体服务器，它不仅可以进行转发，还可以在通过它的媒体流上操作（比如将所有视频或者音频流混合成一个）。

**视频录制**

让所有视频流都通过一个媒体服务器的主要好处之一是我们可以录制媒体并且存储下来以备它用，比如一些在网格结构中非常难完成的事情。

**与其他通信技术的整合**

使用媒体服务器的另一个优势是可以与超出网页技术允许范围内的系统相通信，比如说通过SIP中继与PSTN进行通信，或者穿过RTMP流传输到支持它的服务中，比如Facebook Live和YouTube流直播。

**媒体流的处理**

一些媒体服务器允许对视频和音频流做底层上的处理，比如能够在视频上运行计算机视觉模型或者将音频流发送到语音识别引引擎，例如google Speech。这些功能将webrtc提升到另外一个层次。依我看来，它提供了更加丰富和创新性的实时交互，为一个普通的通信平台增加了很多价值。

## 4.3常用的媒体服务器

### 4.3.1 [Jitsi平台](https://jitsi.org/jitsi-meet/)

Jitsi不仅是一个WebRTC媒体服务器，而且还是一个围绕其搭建的完整的平台。Jitsi的产品家族包括了Jitsi Videobridge（媒体传输，SFU），Jitsi Meet（会议网络客户端），Jicofo（Jitsi会议焦点），Jitsi SIP电话等等。Jitsi平台最吸引人的特点是它包含了一个交流平台的一切，可以在几个小时内就启动和实现运行。它还使用了Jingle（XMPP）和全功能的Web界面实现自己的信号传输。遗憾的是，最大的缺点就是实施媒体录制，因为Jitsi没有可靠的，易于使用的解决方案。

### 4.3.2 [Kurento媒体服务器](https://www.kurento.org/)

这是目前最通用的解决方案之一。它不仅仅是一个媒体服务器，还是一个用来搭建媒体服务器的工具箱。Kurento的主要优势在于它的多功能性，通过引入媒体工作流的概念允许在代码中定义媒体要如何进行流动，以及向哪里流动。这使得WebRTC开发者能够编写和集成注入机器视觉（例如二维码扫描，人脸识别），实时媒体修改，以及与RTP（VoIP）服务互操作的非常有趣的特性。Kurento在一个实例中也可以配置充当SFU或者MCU，或者两者。

### 4.3.3 [Licode](http://lynckia.com/licode/index.html)

Licode是基于webRTC技术之上的开源项目，通过更便捷（easy，fast and scalable）的接口你可以快速搭建出基于webRTC技术的网络视频会议系统，或者与此类似的系统。

