研究与开发

基于 WebRTC 技术的应用及平台技术开发与设计

林　鸿[1]，王　松[1]，杨　鑫[2]，付　斌[2]

（1.法国电信北京研发中心　北京 100190；2.中国电信股份有限公司北京研究院　北京 100035）

摘　要：WebRTC 具有开放标准、简单易扩展、开源免费等特点。除了通过调用 WebRTC 提供的标准 Web API 方便快速开发 Web 应用外，还可以利用 WebRTC 的核心实现库开发高质量的实时音视频通信移动客户端应用。总结了 WebRTC 的主要技术特点，对设计和开发基于 WebRTC 技术的移动客户端应用和 Web 应用进行技术分析，给出基于 WebRTC 技术的移动客户端应用、Web 应用和服务平台的参考设计和实现。

关键词：Web 实时通信；移动客户端；Web 应用；JavaScript 会话建立协议；STUN；TURN

doi: 10.3969/j.issn.1000-0801.2013.09.004

Research and Design of Mobile Client and Web Application Based on WebRTC Technology

Lin Hong[1], Wang Song[1], Yang Xin[2], Fu Bin[2]

（1. France Telecom R&D Beijing Centre, Beijing 100190, China;

2. Beijing Research Institute of China Telecom Co., Ltd., Beijing 100035, China）

Abstract: WebRTC technology has some good characteristics of open and standardized, simple and extendable, open sourced and free. The Web applications can be easily and quickly developed by calling the Web API which are provided by WebRTC technology and being standardized. Besides, the mobile clients and applications of real-time audiovisual communications with high quality can also be developed by fully exploiting the WebRTC core implementation library. The main technical features of WebRTC technology were summarized. Based on the summary, the technical analysis was given about how to design and develop the mobile client application and Web application based on WebRTC technology. The reference design and the implementation of the mobile clients applications, the Web application and the service platform which supports these two kinds of the applications were provided.

Key words: WebRTC, mobile client, Web application, JavaScript session establishment protocol, STUN, TURN

1　引言

WebRTC [1]（Web real-time communication，Web 实时通信）是一项在浏览器内部进行实时视频和音频数据通信的技术，是 HTML5[2]标准之一。Web 2.0 在过去的几年里将可编程性和交互性嵌入浏览器，而不只是显示静态内容和格式。但是，Web 技术还不能够应付实时双向的语音和视频通信需要。使用如 Adobe 的 flash 浏览器插件有明显的灵活性和性能等方面的限制。通过 WebRTC 技术可以开发具有实时音视频通信的 Web 应用，利用其核心实现还可以开发出具有实时音视频通信的移动应用。在这些基础应用上，结合其他的先进技术，可以开发出更多创新的 Web 和

移动应用。

2 WebRTC 主要技术特点

如图 1 所示[1],在 WebRTC 系统架构中,包括面向 Web 应用开发者的 Web API 和 WebRTC 核心库。WebRTC 核心库包含语音引擎 (voice engine)、视频引擎(video engine)、传输层、会话管理(session management)和 C++ API,它们都内嵌在浏览器里。

Web 应用开发者可以调用标准的 JavaScript API 开发基于 WebRTC 的应用, 主要包括:GetUserMedia API 用于控制本地设备的接入,如设备上的摄像头和麦克风,也叫 MediaStream API;PeerConnection API 用于管理在两个浏览器之间双向媒体流的发送和接收, 通过 JSEP(JavaScript session establishment protocol,JavaScript 会话建立协议)[3]实现媒体参数的协商, 这是 WebRTC 最典型的 P2P 应用场景;DataChannels API 用于浏览器之间发送和接收非多媒体的数据流。

在 WebRTC 核心库中,语音引擎实现了从声卡到网络整个音频媒体链的技术框架,包括对语音的声学处理以及 iSAC、iLBC 和 Opus 音频编解码的实现。视频引擎实现了从摄像头到网络、从网络到屏幕整个视频媒体链的技术框架,还包括视频图像的处理和 VP8 视频编解码的实现。会话控制是为支持呼叫建立和管理的抽象会话层,应用开发者决定如何实现具体协议。C++ API 是浏览器厂商实现 Web API 所需的函数集。

