# WebRTC项目实施建议方案

## 项目需求

1. 局域网内视频/语音通话，一对一和一对多（视频会议）的功能

2. 指定当前用户的通话，单个画面进行放大和缩小

3. 支持的终端 Web、 PC、Android 、IOS

## WebRTC通用解决方案

**对接第三方成熟平台（PaaS方案）**

|  |  |  |
| --- | --- | --- |
| **三方平台** | **项目** | **特点** |
| 腾讯SDK（TRTC） | https://cloud.tencent.com/document/product/647 | 1. 只需对接SDK，开发快，集成简单 2. WebRTC服务端由平台提供，节点多，性能稳定 3. 按需计费（终端数、转发流量） 4. 不能在局域网（与外网隔离）内使用 |
| 声网（YY团队） | https://www.agora.io/cn/ |
| 即构（QQ团队） | https://www.zego.im/html/document/ |
| 叁体（WebEx/Cisco） | https://www.cisco.com/c/zh\_cn/solutions/collaboration/webex-teams.html |
| 野狗 | [https://www.wilddog.com/product/webrtc](http://jump2.bdimg.com/safecheck/index?url=rN3wPs8te/pL4AOY0zAwh1oxD2Rtn4IxtZqHBtZOSovv/x/kEEf6FmC2NavBq2OyD8tklOvQIXCM/JyY4gNNnnZKUW8RozkK5n2r34E+m8gtYFuzxcrhobtgtHKWL7bN/spIHqleqpBnzZmkAijtk3Y9qHh6BM0y" \t "https://tieba.baidu.com/p/_blank) |

**采用完整的开源系统**

|  |  |  |
| --- | --- | --- |
| **方案** | **项目** | **特点** |
| Rocket.Chat | https://github.com/RocketChat/Rocket.Chat | 1. 系统成熟度较高，几乎涵盖了通讯方面所有的功能，以及许多附加功能文件传输等 2. 系统独立性很高，难以融入现有平台，只能作为独立服务进行部署 3. 开源免费，维护方便 4. 内、外网皆可部署 |

**基于WebRTC的自研系统**

WebRTC非常适合点对点（一对一）的音视频会话，实现多对多的视音频会话时，也可以采用点对点的连接方式（Mesh），构成网状结构。当参与用户增多时（大于4个），Mesh结构就不能适用了，一是终端（编解码）负荷增大，二是网络带宽需求增大，按照系统支持不同的用户数，采用不同的方案：

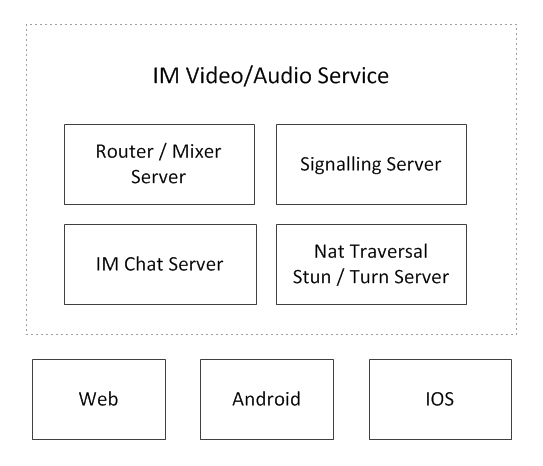
|  |  |  |
| --- | --- | --- |
| **方案** | **项目** | **特点** |
| Mesh网状结构（<=5） | WebRTC | 1. 基于WebRTC提供的开源代码即可实现 2. 对客户端处理能力及网络带宽要求高、无需服务端支持 3. 支持同时会话用户数受限 |
| SFU中继结构（<=5） | WebRTC + Router Server | 1. 中继只做数据包检测和转发，服务器性能要求不是太高 2. 客户端和Mesh结构一样，还是需要解码N - 1路视频流 3. 网络带宽相比Mesh可以节约许多，每个终端只需推送一路，接受N - 1路即可 |
| MCU传统结构 | WebRTC + Mixer Server | 1. 由服务器端将多路视频流进行合成，基础设施成本高 2. 由于混合需要解码和再编码，会产生延迟和质量的损失 3. 客户端最终收到的是一路视频流，没法做单个用户画面的放大和缩小 4. 支持用户数取决于Mixer服务器的处理能力 |

基于WebRTC做自研开发，可以很好的与当前业务进行融合，业务可控性强，内外网也都可以进行部署，缺点开发周期长，技术相对复杂。

## 项目实现分析

1. 之前开发过完整的IM系统，包括联系人关系、群组功能、消息发送、离线消息、点对点的视音频通信，终端支持Android、Web、IOS；
2. 多点通信，如果采用Mesh结构，开发难度比点对点的稍大，应该还是可以实现的，如果采用SFU或者MCU架构，需要对接第三方开源的 Router / Mixer Server，这部分还需要花时间去调研；
3. Windows桌面应用，目前没有采用c++实现过客户端，可以考虑通过嵌入CEF，通过Web的方式来实现。

## 基于WebRTC多端通信系统结构



**Signalling Server**

信令服务器，客户端之间的信令交互，一般都是采用WebSocket协议，自行研发。

**Nat Server**

穿透服务器，当遇到特殊网络，点对点穿透不了时，会使用TURN服务器来做中继，这类服务都遵循相应的标准RFC3489、RFC5766、RFC5389，一般都采用开源的实现。

**Router / Mixer Server**

中继或混合服务器，用来实现多人视频会议，这类服务器没有固定的标准，各家开源的实现各有特点，Router Server 的开源项目有jitsi、 Janus（可做为中继使用） ，Mixer Server的开源项目有licode、 kurento等。

**IM Chat Server**

聊天服务器，提供用户好友关系、分组、群组、在线用户等服务，用于进行辅助通信，一般都是自行研发。