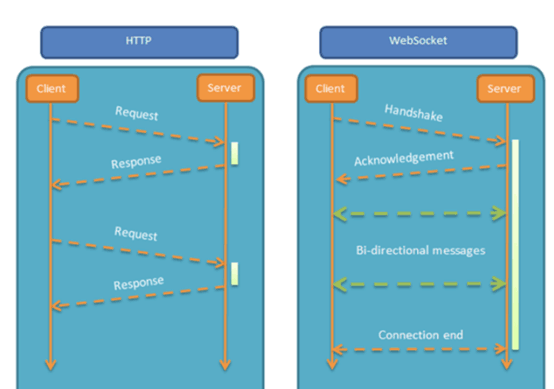
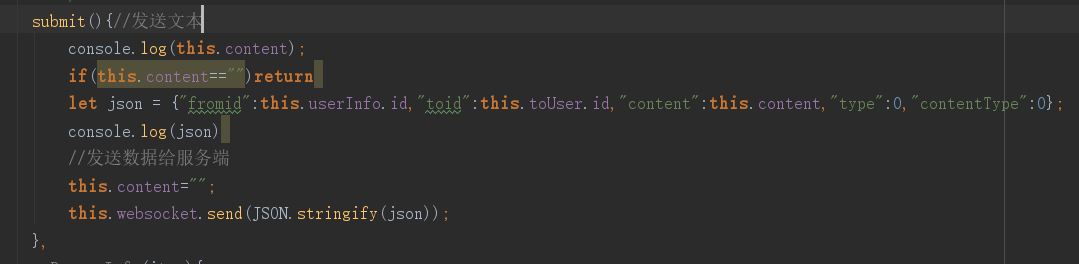
## 一．WebSocket



实现发送文本,图片,文件的业务逻辑：当发送信息的时候,我构造了如下的json数据给后端的websocket服务器：fromid是发送者的id,toid是接收者的id,contentType是内容类型,然后后端做了处理



文本

描述已自动生成

文本

描述已自动生成

解释一下下面字段的含义：  
**FIN**：标识是否为此消息的最后一个数据包，占 1 bit

**RSV1, RSV2, RSV3**: 用于扩展协议，一般为0，各占1bit

**Opcode**：数据包类型（frame type），占4bits

**MASK**：占1bits，用于标识PayloadData是否经过掩码处理。

**Payload length**：Payload data的长度。

1、如果其值在0-125，则是payload的真实长度。

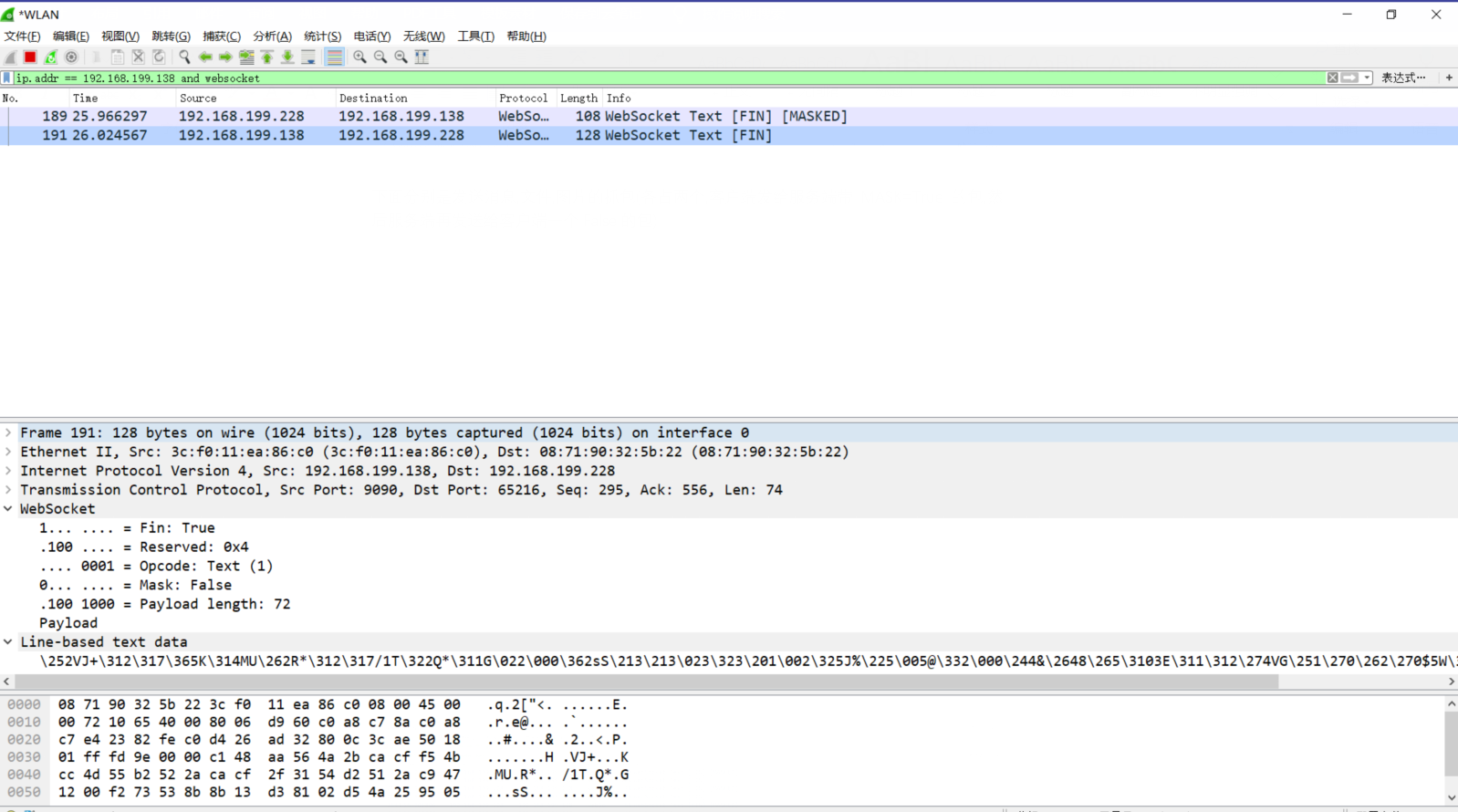
2、如果值是126，则后面2个字节形成的16bits无符号整型数的值是payload的真实长度。