在传输层中,WebRTC 利用 ICE[4]/STUN[5]/TURN[6]机制来建立不同类型网络间的呼叫连接,解决 NAT 穿越问题。WebRTC 通过 RTP 传输音频和视频媒体流, 通过 SCTP 实现传输可靠数据流。90%左右 NAT 类型均可以利用 STUN 获得对方的公网 IP 地址和端口,实现穿越,不需要媒体服务器进行中继。只有 10%左右 NAT 为对称性 NAT,可以采用在互联网上部署 TURN 中继服务器的方式实现 NAT 穿越。可见,WebRTC 技术不同于传统的 SIP/IMS 网络部署,在不考虑监管等因素的情况下,大大降低了媒体服务器的负载和部署成本。

WebRTC 技术主要优点如下。

(1)开放的标准

WebRTC 是 HTML5 标准之一,也是由 W3C 和 IETF 标准组织共同定义的一个开放的标准。W3C 的 WebRTC 工作组[7]为 Web 开发者定义了基于浏览器的 Web API[8]用于开发具有语音、视频和数据功能的 Web 应用。而 IETF 的 RTCWeb 工作组[9]定义了浏览器之间媒体流交互的协议和规范, 但没有定义呼叫控制协议。3GPP 也正在制定 WebRTC 和 IMS 之间互联需求(TR 23.701[10])的标准。

(2)简单和易扩展性

WebRTC 提供了一个简单可扩展的技术框架和方案选型,方便开发者通过互联网提供语音、视频和数据等多种应用和服务。WebRTC 本身并不定义同用户之间的交互

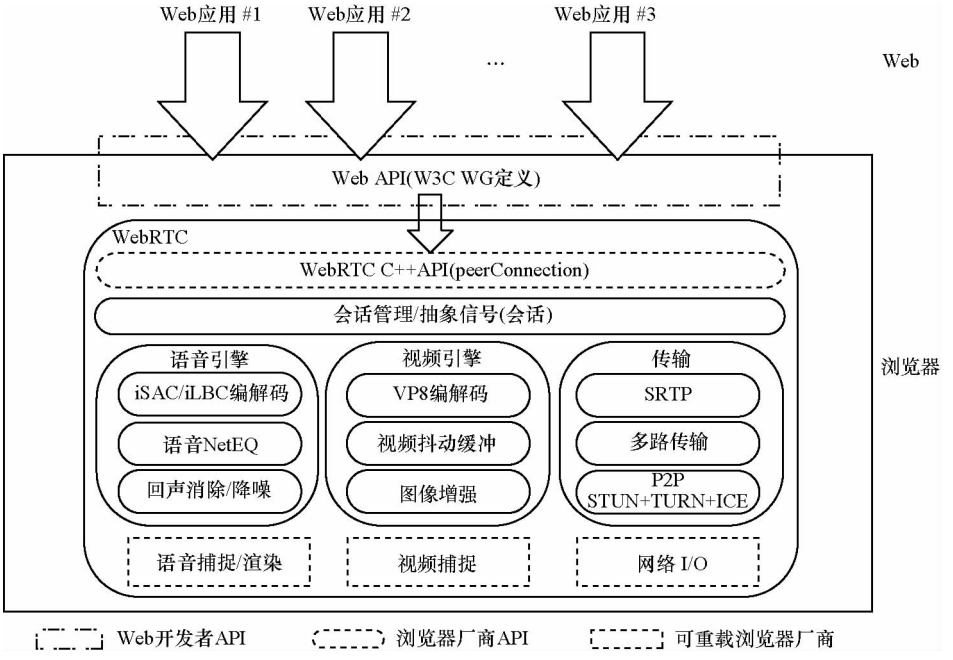

图 1 WebRTC 系统架构

方式、媒体流的路由方式、用户身份认证、呼叫协议和控制以及同其他网络的互联方式等。这些课题由开发者和服务提供商根据不同的业务场景和技术需要进行灵活选择和配置。

(3)来自产业链的广泛支持

WebRTC 技术获得来自产业链的支持。除了浏览器厂商,如 Google、Mozilla 和 Opera 外,其他大公司在 WebRTC 上也表现积极,如运营商 AT&T、Telefonica,设备商 Alcatel Lucent、Ericsson、Cisco、Avaya、Acme Packet 等以及来自从企业通信到游戏等领域的众多开发者和初创公司。许多围绕 WebRTC 技术的工作在美国、欧洲和亚洲,特别是在中国和韩国,都在积极进行之中。

