**基于WebRTC的直播新方案**

使用WebSocket作为高可用信令服务器，使用Kurento Media Server作为WebRTC媒体服务器，使用coturn作为NAT穿越服务器。实现设备与客户端之间点对点直播或设备与多客户端之前一对多直播间，降低直播延时，服务端实时维护设备与客户端直接连接，Kurento支持直播同时创建录像点；

视频流传输：

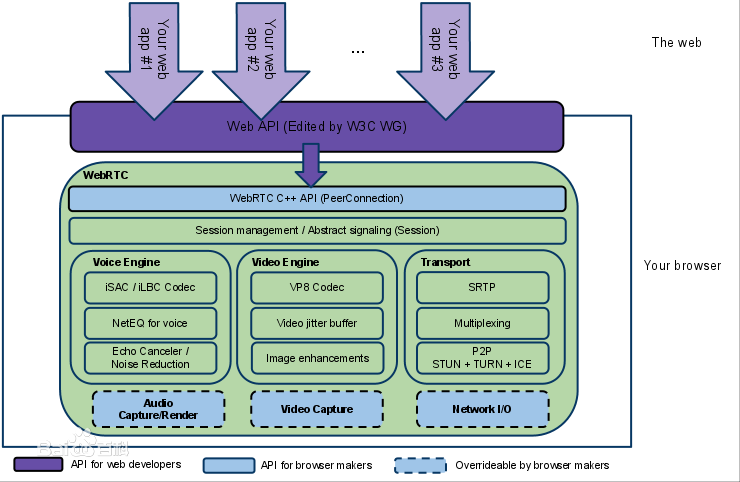
WebRTC/OpenWebRTC：

WebRTC，名称源自网页实时通信（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技术；2011年5月开放了工程的源代码，在行业内得到了广泛的支持和应用，成为下一代视频通话的标准。

WebRTC实现了基于网页的视频会议，标准是WHATWG 协议，目的是通过浏览器提供简单的javascript就可以达到实时通讯（Real-Time Communications (RTC)）能力。

WebRTC项目的最终目的主要是让Web开发者能够基于浏览器（Chrome\FireFox\...）轻易快捷开发出丰富的实时多媒体应用，而无需下载安装任何插件，Web开发者也无需关注多媒体的数字信号处理过程，只需编写简单的Javascript程序即可实现，W3C等组织正在制定Javascript 标准API，另外WebRTC还希望能够建立一个多互联网浏览器间健壮的实时通信的平台，形成开发者与浏览器厂商良好的生态环境。同时，Google也希望和致力于让WebRTC的技术成为HTML5标准之一，可见Google布局之深远。

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。



PS:

WebRTC默认使用的是VP8,Google出品，如果需要去兼容H264，则需要自己去兼容，兼容参考<http://www.07net01.com/2015/08/912681.html>，或使用openWebRTC（兼容H264）参考<http://blog.csdn.net/freewebsys/article/details/46649667>

COTURN:

COTURN是GOOGLE使用GO语言实现的开源TURN服务器，在介绍TURN与STURN服务器之前，首先需要了解NAT：

在现实Internet网络环境中，大多数计算机主机都位于防火墙或NAT之后，只有少部分主机能够直接接入Internet。很多时候，我们希望网络中的两台主机能够直接进行通信，即所谓的P2P通信，而不需要其他公共服务器的中转。由于主机可能位于防火墙或NAT之后，在进行P2P通信之前，我们需要进行检测以确认它们之间能否进行P2P通信以及如何通信。这种技术通常称为NAT穿透（NAT Traversal）。最常见的NAT穿透是基于UDP的技术，如RFC3489中定义的STUN协议。

NAT：

NAT有4种不同的类型

* 完全锥形NAT（Full Cone）

这种NAT内部的机器A连接过外网机器C后,NAT会打开一个端口.然后外网的任何发到这个打开的端口的UDP数据报都可以到达A.不管是不是C发过来的.  
例如 A:192.168.8.100 NAT:202.100.100.100 C:292.88.88.88  
A(192.168.8.100:5000) -> NAT(202.100.100.100:8000) -> C(292.88.88.88:2000)  
任何发送到 NAT(202.100.100.100:8000)的数据都可以到达A(192.168.8.100:5000)

