

23 | 实战演练：通过WebRTC实现一个1对1音视频实时直播系统

2019-09-05 李超

从0打造音视频直播系统

[进入课程 >](#)



讲述：李超

时长 15:36 大小 14.29M



在前面的 22 篇文章中，我分步骤向你介绍了如何在浏览器上实现一套简单的直播系统。比如，如何采集音视频数据，如何在同一个浏览器上进行音视频通话，如何传输非音视频数据，等等。

但这些知识点都是一个独立的，并没有形成一个完整的系统，那么本篇文章我们就将所有这些知识点串联起来，做成一个完整的系统。当这套系统搭建完成之后，你就可以在浏览器上实现从北京到上海这种远距离的实时通信了。

接下来我们就来详述这套系统的实现步骤（运行环境的搭建，Web 服务器的实现，信令系统的实现，TURN 服务器的搭建，RTCPeerConnection 的创建，音视频数据的采集、传输、渲染与播放）。通过这部分内容的讲解，再配合我们之前的文章，相信你一定可以快速搭建出这样一套系统来。

运行环境搭建

要实现这套系统，运行环境是特别关键的，其中有三个条件是必须要满足的：**云主机、2M 以上的带宽和 HTTPS 证书**。

为什么需要这三个条件呢？下面我们就来详细看一下。

云主机就不必多说了，要实现远距离的音视频通信，就得进行信令的交换和数据的中转，这些都需要我们有一台云主机。换句话说，有了云主机后，通信的双方才可以进行信令的交换；在 P2P 不通的情况下，通过服务器中转的方式才能保障端与端之间的连通性。

另外，在实现这套系统时，对云主机的性能要求并不高，2 核的 CPU、1G/2G 的内存就够了。但对**带宽**的高求就比较高了，至少要 2M 以上的带宽，而且是越多越好。

需要 2M 以上带宽的原因我们之前就分析过了，对于一路 720P 的视频，一般情况下需要 1.2M 的带宽，因此在两人通信时，如果都走服务器中转的话就需要 2.4M 的带宽了。所以为了保证双方通信顺畅，带宽是必须要能跟得上的。


第三个必要条件是 HTTPS 证书。那为什么需要这个条件呢？出于安全的原因，浏览器要求我们使用 HTTPS 协议从服务器请求 JavaScript 脚本，只有通过 HTTPS 请求的脚本才能访问音视频设备。这是因为通过 HTTPS 访问的服务都是通过安全机构认证过的，这样就保证了音视频设备被访问的安全性。而 HTTP 协议就做不到这一点（因为没有任何安全机构对它们做认证），为了保证用户不会在自己不知情的情况下被偷拍或被录音，**因此在浏览器上通过 HTTP 请求下来的 JavaScript 脚本是不允许访问音视频设备的**。

我曾见过很多同学抱怨使用 HTTP 请求的 JavaScript 脚本不能访问音视频设备的问题，其实这就是因为他们没有理解隐私安全，也没有站在用户的角度去思考问题。

最后，额外说明一下我个人的一些关于云主机的使用心得吧，云主机的操作系统最好选择使用 **Ubuntu**，并且**版本越新越好**，这是因为在 Ubuntu 上边安装依赖库比较方便。另外，在国内申请域名时还要进行备案，而备案时间需要 10 天以上，非常繁琐。所以如果你特别着急的话，可以购买国外的云主机和域名，一般可以免费使用一年，并且在国外购买域名不需要进行备案，这样可以节省不少时间。

实现 Web 服务器

对于这部分内容我们在[《11 | 如何通过 Node.js 实现一套最简单的信令系统？》](#)一文中已经向你做过介绍了，但没有讲 HTTPS 的部分。实际上，在 Node.js 中 HTTPS Server 的实现与 HTTP Server 的实现是类似的，我们来看一下具体的代码：