(4)同运营商网络的互联互通

WebRTC 的一些应用场景可以作为运营商既有业务和网络的有效补充,比如通过 WebRTC 提供 IMS 服务、会议和呼叫中心的升级、企业统一通信业务的改进等以及一些垂直业务的扩充,如 M2M、教育和医疗等。

(5)和其他技术的结合

WebRTC 技术容易和其他的先进技术,如虚拟现实、手势控制、人脸识别等技术结合,实现 Web 应用的快速 mash-up 开发和实现。

3 基于 WebRTC 技术的移动客户端和 Web 应用的技术

基于 WebRTC 的 Web API 能够比较容易开发支持实时语音视频通信及数据可靠通信的 Web 应用以及基于这种基本能力的其他 Web 应用。但是这种 Web 应用还存在一些实际问题。比如,如何在浏览器中通知用户来电问题,一般情况下,浏览器处理呼入电话总是比出站连接更难。这种情况在手机上更复杂,来电通知可能需要常规的铃声,而不是简单的提示;如果用户已经处于一个 CS 或者 VoIP 通话,如何提示和区别这个 WebRTC 的来电是需要探讨的问题。另外,用户如果在呼叫中刷新了网页,这时 JavaScript 执行的上下文发生重置,会话中间状态会丢失。这些是 Web 应用实现实时通信需要解决的问题。

除了利用 Web API 开发 Web 应用外,利用 WebRTC 核心库也可以开发出具有实时通信功能的移动客户端。WebRTC 核心库具有如下特点。

(1)支持多种开源音视频编解码

音频编解码中支持 iSAC、iLBC、Opus 等免费音频编解码和 VP8 视频编解码。iSAC 是一种用于 VoIP 和流音频的宽带和超宽带音频编解码器。iLBC 是用于 VoIP 和流音频的窄带语音编解码器。其标准由 IETF RFC3951[11]和 RFC3952[12]定义。Opus 支持持续和可变码流编解码。其标准由 IETF RFC6716[13]定义。VP8 基于 WebM 项目[14],非常适合低时延的视频通信。

(2)具有强大的音视频引擎

WebRTC 语音引擎支持自适应抖动控制算法和语音分组丢失隐藏算法,用于缓解网络抖动和分组丢失引起的负面影响,使其能够快速且高解析度地适应不断变化的网络环境,确保音质优美且缓冲时延最小,提高语音通话质量;支持基于软件的信号处理来消除回声,实时地去除手麦克风采集到的回声;支持基于软件的信号处理来抑制噪声,用于消除与相关语音通话中某些类型的背景噪声。动态抖动缓存和错误隐藏算法,用于缓解网络抖动和分组丢失引起的负面影响。视频引擎支持视频动态抖动控制,缓解抖动和分组丢失引起的负面影响,有助于提升整个视频的通信质量;支持图像增强,去除从摄像头抓取图像的噪声。

(3)支持多移动平台

WebRTC 核心库可以移植到不同的移动平台,如 Android、iOS、Windows Phone 等。

(4)核心技术获业界认可

WebRTC 语音引擎如前所述来自 Global IP Solutions 公司[15]。许多大公司的产品,如 Skype、QQ、WebEX、AOL 等都曾采用该公司的 IP 语音引擎和相关产品。

所以,通过增加用户管理、会话协议控制和用户交互界面等工作,结合这个 WebRTC 核心库就可以开发跨平台的实时音视频通信的移动互联网应用。美国上市公司 Vonage,也开始考虑充分利用 WebRTC 的核心库开发 Android 和 iOS 移动平台的客户端应用,减少对私有和付费技术方案的依赖。

4 基于 WebRTC 技术的移动客户端和 Web 应用的参考设计和实现

使用统一的服务平台支持移动客户端和 Web 应用。服务平台提供一组标准的客户端 API,方便移动客户端和Web 应用调用,实现服务平台和前端应用的松耦合。目前大部分基于 WebRTC 的服务都部署在 GAE(Google App engine)上,用户之间 P2P 通道的建立以及呼叫信令的传

