文 使用WebRTC搭建前端视频聊天室——点对点通信篇 (/a/1190000000733774)

websocket (/t/websocket/blogs) webim (/t/webim/blogs) webrtc (/t/webrtc/blogs) java (/t/java/blogs) node.js (/t/node.js/blogs)

天镶 (/u/lingyucoder) 2014年10月21日发布

WebRTC给我们带来了浏览器中的视频、音频聊天体验。但个人认为，它最实用的特性莫过于DataChannel——在浏览器之间建立一个点对点的数据通道。在DataChannel之前，浏览器到浏览器的数据传递通常是这样一个流程：浏览器1发送数据给服务器，服务器处理，服务器再转发给浏览器2。这三个过程都会带来相应的消耗，占用服务器带宽不说，还减缓了消息从发送到接收的时间。其实最理想的方式就是浏览器1直接与浏览2进行通信，服务器不需要参与其中。WebRTC DataChannel就提供了这样一种方式。

如果对WebRTC和DataChannel不太了解的同学，可以先阅读如下文章：
- WebRTC的RTCDataChannel (http://lingyu.wang…
- 使用WebRTC搭建前端视频聊天室——信令篇 (ht…
- 使用WebRTC搭建前端视频聊天室——入门篇 (ht…

老刘和老姚

当然服务器完全不参与其中，显然是不可能的，用户需要通过服务器上存储的信息，才能确定需要和谁建立连接。这里通过一个故事来讲述建立连接的过程：

不如钓鱼去

一些背景：
- 老刘和老姚都住在同一个小区但不同的片区，小区很破旧，没有电话
- 片区相互隔离且片区门口有个保安，保安只认识自己片区的人，遇到不认识的人就需要查询凭证才能通过，而凭证需要找物业才能确定
- 门卫老大爷认识小区里的所有人但是不知道都住哪，有什么消息都可以在出入小区的时候代为传达

现在，老刘听说老姚钓鱼技术高超，想和老姚讨论钓鱼技巧。只要老刘和老姚相互之间知道对方的门牌号以及凭证，就可以串门了:

1. 门卫老大爷认识老刘和老姚
2. 老刘找物业确定了自己片区的出入凭证，将凭证、自己的门牌号以及意图告诉门卫老大爷，让其转交给老姚
3. 老姚买菜归来遇到门卫老大爷，门卫老大爷将老刘的消息传达给老姚。于是老姚知道怎么去老刘家了
4. 老姚很开心，他也找物业获取了自己小区的凭证，并将凭证、自己的门牌号等信息交给门卫老大爷，希望

他传达给老刘

5. 老刘吃早餐回来遇到门卫老大爷，老大爷把老姚的小区凭证、门牌号等信息告诉老刘，这样老刘就知道了怎么去老姚家了

老刘和老姚相互之间知道了对方的门牌号和小区出入凭证，他们相互之间有什么需要交流的直接串门就行了，消息不再需要门卫老大爷来代为传达了

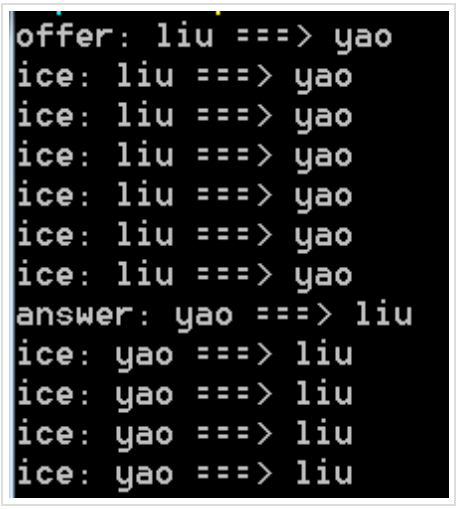
换个角度

我们把角色做一个映射：
- 老刘：浏览器1
- 老姚：浏览器2
- 片区：不同网段
- 保安：防火墙
- 片区凭证：ICE candidate
- 物业：ICE server
- 门牌号：session description
- 门卫老大爷：server

于是乎故事就变成了这样：

1. 浏览器1和浏览器2在server上注册，并保有连接
2. 浏览器1从ice server获取ice candidate并发送给server，并生成包含session description的offer，发送给server
3. server发送浏览器1的offer和ice candidate给浏览器2
4. 浏览器2发送包含session description的answer和ice candidate给server
5. server发送浏览器2的answer和ice candidate给浏览器1

这样，就建立了一个点对点的信道，流程如下所示：



礼物

故事

老刘和老姚已经可以相互串门了，经过一段时间的交流感情越来越深。老姚的亲友送了20斤葡萄给老姚，老姚决定送10斤给老刘。老姚毕竟年事已高，不可能一次带10斤。于是乎，老姚将葡萄分成了10份，每次去老刘家串门就送一份过去。

这里可以做如下类比：
1. 10斤葡萄：一个文件（尽管文件分片没有意义，葡萄分开还可以单独吃，但是实在找不到啥好的比喻了）
2. 分成10份：将文件分片，转成多个chunk
3. 老姚一次只能带一斤：datachannel每次传输的数据量不宜太大（
找到最合适的大小 (http://stackoverflow.com/que… ）