 复制代码

```
1 var https = require('https');
2 var express = require('express');
3 var serveIndex = require('serve-index');
4 ...
5 // 使用 express 实现 WEB 服务
6 var app = express();
7 app.use(serveIndex('./public'));
8 app.use(express.static('./public'));
9
10 //HTTPS 证书和密钥文件
11 var options = {
12   key : fs.readFileSync('./cert/1557605_www.learningrtc.cn.key'),
13   cert: fs.readFileSync('./cert/1557605_www.learningrtc.cn.pem')
14 }
15
16 //https server
17 var https_server = https.createServer(options, app);
18 var io = socketIo.listen(https_server);
19 https_server.listen(443, '0.0.0.0');
20 ...
```

上面的代码中引入了 express 库，它的功能非常强大，用它来实现 Web 服务器非常方便。上面的代码同时还引入 HTTPS 服务，并让 Web 服务运行于 HTTPS 之上，这样 HTTPS 的 Web 服务器就实现好了。

有了 HTTPS 的 Web 服务器之后，我们就可以将客户端代码（如 HTML、CSS、JavaScript）放到 Web 服务的 public 目录下，这样在通过域名访问时，浏览器就将客户端代码下载下来，并渲染到浏览器上，然后我们就可以从浏览器上看到用户界面了。

最简单的信令系统

信令是系统的“灵魂”。在我们要实现的这个直播系统中，由谁来发起呼叫、什么时间发 SDP 等各种操作都是由信令控制的。

我们这个直播系统的信令相对还是比较简单的，其客户端和服务端的信令可大致总结为如下几种。

客户端命令

join，用户加入房间。

leave，用户离开房间。

message，端到端命令（offer、answer、candidate）。

服务端命令

joined，用户已加入。

leaved，用户已离开。

other_joined，其他用户已加入。

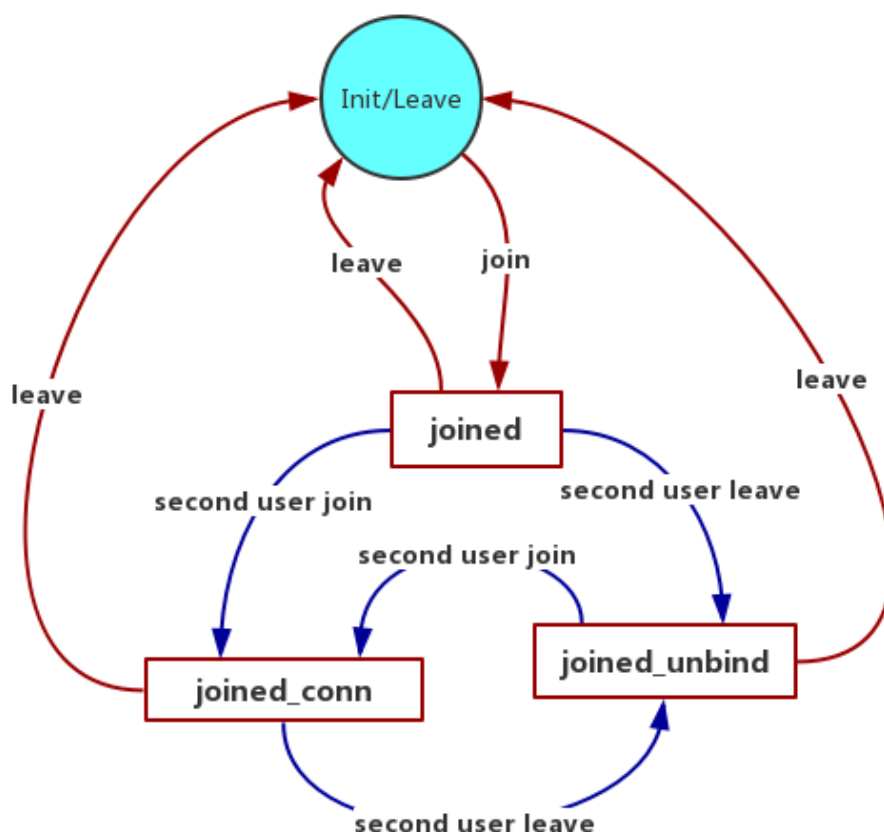
bye，其他用户已离开。

full，房间已满。

那这些信令之间是怎样一种关系呢？在什么情况下该发送怎样的信令呢？要回答这个问题我们就要看一下信令状态机了。

信令状态机

我们这个系统的信令是由一个信令状态机来管理的，在不同的状态下，需要发送不同的信令。当收到服务端或对端的信令后，状态也会随之发生改变，下面我们来看一下这个状态的变化图吧。