* 限制锥形NAT（Address Restricted Cone）

这种NAT内部的机器A连接过外网的机器C后,NAT打开一个端口.然后**C可以用任何端口**和A通信.其他的外网机器不行.  
例如 A:192.168.8.100 NAT:202.100.100.100 C:292.88.88.88  
A(192.168.8.100:5000) -> NAT(202.100.100.100 : 8000) -> C(292.88.88.88:2000)  
任何从C发送到 NAT(202.100.100.100:8000)的数据都可以到达A(192.168.8.100:5000)

* 端口限制锥形NAT（Port Restricted Cone）

这种NAT内部的机器A连接过外网的机器C后,NAT打开一个端口.然后**C可以用原来的端口**和A通信.其他的外网机器不行.  
例如 A:192.168.8.100 NAT:202.100.100.100 C:292.88.88.88  
A(192.168.8.100:5000) -> NAT(202.100.100.100 : 8000) -> C(292.88.88.88:2000)  
C(202.88.88.88:2000)发送到 NAT(202.100.100.100:8000)的数据都可以到达A(192.168.8.100:5000)

注意：**以上三种NAT通称Cone NAT(圆锥形NAT).我们只能用这种NAT进行UDP打洞.**

* Symmetric（对称形）

对于这种NAT。连接不同的外部Server，NAT打开的端口会变化。也就是内部机器A连接外网机器B时，NAT会打开一个端口，连接外网机器C时又会打开另外一个端口。

对于双方都是Port Restricted Cone NAT的时候，则需要利用UDP打洞原理进行“先打洞，然后才能直接通信”。  
对于Cone NAT.要采用UDP打洞.需要一个公网机器server C来充当”介绍人”.处于NAT之后的内网的A,B先分别和C通信,打开各自的NAT端口.C这个时候知道A,B的公网IP: Port. 现在A和B想直接连接.比如A给B直接发包，除非B是Full Cone，否则不能通信.反之亦然.  
为什么啊？因为对于处于NAT之后的A,B。如果想A要与外界的D通信，则首先必须要A发包到D，然后A经过NAT设备NA，NA把A的内网地址和端口转换为NA的外网地址和端口。和D通信之后，D才能经过NA和A通信。也就是说，只能A和外界主动通信，外界不能主动和处于NA之后的A通信。这种包会被NA直接丢弃的。这也就是上面所说的Port Restricted Cone 的情形啊！ A(192.168.8.100:5000) -> NA(202.100.100.100:8000) -> D(292.88.88.88:2000)但是我们可以这样.  
A --- NA --- Server C --- NB --- B

1. A,B 为主机；
2. NA, NB 为NAT设备；
3. Server C为外网的机器；
4. 如果A想与B通信；
5. A首先连接 C, C得到A的外网NA的地址和端口；
6. B也要连接C，C得到B的外网NB的地址和端口；
7. A告诉C说我要和B通讯；
8. C通过NB发信息给B，告诉B A的外网NA的地址和端口；
9. B向NA发数据包（肯定会被NA丢弃，因为NA上并没有 A->NB 的合法session），但是NB上就建立了有B->NA的合法session了；
10. B发数据包给C，让 C 通知 A，我已经把洞打好了；
11. A接受到通知后向 B 的外网发NB数据包，这样就不会被丢弃掉了。因为对于NB来说，它看到的是A的外网NA的地址，而通过第6步，B已经让NA成为NB的合法通信对象了。所以当NA发数据包给NB时，NB就会接收并转发给B；

注意： 路由器和防火墙的UDP打洞的端口有个时间限制的，在一定时间内如果没有数据通讯会自动关闭

STUN，首先在RFC3489中定义，作为一个完整的NAT穿透解决方案，英文全称是Simple Traversal of UDP Through NATs，即简单的用UDP穿透NAT。

在新的RFC5389修订中把STUN协议定位于为穿透NAT提供工具，而不是一个完整的解决方案，英文全称是Session Traversal Utilities for NAT，即NAT会话穿透效用。RFC5389与RFC3489除了名称变化外，最大的区别是支持TCP穿透。

TURN，首先在RFC5766中定义，英文全称是Traversal Using Relays around NAT:Relay Extensions to Session Traversal Utilities for NAT，即使用中继穿透NAT:STUN的扩展。简单的说，TURN与STURN的共同点都是通过修改应用层中的私网地址达到NAT穿透的效果，异同点是TURN是通过两方通讯的“中间人”方式实现穿透。