3、如果值是127，则后面8个字节形成的64bits无符号整型数的值是payload的真实长度。

图形用户界面, 文本, 应用程序

描述已自动生成

下面分别是发送消息,文件,图片和用户断开连接的抓包(各占两个,客户端发给服务端带MASK=True的包,然后服务端再发送给客户端一个False的包)



文本

描述已自动生成

### 二.WebRTC

图片包含 文本

描述已自动生成

下面是进行网络信息交换的流程图

图示

描述已自动生成

下面是通过信令服务器建立P2P信道的流程图

图片包含 图示

描述已自动生成

下面是大致的实现流程图

图示

描述已自动生成

下面是websocket作用信令服务器处理信息的代码

文本

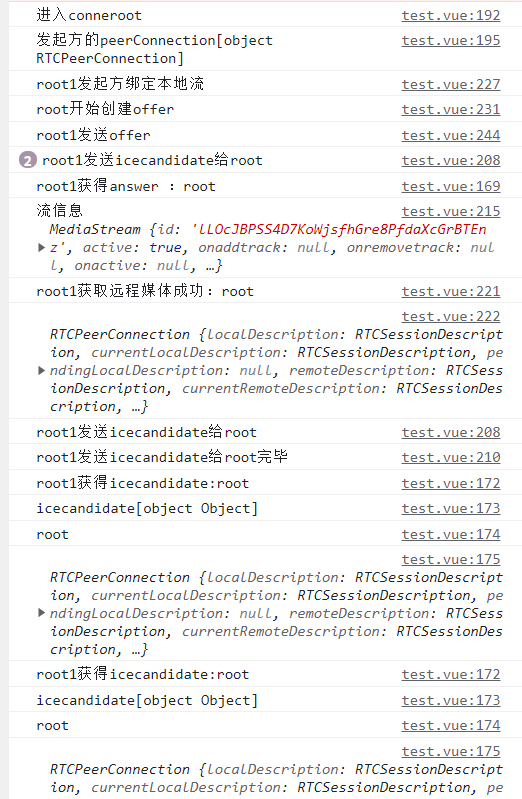
描述已自动生成

下面是客户端之间实现p2p连接的过程

1. 主动连接方创建和发送offer(也就是sessionDescription)
2. 调用RTCPeerConnection.setLocalDescription()方法将本地的localDescription设置为刚才创建的sessionDescription
3. answer方接收offer
4. answer方接收发送方的sessionDescription,并且用setRemoteDescription方法设置
5. 调用RTCPeerConnection.createAnswer方法产生自己的sessionDescription。
6. 接收方创建本地的sessionDescription
7. 发送sessionDescription给发送方,此时数据的type为answer
8. 各自向对方icecandidate并调用addIceCandidate来建立连接

9. 其中一方发送完全部的icecandidat并用addIceCandidate来建立连接

下面第一张图是发送方的,第一张图是发送方的控制台打印信息



图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

STUN协议：简单来说就是明确自己的外网地址的。它允许位于NAT（或多重NAT）后的客户 端找出自己的公网地址，查出自己位于哪种类型的NAT之后以及NAT为某一个本地端口所绑定的Internet端端口。这些信息被用来在两个同时处于 NAT路由器之后的主机之间建立UDP通信。

DTLS:(Datagram Transport Layer Security)即数据包传输层安全性协议。[TLS](https://baike.baidu.com/item/TLS/2979545)不能用来保证[UDP](https://baike.baidu.com/item/UDP/571511)上传输的数据的安全，因此Datagram TLS试图在现存的[TLS协议](https://baike.baidu.com/item/TLS%E5%8D%8F%E8%AE%AE/7129331)架构上提出扩展，使之支持UDP，即成为TLS的一个支持数据包传输的版本。DTLS 1.0 基于 TLS 1.1, DTLS 1.2 基于TLS 1.2。