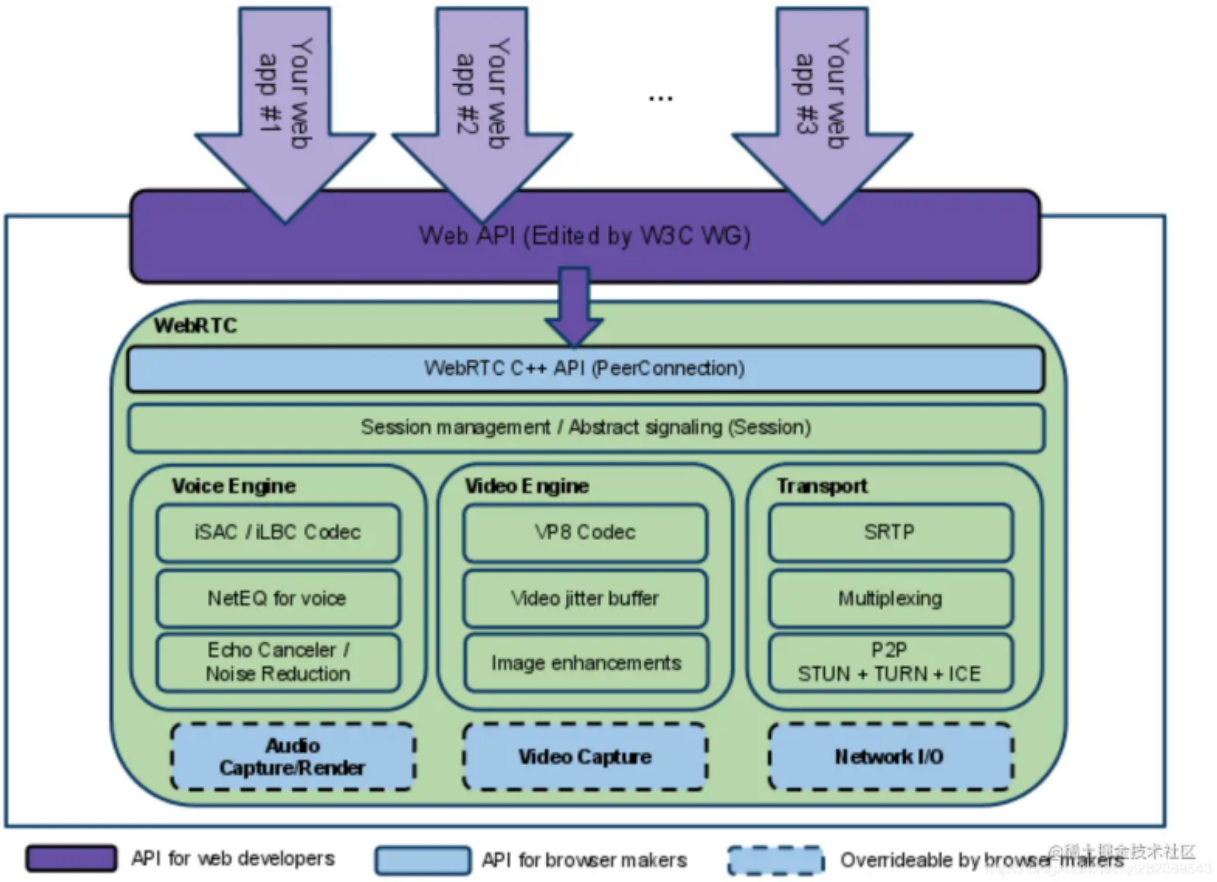
WebRTC介绍

webrtc是网页即时通信，是一个音视频处理+即时通信的开源库，是一个非常优秀的多媒体框架，而且还跨平台，优势是整个网络中实现音视频的传输，对网络抖动，丢包，在网络层面的算法优化保证了音视频传输的稳定，还对网络传输经常发生的回音等问题进行了优化处理，比如回音消除，降噪。

