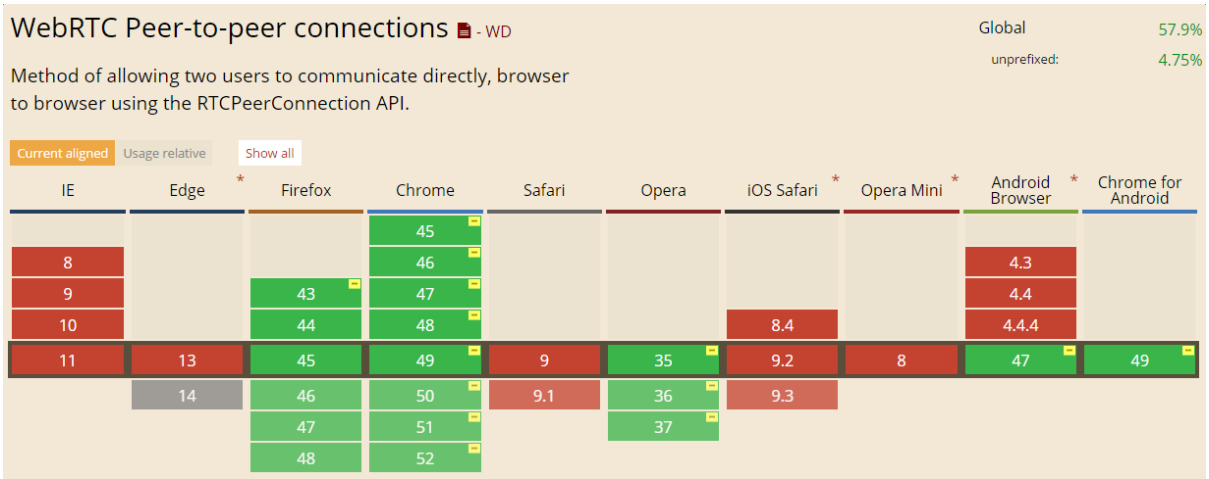
使用 Wilddog 搭建本地视频聊天室

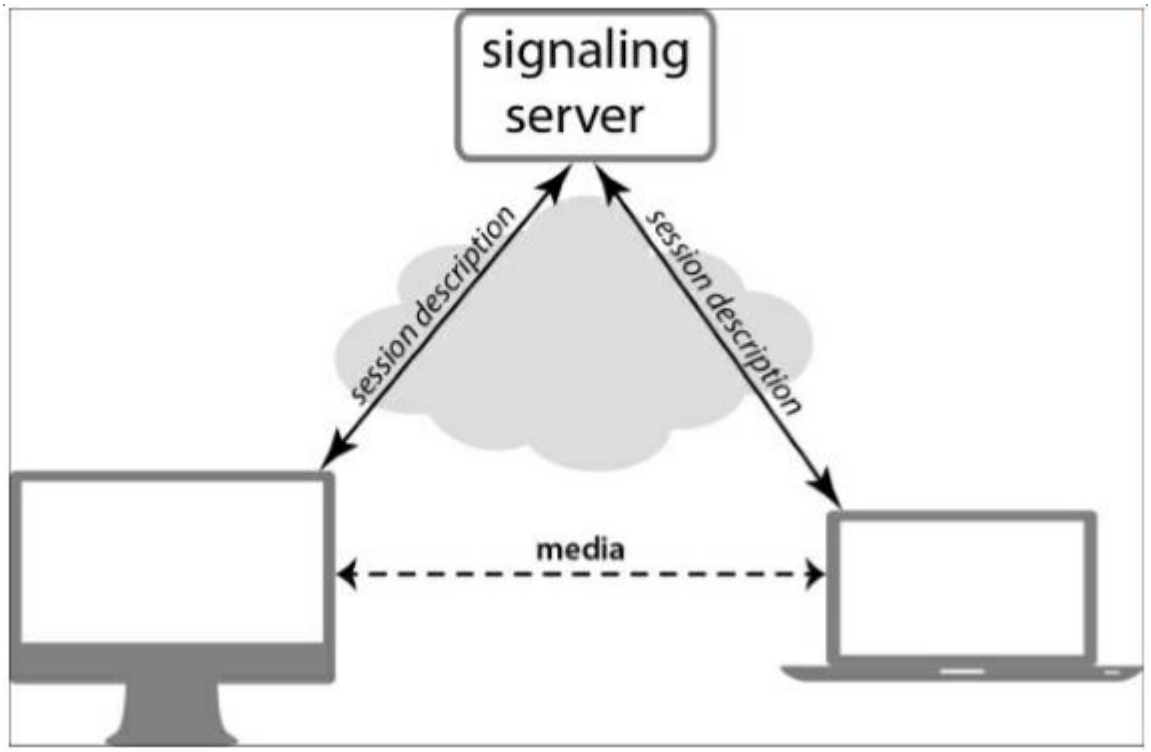
-- 基于 WebRTC 的技术实践

一直以来，实时视频聊天开发因为涉及音视频编解码、P2P 穿越等许多技术，同时还需要客户端以及浏览器插件的支持，都是大公司的专利。WebRTC（ http://webrtc.org/） 是 google 主推的一个开源项目，旨在使得浏览器能为实时通信（RTC）提供简单的 JavaScript 接口。有了它，直接通过浏览器的 Web 页面就可以实现音视频聊天功能，无需任何插件。