# 5.webRTC示例搭建

## 5.1 webRTC视频需要的准备工作

以Mesh为例（如若想要其他架构方案需要搭建媒体服务器）：

1. Coturn服务器搭建（[详情查看1.2](#_1.2_coturn安装)）
2. 选择websocket作为信令服务器（具体搭建请查看煤气化项目）

以上两步不做详细介绍。

## 5.2视频连接具体实现

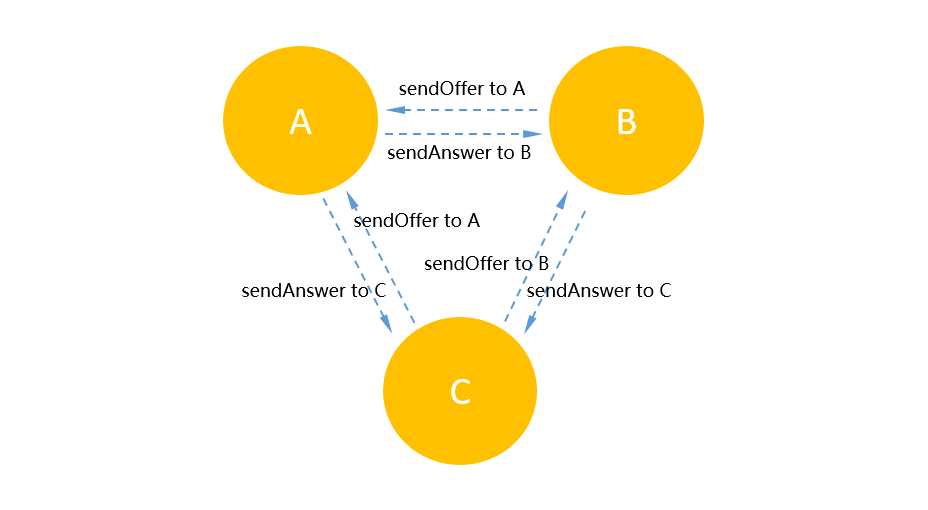


图5.2

**以图5.2为例**

### 5.2.1 A用户进入房间

当第一个用户 A 接入通话房间时，由于只有一个用户，所以除了让用户 A 调用本地摄像头获取视频流之外，不做任何操作，如图5.2.1获取本地摄像头：

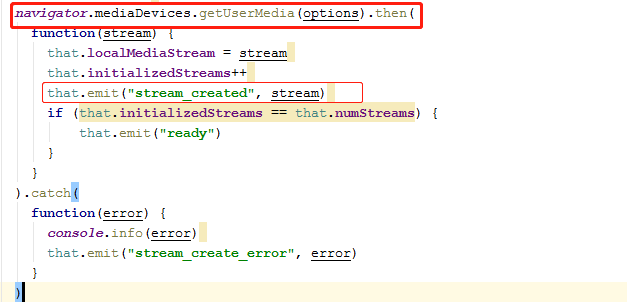


图5.2.1

### 5.2.2 B进入房间

当第二个用户 B 接进来时，除了让 B 调用本地摄像头获取视频流之外，还需要在B端创建该房间除 B 外（即 A）的所有 RTCPeerConnection 实例，可以将这些实例以用userId索引的方式保存在一个数组里，如：**this**.**peerConnections** = {}。与此同时 A 端也需要创建新加入用户 B 的 RTCPeerConnection 实例。

需要说明：当B进入房间时，websocket会推送B的信息给A如图5.2.2 (1)，A端会创建B的peer。

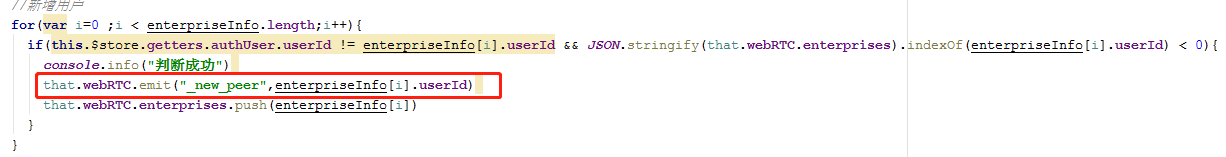


图5.2.2(1)

当B端创建peer以后，同时B会除B之外给A发送Offer请求如5.2.2(2)，并且把本地生成的SDP（[详解](https://datatracker.ietf.org/doc/draft-nandakumar-rtcweb-sdp/?include_text=1)）通过websocket.send发送给A端，A端会接收到offer请求会创建answer应答生成SDP发给B如图5.2.2(3)

B端会除B之外给A发送请求



图5.2.2(2)

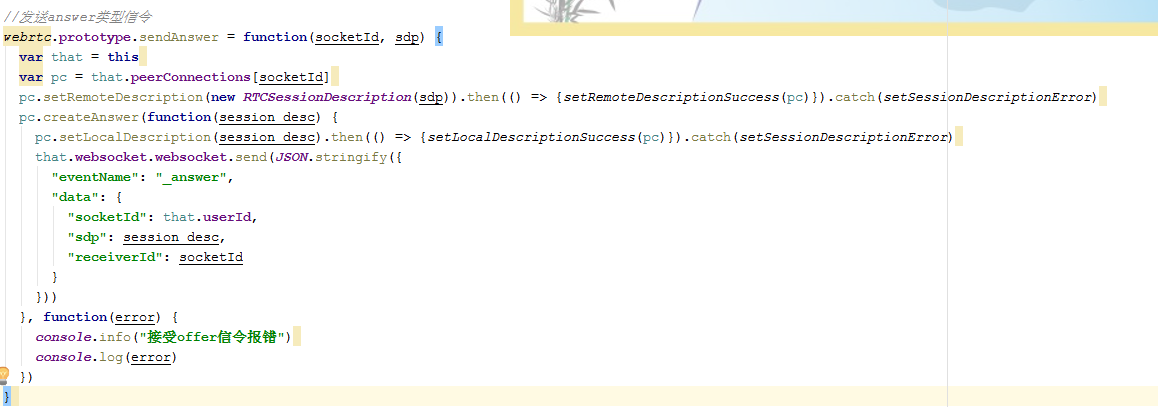


图5.2.2(3)