浏览器支持情况

除了IE之外，几乎所有的主流浏览器都支持，chrome，苹果safari，firefox，qq浏览器，360和edge。

webrtc架构



整个浅绿色的部分是webetc核心架构层，它封装各种给web端使用的API,紫色的部分属于应用层，使用核心层提供的API,可以在应用层拓展相关API,调用核心层的接口。

核心层又分为四层:

- WebRtC C++API(PeerConnection):这层的API相对比较少，最主要就是实现P2P连接。PeerConnection里面有很多的接口，如音视频采集，音视频传输等等。
- Session(会话层):用来管理音视频，非音视频数据传输，处理相关逻辑。
- 最核心的第三层，包含了音频引擎，视频引擎和传输三大模块。
- 最底层是硬件相关的适配层，包含了音配的采集，视频的捕捉等。同时这些模块可以自己去重载，增加了WebRTC的灵活性，为跨平台实现了基础。

WebRTC最核心的三大模块，VoiceEngine，VideoEngine，transport，其中vioce只处理音频，videoEngine只处理视频，音频和视频相互独立的，音视频同步不在这里面。

VoiceEngine包含了三大模块

- ISAC:主要是编码和解码比如G711,Opus
- NetEq:是一个音频的缓冲区buffer，用于做网络的适配，如我们做防止音频抖动，这里面设计了很多相关的算法。
- Echo Canceler:这里面解决了回音消除，降噪等问题的处理算法。

VideoEngine包含了三大模块:

- VP8 Codec:主要是视频编码和解码器。VP8和VP9都是google开发的，还支持H264,H265
- Video jitter buffer:视频缓冲区buffer，也是用来防止视频抖动的
- image enhancements:这块是做图像相关的，如图像增强，它里面做的比较少，把相应的接口留了出来，我们可以自己做一些比如美颜，贴图等加进去。

Transport传输模块包含了三大模块，传输底层使用的是UDP协议，因为它对及时性要求更高，允许部分丢帧，所以UDP很合适，同时它利用各种成熟的算法保证高质量的音视频传输，所有音视频的数据发送接受都是传输层来做，架构层次是非常清晰的。

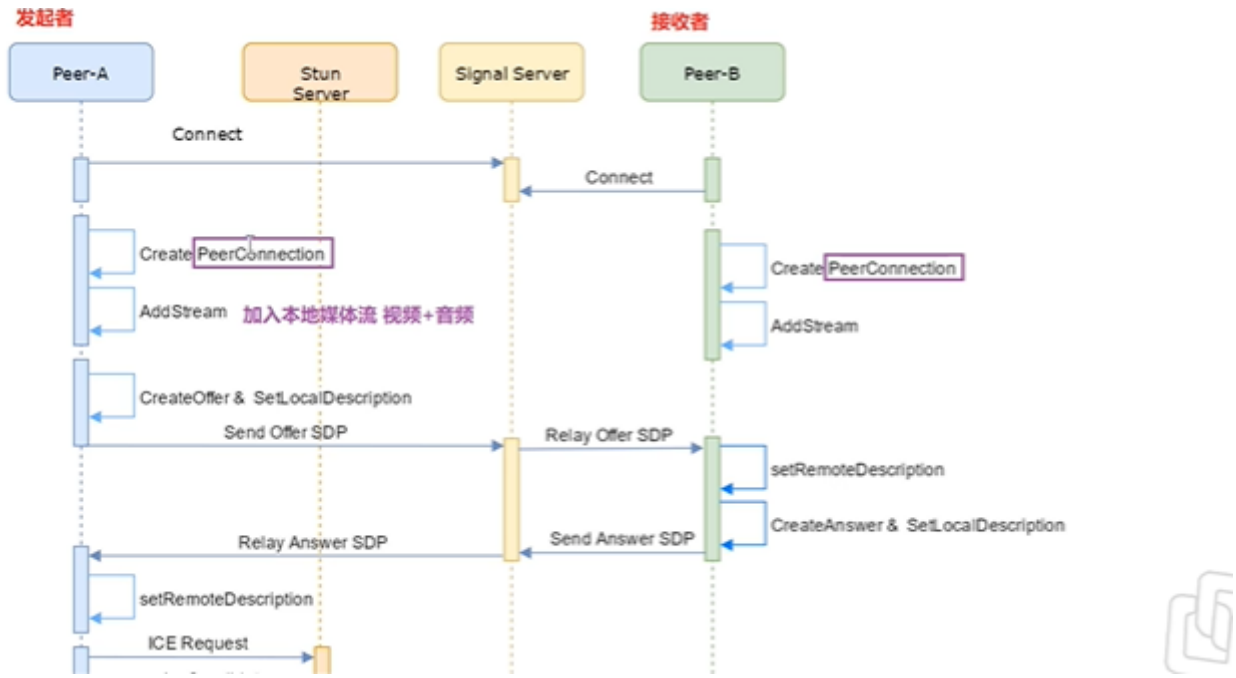
- SRTP:一般正常视频传输用的是RTP协议，浏览器对按前行要求较高，所以增加了加密处理用的是SRTP协议，
- Multplexing:这里是多个流用一个通道。
- P2P(STUN+TURN+ICE)是P2P相关的技术，是webrtc最核心的技术。