从著名的浏览器兼容性查询网站 http://caniuse.com/可以查询到各大浏览器对 WebRTC 的支持情况；其中，Edge 浏览器支持类似 WebRTC 的 ORTC。可以说，从市场占有率来看，大部分场景下都能够使用 WebRTC 进行通信。



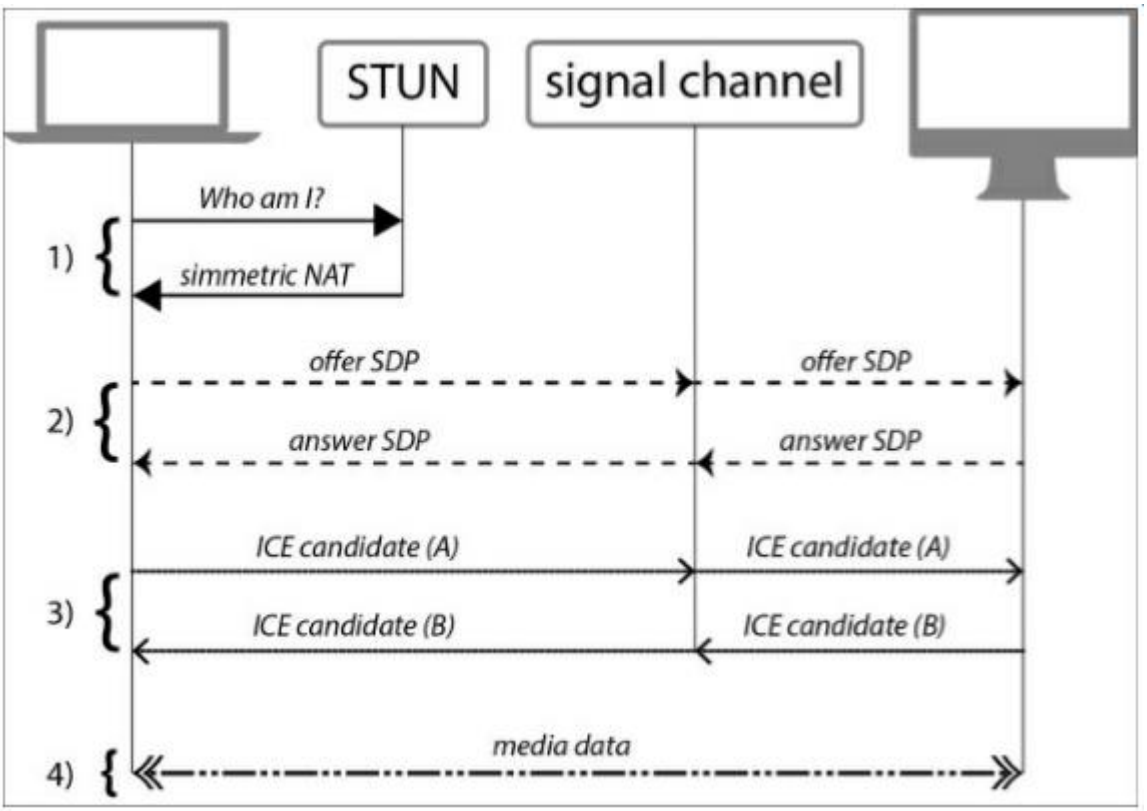
1. 基本概念



这是一个简单的 WebRTC 三角关系图。图中，两台电脑 A 和 B 想要建立实时音视频聊天。由于它们在不同的内网，因此无法直接建立音视频管道。它们首先连接到一个服务器，WebRTC 称之为信令服务器（signal server），通过信令服务器，二者交换建立管道的一些必要信息（信令，采用 SDP 协议编码），如音视频格式、帧率，二者的 ip 地址等。