这时信号交换完成peer连接成功iceConnectionState的状态是connected和completed。获取视频流的方法如图5.2.2(4)，pc = RTCPeerConnection

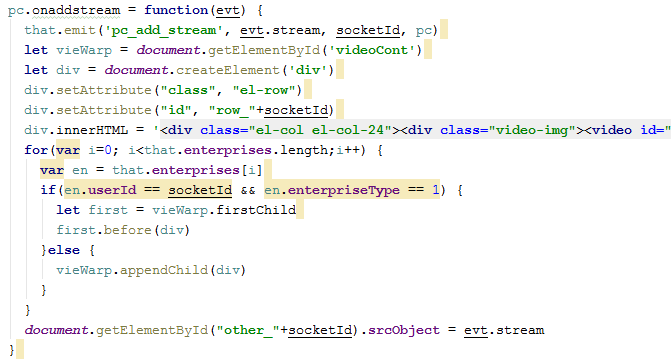


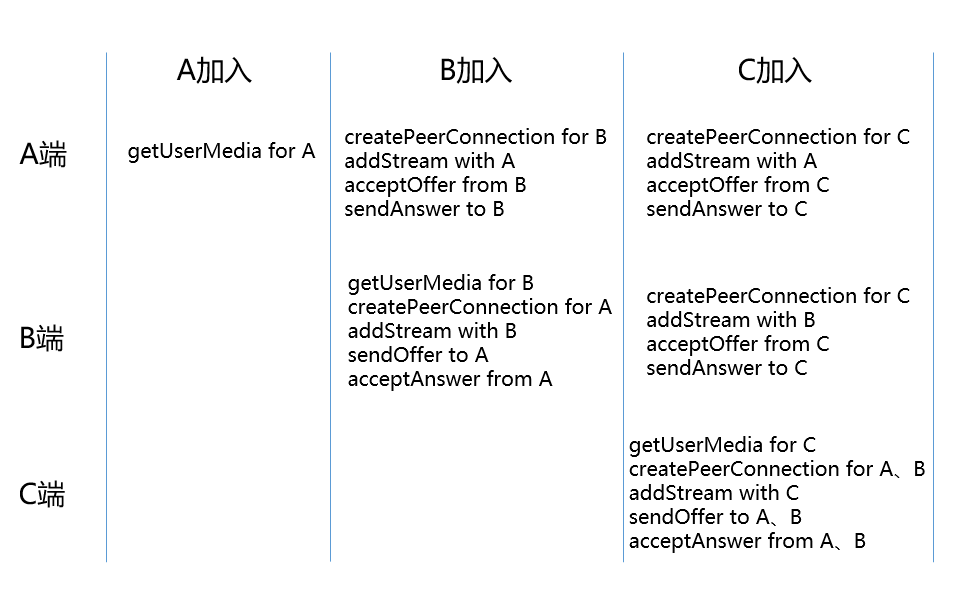
图5.2.2(4)

需要说明：如果内网环境下icecandidate候选连接地址不需要做穿透连接。Coturn只有在外网环境下才做穿透服务。

### 5.2.3 C进入房间

当第三个用户 C 接进来时，除了让 C 调用本地摄像头获取视频流之外，还需要在 C 端创建该房间除 C 外（即 A、B）的所有 RTCPeerConnection 实例，可以将这些实例以用户userId索引的方式保存在一个数组里，如：peerConnections[userId] = new RTCPeerConnection(configuration)。与此同时 A 端也需要创建新加入用户 C 的 RTCPeerConnection 实例，B 端也需要创建新加入用户 C 的 RTCPeerConnection 实例。并且C端分别会给A和B发送offer请求，当A和B接收到offer以后分别生成answer应答发送给C。至此A、B、C相互之间视频连接成功。

全网状拓扑结构的视频通话整个流程梳理如下：



## 5.3webRTC拓展

1. 点对点连接时不能两端同时发offer或者answer
2. [RTCDataChannel](https://developer.mozilla.org/en-US/docs/Web/API/RTCDataChannel)数据传输在此次开发中未实现
3. 用户量在10以上需要更换架构mcu或者sfu,此两种架构都需要[媒体服务器](#_4.webRTC媒体服务器)
4. 音频功能未放开，如果后续需要可以添加，mesh方案音频不存在噪音，所以此方案也不需要除燥，其他架构需要除燥，并且视频也需要转码处理。工作量较大且可能存在丢包的可能。