输,都是通过 GAE 完成。但是 GAE 环境国内不能访问。笔者基于开源软件项目部署自己的应用服务器,实现了基于 WebRTC 的服务。

WebRTC 没有定义用户呼叫管控制协议。SIP、XMPP[16]、其他标准协议或者完全私有协议都可以用于呼叫控制协议。SIP 是最常用的 VoIP,广泛用于企业的 IP PBX 和运营商的 IMS 平台。采用 XMPP/Jingle[17]用于呼叫控制,主要基于以下几点考虑:XMPP 本身比 SIP 更简单;客户端协议栈实现和 JavaScript 实现更轻量;XMPP 后台处理不需要安装类似 SBC 网关做 NAT 穿越;没有同 IMS 网络互通的需求等。

下面分别详细描述基于 WebRTC 技术的移动客户端、Web 应用和支持它们的服务平台的设计及具体实现。

4.1 移动客户端设计与实现

基于 WebRTC 开发实现了移动客户端微呼,这是一款支持新浪微博好友之间进行免费语音、文字通信的创新应用。通过调用服务平台提供的客户端 API 和服务平台交互,包括用户注册、登录、订阅用户状态、获取用户状态等。通过 XMPP 实现用户之间语音通信时的呼叫建立、信令协商、会话管理和 SDP[18]协商;通过 XMPP 实现用户之间的文字消息的传递;通过 RTP 传送语音媒体流,RTCP 监控媒体流传送质量;通过 STUN 和 TURN 协议来建立不同类型网络间的媒体链接,实现 P2P 的媒体流传输,减少中间媒体服务器的要求。

WebRTC 移动客户端基于 Android SDK 进行开发,集成了 libjingle 和 WebRTC 两个开源项目,其中 libjingle 库提供了 XMPP 的处理和解析等功能,WebRTC 库提供了语音编码和声学处理等功能。同时客户端也集成了友盟 SDK 的统计功能和新浪微博 SDK 的相关功能。软件架构如图 2 所示。

其中核心逻辑分层结构如下。

- package com.orange.weihu.activity:用户交互相关的核心逻辑模块。
- package com.orange.weihu.common:通用数据辅助模块。
- package com.orange.weihu.component:自定义相关视图组件模块。
- package com.orange.weihu.data:数据存储模块。
- package com.orange.weihu.jni:JNI (Java native interface) 调用接口层。
- package com.orange.weihu.net:网络处理模块。

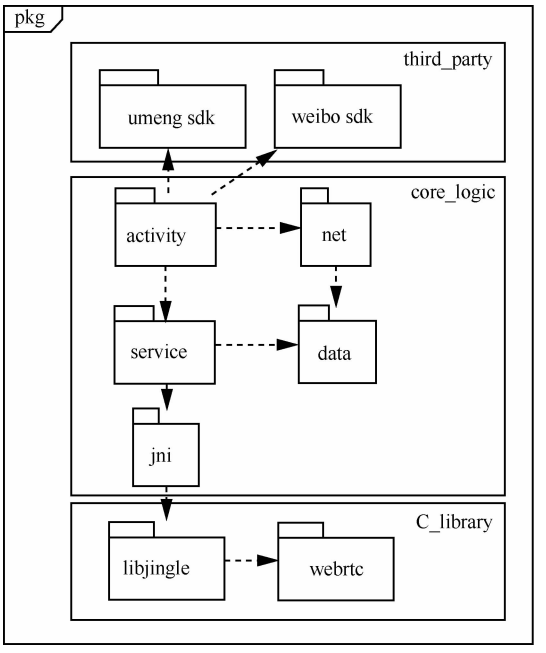

图 2 WebRTC 移动客户端的软件架构

- package com.orange.weihu.service:后台服务模块。
- package com.orange.weihu.util:通用辅助模块。

WebRTC 移动客户端模块间相互依赖关系如图 3 所示。其中,activity 包封装了用户交互相关的核心逻辑,包括微博认证登录管理和微博浏览展现,用户的微博好友关系维护和 WebRTC 好友关系的维护、文本消息的收发和记录管理、通话的发起和接收以及通话记录的管理。service 包封装底层通话相关库的核心逻辑,包括共享库的加载及生命周期管理、开机启动和微博信息获取等功能。data 包封装了需要持久化的核心逻辑,包括微博相关信息、好友关系相关信息、文本消息相关信息和通话记录相关信息。