收到对方信令后，它们分别将其交给 WebRTC 底层，由底层实现二者间管道的建立。管道建立完成后，数据直接通过管道交互，无需再经过信令服务器。

2. 交互过程



它们之间交互的流程如上图：

1. 电脑 A 获知自己的外网 ip 和端口，并通过信令服务器向电脑 B 发送请求（offer 信令）；

2. 电脑 B 收到 A 的请求后，处理 A 的 offer 信令，并将结果（answer 信令）通过信令服务器返回给 A；

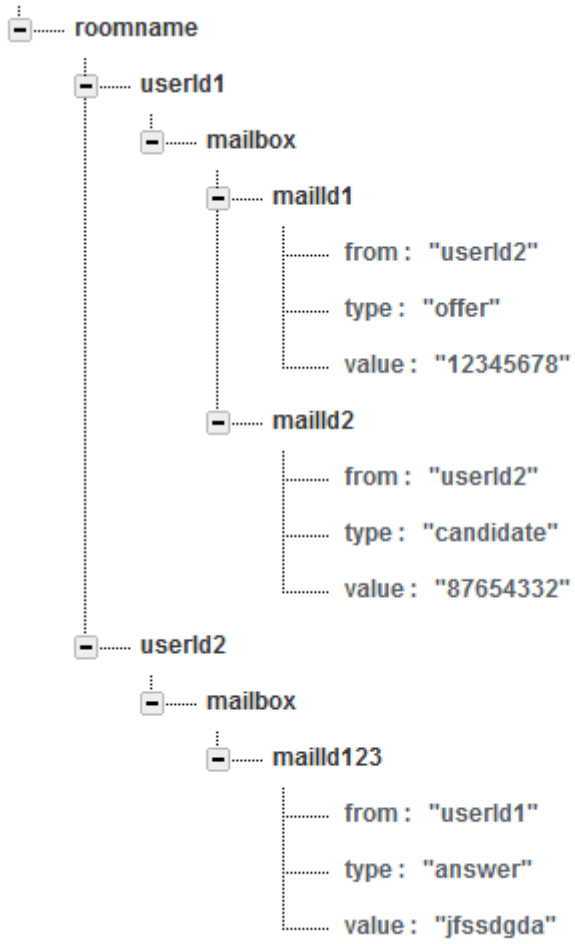
3. 在此过程中，WebRTC 底层同时会生成很多 candidate（候选项）信息，candidate 信息包括电脑的内网 IP，外网 IP 等。电脑 A 和 B 将生成的 candidate 也通过信令服务器传递给对方；

4. WebRTC 底层根据这些信息，建立管道。

A 和 B 之间的数据交换，需要公网服务器来做中转。对大部分个人爱好者来说，搭建自己的公网服务器并不现实。我们注意到，A 和 B 的数据交换实质上就是二者数据同步的过程，而实时数据同步可是 Wilddog 的强项，那么通过 Wilddog 就可以实现 A 和 B 的数据交换，无需搭建公网服务器。