这其实就是通过datachannel传输文件的方式，首先将文件分片，然后逐个发送，最后再统一的进行组合成一个新的文件

分片

通过HTML5的File API可以将type为file的input选中的文件读取出来，并转换成data url字符串。这也就为我们提供了很方便的分片方式：

```
var reader = new window.FileReader(file);
reader.readAsDataURL(file);
reader.onload = function(event, text) {
    chunkify(event.target.result);//将数据分片
};
```

组合

通过datachannel发送的分片数据，我们需要将其进行组合，由于是data url字符串，在接收到所有包之后进行拼接就可以了。拼接完成后就得到了一个文件完整的data url字符串，那么我们如何将这个字符串转换成文件呢？

方案一：直接跳转下载

既然是个dataurl，我们直接将其赋值给 `window.location.href` 自然可以下载，但是这样下载是没法设定下载后的文件名的，这想一想都蛋疼

方案二：通过a标签下载

这个原理和跳转下载类似，都是使用dataurl本身的特性，通过创建一个a标签，将dataurl字符串赋值给href属性，然后使用download确定下载后的文件名，就可以完成下载了。但是很快又有新问题了，稍微大一点的文件下载的时候页面崩溃了。这是因为dataurl有大小限制 (http://stackoverflow.com/questions/69…

方案三：blob

其实可以通过给a标签创建blob url的方式来进行下载，这个没有大小限制。但是我们手上是dataurl，所以需要先进行转换：

```
function dataURItoBlob(dataURI, dataTYPE) {
    var binary = atob(dataURI.split(',')[1]),
        array = [];
    for (var i = 0; i < binary.length; i++) array.push(binary.charCodeAt(i));
    return new Blob([new Uint8Array(array)], {
        type: dataTYPE
    });
}
```

获得blob后，我们就可以通过URL API来下载了：

```
var a = document.createElement("a");
document.body.appendChild(a);
a.style = "display: none";
var blob = dataURItoBlob(data, 'octet/stream');
var url = window.URL.createObjectURL(blob);
a.href = url;
a.download = filename;
a.click();
!moz && window.URL.revokeObjectURL(url);
a.parentNode.removeChild(a);
```

这里有几个点：
1. datachannel其实是可以直接传送blob的，但是只有ff支持，所以传data url
2. chrome下载是直接触发的，不会进行询问，firefox会先询问后下载，在询问过程中如果执行了 `revokeObjectURL` ，下载就会取消，囧

升级

如我们所知，WebRTC最有特点的地方其实是可以传输getUserMedia获得的视频、音频流，来实现视频聊天。但事实上我们的使用习惯来看，一般人不会一开始就打开视频聊天，而且视频聊天时很消耗内存的（32位机上一个连接至少20M左右好像，也有可能有出入）。所以常见的需求是，先建立一个包含datachannel的连接用于传输数据，然后在需要时升级成可以传输视频、音频。

看看我们之前传输的session description，它其实来自Session Description Protocol (http://datatracker.i…。可以看到wiki上的介绍：

> The Session Description Protocol (SDP) is a format for describing streaming media initialization parameters.

这意味着什么呢？我们之前建立datachannel是没有加视频、音频流的，而这个流的描述是写在SDP里面的。现在我们需要传输视频、音频，就需要添加这些描述。所以就得重新获得SDP，然后构建offer和answer再传输一次。传输的流程和之前一样，没什么区别。但这一次，我们不需要传输任何的ice candidate，这里我曾经遇到了坑 (http://stackoverflow.com/questions/26… ，经过国外大大的点拨才明白过来。

> from mattm: You do not need to send ICE candidates on an already established peer connection. The ICE candidates are to make sure the two peers can establish a connection through their potential NAT and firewalls. If you can already send data on the peer connection, ICE candidates will not do anything.

Peertc

我将datachannel和websocket组合，实现了一个构建点对点连接的库Peertc，它提供非常简洁的方式来建立连接和发送数据、文件和视频/音频流，详情见github (https://github.com/LingyuCoder/peertc)。走过路过的记得star一下哦，有什么bug也非常希望能够提出来。

最后

WebRTC的点对点方式能够运用在很多场景：
- 如web qq这种Web IM工具，这就不说了
- 如象棋这种双人对战游戏，每一步的数据服务器时不关心的，所以完全可以点对点发送
- 一对一在线面试、在线教育，这其实是即时通信的一个业务方向
- 视频裸（ ），当我没说

就酱，另外打个广告及拉点搜索引擎权重：我的博客 (http://lingyu.wang)

2014年10月21日发布 (/a/1190000000733774)

0 推荐 收藏

你可能感兴趣的文章

使用WebRTC搭建前端视频聊天室——入门篇 (/a/1190000000436544) 87 收藏，42.1k 浏览

WebRTC 工作流程 (/a/1190000000608413) 1 收藏，1.9k 浏览

使用WebRTC搭建前端视频聊天室——数据通道篇 (/a/1190000000733779) 25 收藏，9.3k 浏览

讨论区

请先 登录 后评论

本文隶属于专栏

说学逗唱 (/blog/skyinlayer)

天镶 (/u/lingyucoder)
作者

关注专栏

系列文章

使用WebRTC搭建前端视频聊天室——数据通道篇 (/a/1190000000733779) 25 收藏， 9.3k 浏览

分享扩散：

•••

Copyright © 2011-2016 SegmentFault. 当前呈现版本 16.02.02
浙ICP备15005796号-2 (http://www.miibeian.gov.cn/)
移动版桌面版