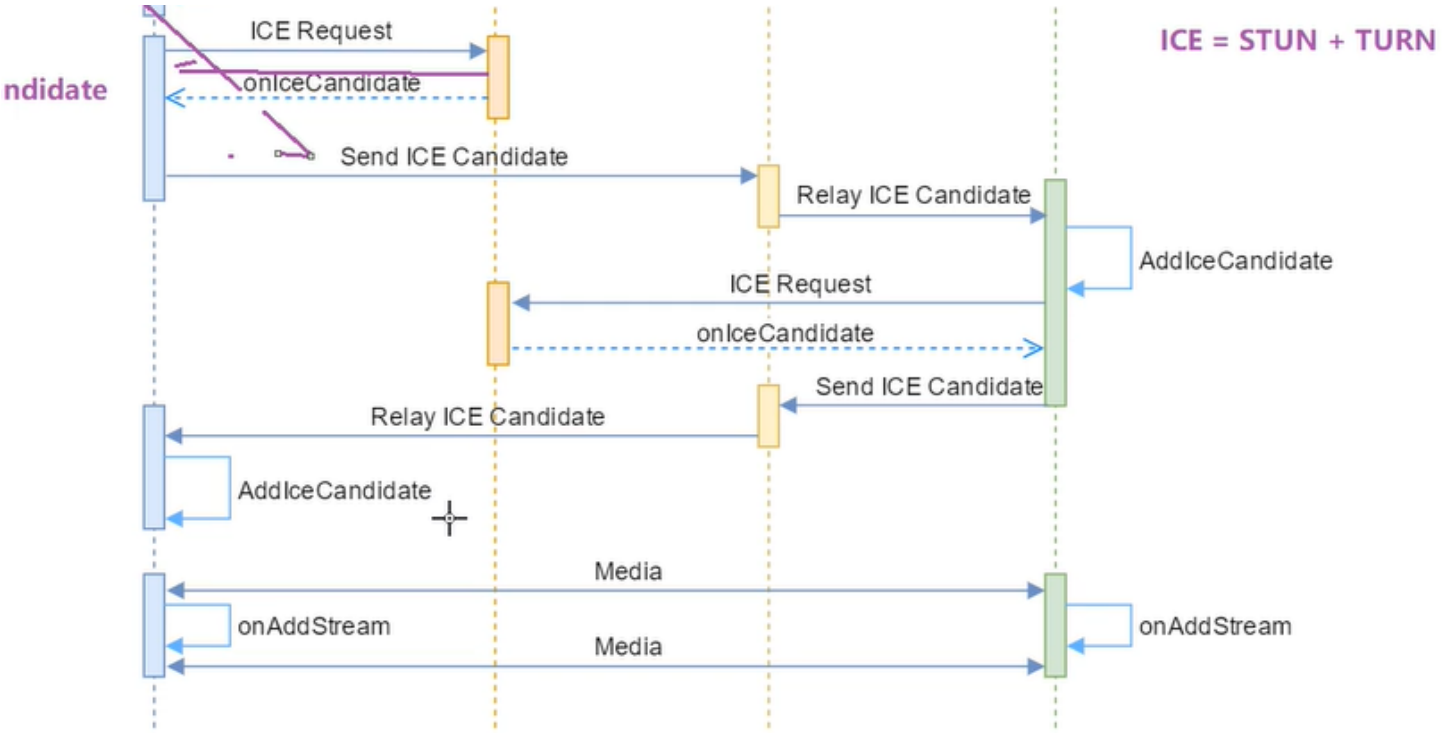
WebRTC APIs

1. MediaStream-MediaStream用来表示一个媒体数据流，通过getUserMedia接口获取，访问输入设备如麦克风和摄像头等。

2. RTCPeerConnection- RTCPeerConnection对象，允许用户在两个浏览器之间建立通信，可以将网络捕获的音频和视频流实时发送到另一个WebRtc端点。

一对一通信流程





- 媒体协商
  先创建peerconnection，然后把本地媒体流加入进去，之后发起者调用它的createOffer方法，获取到SDP(浏览器具有的编码解码能力),然后通过setLocalDescription放到PeerConnection中去，然后

把SDP发送到信令服务器，然后信令服务器会把SDP发到接收方，然后接收方会把发送方的SDP放到remoteDescription(设置自己的SDP是local，别人的是remote)，然后通过createAnswer获取到接收方的SDP，然后通过setLocalDescription保存到peerconnection，然后把它发送到信令服务器，信令服务器会把SDP发送到发送方去，发送方存到remote里。这样就完成了媒体协商

- 网络协商