3. 使用 Wilddog 建立视频聊天室

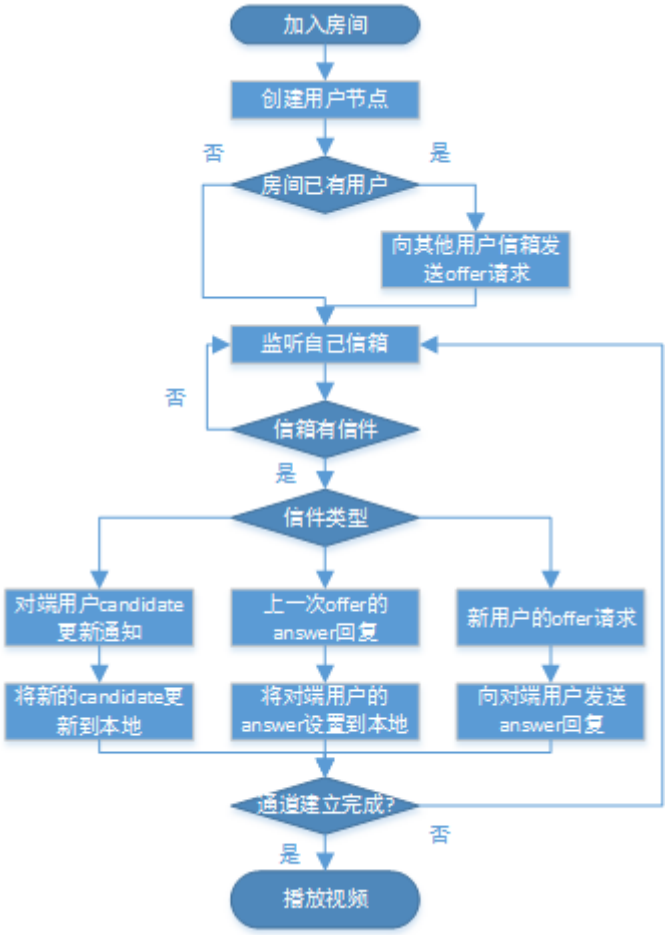
为了方便处理，我们设计下图的结构：



roomname 为房间名称，每个用户进入房间后，在房间下面根据自己的 userId 创建子节点，如果有向其他用户通信的需求，则向其他用户的 mailbox 发送信件。同时，用户会监听自身的 mailbox，当发现 mailbox 中来了新的信件，则处理这封信件，需要回复的话则向对方的 mailbox 回信。

我们将 offer、answer、candidate 这几种信令都抽象成信件，通过 Wilddog 的实时同步，将信件同步给对方。

整体流程图如下：



让我们以电脑 A 和 B 来举例：

1. A 和 B 打开浏览器后都调用 getUserMedia 获取本地视频流，并在浏览器上播放；

2. B 先进入房间，此时房间中没有其他用户，B 在房间下建立属于自己的节点，且监听自己的信箱；

3. A 进入房间，发现 B 已存在，此时 A 调用 createOffer 生成 offer 信令，并使用 setLocalDescription 存储到本地，然后向 B 的节点下面的信箱发送 offer，同时，A 也监听自己的信箱；

4. B 发现信箱下有信件，查知此信为 A 的 offer，B 调用 setRemoteDescription 将其存储到本地，然后使用 createAnswer 生成 answer 信令，使用 setLocalDescription 存储到本地，然后向 A 的信箱发送 answer；

5. A 发现信箱下有信件，查知此信为 B 的 answer，A 调用 setRemoteDescription 将其存储到本地；

6. 与此同时，A 和 B 都可能触发 onicecandidate 事件，WebRTC 底层会将收集的 candidate 信令报告给上层，这种信令也需要向对方的信箱发送，对方收到这类信件后，调用 addCandidate 将之保存到本地；

7. 在收集足够多的信令后，WebRTC 底层会将 A 和 B 之间的管道打通，A 和 B 会触发 onaddstream 事件，事件的回调函数中，会收到对方的 stream 流，可以用来在浏览器端播放。

4. 实现源码

https://github.com/itolfh/demo-webrtc-onepeer，这段源码实现了一对一视频聊天，多人聊天改动也不会太大，就由各位实现吧。

注意：chrome 的 getUserMedia 接口要求 html 页面必须使用 https 服务，不能使用 http 服务或者本地打开；而使用 firefox 浏览器则无此限制，因此如果采用 firefox，可以将网页传给好友并本地打开，两边输入相同房间号，即可以进行实时音视频聊天；chrome 则需两边分别搭建本地 https 服务。

5. 总结

使用 Wilddog，我们轻松解决了信令服务器的问题，无需服务器，也能够搭建本地视频聊天室了。然而大家可能会好奇，两个在不同内网的电脑，是如何能够建立管道的？candidate 中的 ip 信息，包括什么？又是从哪获取的呢？我们将在后面的文章中解答大家的疑惑。