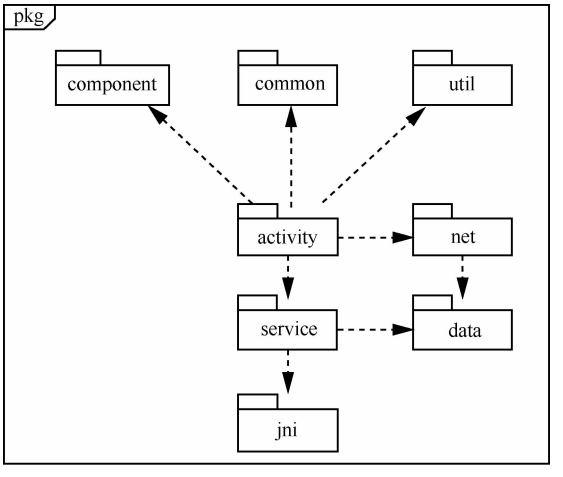

图 3 WebRTC 移动客户端模块间的相互依赖关系

WebRTC 客户端中各个模块调用的基本流程如图 4 所示。用户点击呼叫按钮,activity 向 service 传递消息,service 通过 JNI 调用 libjingle,libjingle 设置完成 WebRTC 相关功能,发起呼叫。用户通过微博账户登录,获得微博好友信息及微博详情,如果没有在微呼服务器注册过,则进行微呼账户建立。启动相关服务,加载底层类库,建立呼叫相关网络连接,注册在线状态。选择在线好友,访问后台服务器,获得好友连接信息,建立通话连接。

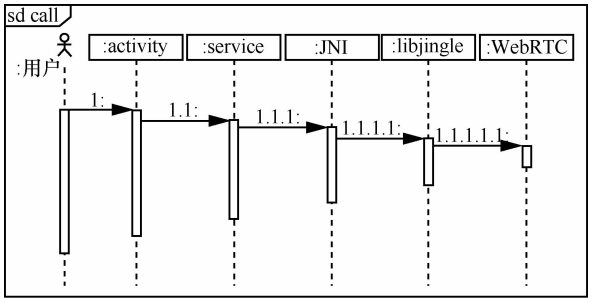

图 4　WebRTC 移动客户端的模块调用基本流程

4.2　Web 应用设计与实现

基于 WebRTC 的实时语音视频通信 Web 应用在无需任何插件的情况下,实现了浏览器之间 P2P 的音频和视频通信,提供了 Web 用户注册、登录、获取好友在线状态以及与在线好友进行实时语音和视频通话等功能。

Web 应用部署在应用服务器 Openfire[19]上,通过 JavaScript 的方式连接并访问 Openfire 服务器。Openfire 服务器提供了 HTTP 绑定的功能,通过 7070 监听端口,接收 Web 客户端发送的 HTTP 请求,登录并访问服务器。Web 应用基于开源 JavaScript 库 Strophe,是基于 JavaScript 的可扩展通信和表示协议(XMPP)客户端库,主要功能包括用户登录和登出、监听并发送 IM 消息、监听服务器推送的 presence 更新信息、监听服务器转发的 IQ 信息。

Jingle 协议是 XMPP 的扩展,用于规范两个 XMPP 客户端之间的信令协议。SDP 信令流程和 Jingle 信令流程的不同之处在于 P2P 通道地址和端口的发送时间,Jingle 信令在发起呼叫时,便会协商 P2P 通道地址和端口;而 SDP 信令将编解码协商和 P2P 通道地址协商分为两个信令发送,在发送发起呼叫信令后,便马上发送 P2P 通道地址和端口。WebRTC 中的信令是基于 SDP 格式的,而 Openfire 应用服务器是基于 XMPP 的,信令格式基于 Jingle 协议,所以需要对信令格式进行转换,包括 SDP 信令和 Jingle 信令的互换,需要将 JSEP 中的 SDP 格式映射为 Jingle 格式[20]。