信令状态机图

在初始时，客户端处于 init/leaved 状态。在 init/leaved 状态下，用户只能发送 join 消息。服务端收到 join 消息后，会返回 joined 消息。此时，客户端会更新为 joined 状态。

在 joined 状态下，客户端有多种选择，收到不同的消息会切到不同的状态。

如果用户离开房间，那客户端又回到了初始状态，即 init/leaved 状态。

如果客户端收到 second user join 消息，则切换到 joined_conn 状态。在这种状态下，两个用户就可以进行通话了。

如果客户端收到 second user leave 消息，则切换到 joined_unbind 状态。其实 joined_unbind 状态与 joined 状态基本是一致的。

如果客户端处于 joined_conn 状态，当它收到 second user leave 消息时，会转成 joined_unbind 状态。

如果客户端是 `joined_unbind` 状态，当它收到 `second user join` 消息时，会切到 `joined_conn` 状态。

通过上面的状态图，我们就可以清楚地知道在什么状态下应该发什么信令；或者说，发什么样的信令，信令状态就会发生怎样的变化了。

信令系统的实现

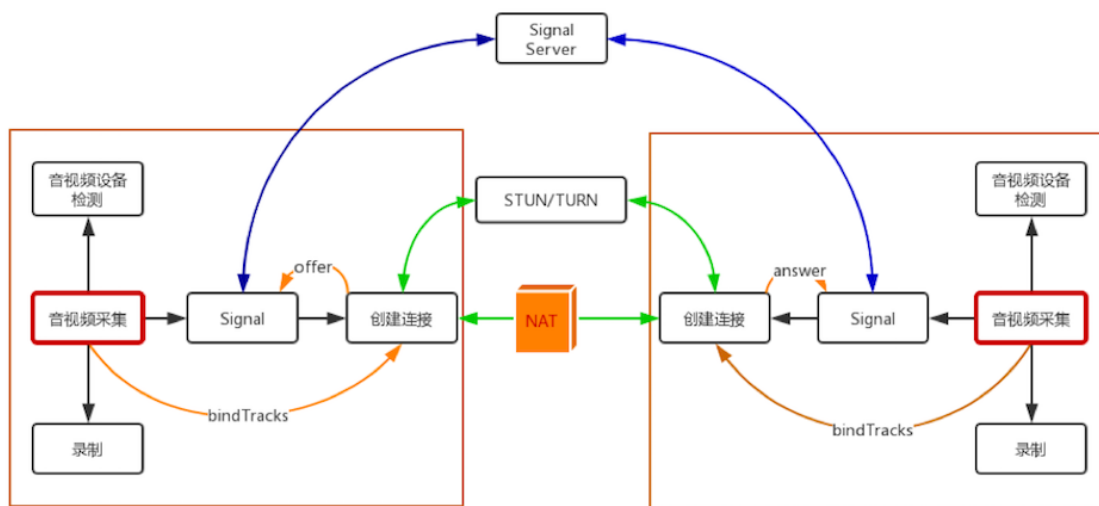
在我们这个系统中，信令的实现还是通过 “Node + socket.io” 来实现的。上面我们已经详细介绍了信令的状态是如何变化的，再结合 [《11 | 如何通过 Node.js 实现一套最简单的信令系统？》](#) 一文中介绍的如何通过 socket.io 实现信令服务器，这样你就可以很轻松地将信令系统实现出来了。

这里我就不将详细的代码列出来了，在文章的最后我会将本文的示例代码展示出来，你可以参考它来实现。

TURN 服务器的搭建

在之前的文章中，有一个重要的知识点一直没介绍，那就是 TURN 服务。它有两个作用，一是提供 STUN 服务，客户端可以通过 STUN 服务获取自己的外网地址；二是提供数据中继服务。

