## 信令与RTCPeerConnection建立

在前一篇文章中介绍过，WebRTC使用RTCPeerConnection来在浏览器之间传递流数据，在建立RTCPeerConnection实例之后，想要使用其建立一个点对点的信道，我们需要做两件事：  
1. 确定本机上的媒体流的特性，比如分辨率、编解码能力啥的（SDP描述符）  
2. 连接两端的主机的网络地址（ICE Candidate）

需要注意的是，由于连接两端的主机都可能在内网或是在防火墙之后，我们需要一种对所有联网的计算机都通用的定位方式。这其中就涉及NAT/防火墙穿越技术，以及WebRTC用来达到这个目的所ICE框架。这一部分在上一篇文章中有介绍，这里不再赘述。

### 通过offer和answer交换SDP描述符

大致上在两个用户（甲和乙）之间建立点对点连接流程应该是这个样子（这里不考虑错误的情况，RTCPeerConnection简称PC）：  
1. 甲和乙各自建立一个PC实例  
2. 甲通过PC所提供的createOffer()方法建立一个包含甲的SDP描述符的offer信令  
3. 甲通过PC所提供的setLocalDescription()方法，将甲的SDP描述符交给甲的PC实例  
4. 甲将offer信令通过服务器发送给乙  
5. 乙将甲的offer信令中所包含的的SDP描述符提取出来，通过PC所提供的setRemoteDescription()方法交给乙的PC实例  
6. 乙通过PC所提供的createAnswer()方法建立一个包含乙的SDP描述符answer信令  
7. 乙通过PC所提供的setLocalDescription()方法，将乙的SDP描述符交给乙的PC实例  
8. 乙将answer信令通过服务器发送给甲  
9. 甲接收到乙的answer信令后，将其中乙的SDP描述符提取出来，调用setRemoteDescripttion()方法交给甲自己的PC实例

通过在这一系列的信令交换之后，甲和乙所创建的PC实例都包含甲和乙的SDP描述符了，完成了两件事的第一件。我们还需要完成第二件事——获取连接两端主机的网络地址

### 通过ICE框架建立NAT/防火墙穿越的连接

这个网络地址应该是能从外界直接访问，WebRTC使用ICE框架来获得这个地址。RTCPeerConnection在创立的时候可以将ICE服务器的地址传递进去，如：

var iceServer = {

"iceServers": [{

"url": "stun:stun.l.google.com:19302"

}]

};var pc = new RTCPeerConnection(iceServer);

当然这个地址也需要交换，还是以甲乙两位为例，交换的流程如下（RTCPeerConnection简称PC）：  
1. 甲、乙各创建配置了ICE服务器的PC实例，并为其添加onicecandidate事件回调  
2. 当网络候选可用时，将会调用onicecandidate函数  
3. 在回调函数内部，甲或乙将网络候选的消息封装在ICE Candidate信令中，通过服务器中转，传递给对方  
4. 甲或乙接收到对方通过服务器中转所发送过来ICE Candidate信令时，将其解析并获得网络候选，将其通过PC实例的addIceCandidate()方法加入到PC实例中

这样连接就创立完成了，可以向RTCPeerConnection中通过addStream()加入流来传输媒体流数据。将流加入到RTCPeerConnection实例中后，对方就可以通过onaddstream所绑定的回调函数监听到了。调用addStream()可以在连接完成之前，在连接建立之后，对方一样能监听到媒体流