  先向STUN服务器发送请求获取公网的IP地址，然后STUN服务器会回发给发送方的peerConnection，通过peerConnection的onIceCandiate方法获取到公网IP，然后将Candiate信息发给信令服务器，然后信令服务器转发，然后接收方会把地址保存，然后也去向STUN服务器请求，获取到公网地址，获取到之后也是用onIceCandiate方法获取到自己的公网地址，然后把它发送给信令服务器，信令服务器会把它发给发送方，发送方收到后添加本地，之后peerconnection会自动去做互联，如果连上了就说明打洞成功，就说明P2P连接成功，这样音视频数据不需要经过信令服务器了，直接点对点传输，通过addStream方法可以获取对方的媒体流。如果不能连接成功就需要通过TURN中继服务器来中转。

媒体协商:交换彼此支持的媒体解码编码格式

NAT:备通常都在一个局域网内，局域网的地址通过端口映射可以获取到外网地址+端口

STUN:STUN是一种网络协议，它允许NAT后的客户端找出自己的公网地址，查出绑定的端口，这些信息会被用来进行后续的UDP通信。STUN服务器就是给无法在公网环境，在局域网的设备分配公网IP然后发送给设备的服务器。
缺点：STUN并不是每次都能成功的分配地址，同时P2P在传输媒体流使用本地宽带，多人视频过程中，质量的好坏取决于使用者本地宽带。
TURN:在STUN分配公网地址失败后，可以通过TURN服务器作为中继地址，来转发，这种方式的带宽由服务器承担，所以多人视频童话本地带宽压力较小。

ICE:ice整合了STUN和TURN
媒体协商:SDP
网络协商:candidate

信令服务器:交换发送方和接收方的网络协商信息和媒体协商信息。