当通信的双方无法通过 P2P 进行数据传输时，为了保证数据的连通性，我们就必须使用中继的方式让通信双方的数据可以互通，如下图所示：



WebRTC 处理过程图

目前最著名的 TURN 服务器是由 Google 发起的开源项目 coturn，它的源码和项目描述可以在 GitHub 上找到，地址为：<https://github.com/coturn/coturn>。

coturn 的编译安装与部署还是比较简单的。首先我们来看看如何编译安装 coturn，其步骤如下：

从 <https://github.com/coturn/coturn> 下载 coturn 代码；

执行 `./configure --prefix=/usr/local/coturn` 生成 Makefile；

执行 `make` 来编译 coturn；

执行 `sudo make install` 进行安装。

接下来是部署，coturn 的配置文件中有很多选项，不过这些选项大部分都可以不用，或采用默认值，因此我们只需要修改 4 项内容就可以将 coturn 服务配置好了，这 4 项配置如下：

```
listening-port=3478      #指定侦听的端口
external-ip=39.105.185.198 #指定云主机的公网IP地址
user=aaaaaa:bbbbbb      #访问 stun/turn服务的用户名和密码
realm=stun.xxx.cn        #域名，这个一定要设置
```

这里需要注意的是，在网上能看到很多讲解 coturn 配置的文章，但其中大部分配置都是有问题的，所以建议你按照本文所讲的步骤进行配置，避免浪费你宝贵的时间。

音视频数据的采集

音视频数据采集的过程我们已经在 [《01 | 原来通过浏览器访问摄像头这么容易》](#) 一文中做过详细介绍了，**通过 getUserMedia 就可以获取到**，就不再赘述了。

不过需要注意的是，我们系统采集音视频数据的时间点与以前不一样了，以前是在浏览器显示页面时就开始采集了，而现在则是在用户点击 “Connect Sig Server” 按钮时才开始采集音视频数据。


创建 RTCPeerConnection

信令系统建立好后，后面的逻辑都是围绕着信令系统建立起来的，RTCPeerConnection 对象也不例外。

在客户端，用户要想与远端通话，首先要发送 join 消息，也就是要先进入房间。此时，如果服务器判定用户是合法的，则会给客户端回 joined 消息。

客户端收到 joined 消息后，就要创建 RTCPeerConnection 对象了，也就是要建立一条与远端通话的音视频数据传输通道。

下面，我们就结合示例代码来看一下 RTCPeerConnection 是如何建立的。

 复制代码

```
1 ...
2 var pcConfig = {
3   'iceServers': [{ // 指定 ICE 服务器信令
4     'urls': 'turn:stun.al.learningrtc.cn:3478', //turn 服务器地址
5     'credential': "passwd", //turn 服务器密码，你要用自己的
6     'username': "username" //turn 服务器用户名，你要用自己的
```



```

7   }]
8   };
9
10  ...
11
12  function createPeerConnection(){
13
14      if(!pc){
15          pc = new RTCPeerConnection(pcConfig); // 创建 peerconnection 对象
16          ...
17          pc.ontrack = getRemoteStream; // 当远端的 track 到来时会触发该事件
18      }else {
19          console.log('the pc have be created!');
20      }
21
22      return;
23  }
24
25  ...

```

上面这段代码需要注意的是创建 RTCPeerConnection 对象时的 pcConfig 参数，在该参数中我们设置了 TURN 服务器地址、用户名和密码，这样当 RTCPeerConnection 通过 P2P 建立连接失败时，就会使用 TURN 服务器进行数据中继。

RTCPeerConnection 对象创建好后，我们要将前面获取的音视频数据与它绑定到一起，这样才能通过 RTCPeerConnection 对象将音视频数据传输出去。绑定的步骤如下：

 复制代码

```

1  function bindTracks(){
2      ...
3      //add all track into peer connection
4      localStream.getTracks().forEach((track)=>{
5          pc.addTrack(track, localStream); // 将 track 与 peerconnection 绑定
6      });
7  }

```

在上面的代码中，从 localStream 中遍历所有的 track，然后添加到 RTCPeerConnection 对象中就好了。

以上工作完成后，接下来最重要的一步就是**媒体协商**了，对于媒体协商的过程我们已经在[《08 | 有话好商量，论媒体协商》](#)一文中做过详细分析了。具体的逻辑我就不再重复了，不过需要注意的是，在本文中媒体协商的消息都是通过信令进行交互的，具体的实现可以参考文末的示例代码。


