WebRTC源码架构浅析

Google 在2010年花了6千8百万美元收购了大名鼎鼎的 Global IP Solutions (GIPS) 公司, 得到了它的 VoIP 专利和软件. 第二年, Google就把这些软件开源了, 不过, 不是作为独立的软件, 而且也和原来的软件功能大不一样, 而是作为所谓的 WebRTC 方案的一部分.

GIPS 主要是提供视频和语音引擎技术和开发包, 而 WebRTC 却要提供一揽子的多媒体聊天解决方案, 特别是嵌入到浏览器中, 使用 Web 相关技术(如 JavaScript, HTML5). 所以, WebRTC 的源码结构非常庞大, 单单是从 trunk 中获取的代码和数据就达到1.2G还多.

不过, 如果明白了 WebRTC 的架构, 就可以从这份开源代码中摘出我们想要的部分. WebRTC 包含下面几个部分:

1. 应用层, 即 WebRTC 技术. 此部分的技术主要是浏览器开发者需要, 但因为其是太过于顶层的应用技术, 过于偏向某一方面, 所以可取可不取.

2. 网络层, 主要是 libjingle. libjingle 就是 Google Talk 所使用的网络框架. 但 libjingle 并没开源服务器端的代码和技术, 再者, 网络层的可替换性很强, 各家公司总是设计自己的私有网络协议和架构, 所以, libjingle 的作用就是作为一个学习的对象.

3. 语音和视频引擎, 这部分才是原来 GIPS 公司的专利和核心技术! 所以, 要详细解读一番.

首先是语音技术, 可以先看这篇入门文章”浅谈网络语音技术“.

音频处理(audio processing)包含最主要的两个模块是: 回音消除(AEC, Acoustic Echo Cancellation), 静音检测(VAD, Voice Activity Detection). 没有前者, 用户只能头戴耳机聊天, 而且要非常小心, 否则会出现回声和啸音. 没有后者, 就要浪费大量的网络流量, 增加服务器混声负担. 其它的技术主要是为了提高单质.

音频编解码(audio coding)的目的是为了压缩, 减少流量. 虽然 GIPS 开发的 iLBC 和 iSAC 两种编解码器非常出名, 还有各种开放格式如 G.xxx, 但较多的资料显示, 开源的 Opus 方案似乎更胜一筹.

音频设备抽象层, 为不同的操作系统提供统一的录音和放音接口. 虽然有开源的 PortAudio 可以支持 Windows/Mac/Linux多个平台, 但似乎并不支持 iOS 和 Andoid. WebRTC 提供的源码支持了几乎所有常见平台.

还有一个技术难点, 是视频会议中的混音技术(mixer). WebRTC 的源码中提供想着的功能, 但似乎是非常简陋的一个实现. 混音技术一般要在服务器中进行, 但发给说话人的数据中, 该说话人的语音必须不在混音中. 因为用户不希望在自己的喇叭中再听到自己说过的话, 这效果和回声差不多.

其次是视频技术. 在编码方案的选择上, 有开源的 VP8 和专利的 H.264. 前者是 Google 收购和开源的技术, 而后者虽然有专利和收费, 但更成熟, 而且有资料表明H.264的技术稍微领先VP8. 最重要的是, H.264 有广泛的硬件设备支持, 例如 iPhone 上就集成 H.264 硬件编解码器(Apple 就是 H.264的专利拥有者之一). 在有硬件支持的设备上使用 H.264, 可以减少耗电量和 CPU 资源, 因为视频编解码需要的运算量非常大.