音视频的渲染与播放

按照上面的步骤，音视频数据就可以被采集到了，RTCPeerConnection 对象也创建好了，通过信令服务器也可以将各端的 Candidate 交换完成了。

此时在 WebRTC 的底层就会进行连通性检测，它首先判断通信的双方是否在同一个局域网内，如果在同一个局域网内，则让双方直接进行连接；如果不在同一局域网内，则尝试用 P2P 连接，如果仍然不成功，则使用 TURN 服务进行数据中继。

一旦数据连通后，数据就从一端源源不断地传到了远端，此时远端只需要将数据与播放器对接，就可以看到对端的视频、听到对方的声音了。

在浏览器上实现这一点非常容易，当数据流过来的时候会触发 RTCPeerConnection 对象的 ontrack 事件，只要我们侦听该事件，并在回调函数中将收到的 track 与<video>标签绑定到一起就好了，代码如下：

 复制代码

```
1 var remoteVideo = document.querySelector('video#remotevideo');
2 ...
3 function getRemoteStream(e){ // 事件处理函数
4     remoteStream = e.streams[0]; // 保存远端的流
5     remoteVideo.srcObject = e.streams[0]; // 与 HTML 中的视频标签绑定
6 }
7 ...
8 pc = new RTCPeerConnection(pcConfig);
9 ...
10 pc.ontrack = getRemoteStream // 当远端的 track 过来时触发该事件
11 ...
```

通过上面的代码，我们可以看到，当 ontrack 事件触发时，会调用 getRemoteStream 函数，该函数从参数 e 中取出 stream 赋值给 remoteVideo (<video>标签) 的 srcObject 属性，这就可以了。

小结

以上就是浏览器端实现 1 对 1 实时通话的整体逻辑。在本文中，我通过对运行环境的搭建、Web 服务器的实现、信令系统的实现、TURN 服务器的搭建、RTCPeerConnection 的创建、音视频数据的采集、传输、渲染与播放这几个主题的介绍，向你完整地讲解了在浏览器上实现 1 对 1 实时音视频通话的过程。

在这几个主题中，运行环境的搭建、信令系统的实现、TURN 服务器的搭建是之前没有讲过的知识，其他的知识点我在前面的文章中都有向你详细介绍过。如果你不记得或者觉得生疏了，可以再复习一下前面的文章。

总之，通过本文的“串联”，你就可以实现一个真正的 1 对 1 的实时音视频直播系统了。赶紧操练起来吧！

思考时间

你是否可以将以前所学的所有功能都集成到一起，形成一套相对完整的直播系统呢？

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

[所做 Demo 的 GitHub 链接（有需要可以点这里）](#)

从 0 打造音视频直播系统

手把手教你打造实时互动音视频直播系统

李超

新东方音视频直播技术专家
前沪江音视频架构师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 22 | 如何保证数据传输的安全（下）？

下一篇 24 | 多人音视频实时通讯是怎样的架构？

精选留言 (4)

写留言



初音韶歌

2019-09-08

老师，请问怎么停止获取桌面与音频呢？我不需要将桌面与音频向外发送的时候总不能继续获取吧？



Joseph

2019-09-05

老师，现在我通过trickle-ice网站测试可以relay回地址，但通过通过浏览器访问IP：3478端口，显示访问不到网页，但是p2p通信是可以的。请问是什么问题呢？

作者回复：turn服务不是http/https服务,你用浏览器访问它是啥意思呢？



刘丹

2019-09-05

请问怎样查询房间号码或者名称列表？另外登录turn服务器时，客户端能否用自己的用户名和密码呢？

作者回复: 登录turn应该要用turn配置中设置的用户名和密码了。你要想查房间号，需要自己做一个小的管理系统。



rencwang

2019-09-05

老师，有个问题想请教一下，网页端支持播放avi格式视频，改怎么实现

作者回复: 系统默认应该是不支持的，你可以用ffmpeg将 avi 转成浏览器支持的格式，一般都是这样做