4.3　服务平台设计与实现

服务平台基于 XMPP,并以 XML 数据元流式来传输数据,为移动客户端和 PC 客户端提供基于 Web 的 IM 传输、VoIP 呼叫、信令传输、用户管理和用户关系维护、好友状态 presence 信息推送以及日志统计等功能。

服务平台的逻辑架构如图 5 所示,服务平台包括 XMPP 服务器、STUN 服务器和 TURN 服务器。其中 XMPP 服务器用于即时消息转发和实时通信信令传输,STUN 服务器用于 NAT 穿越,TURN 服务器用于媒体流的转发。

Openfire 是基于 XMPP、跨平台的即时消息服务器,支持服务器到服务器的连接,用于部署多个 Openfire 服务器

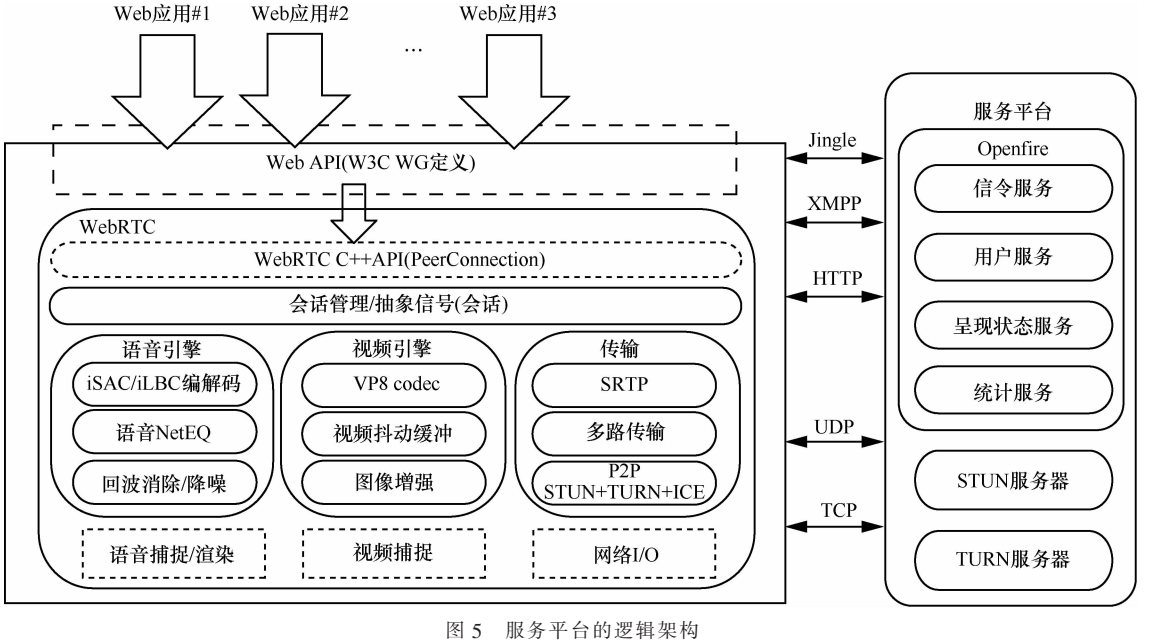

图 5　服务平台的逻辑架构

进行扩展;支持部署多个连接管理器,用于整合到 Openfire 的客户端连接,提高 Openfire 的在线用户数;提供了一系列的可用插件,并支持插件扩展,即开发者可通过部署新的 plugin,进行功能的扩展。Openfire 还作为 VoIP 通信信令服务器,提供端到端的 Jingle 信令的协商和传输。

服务本平台以 plugin 的方式,对 Openfire 进行了功能扩展,包括用户管理和注册、获取用户在线状态和通信日志统计等。服务平台采用不同的协议与客户端进行交互:IM 传输采用 XMPP,信令传输基于 Jingle 协议,对所提供 API 的访问采用 HTTP。

服务平台基于 STUN 和 TURN 的方式进行 NAT 穿越。STUN 服务器检测主机的 NAT 类型,并根据 NAT 类型,在客户端进行打洞,以支持两个客户端之间的 P2P 通信。TURN 服务器用于解决对称型 NAT 的穿越问题。TURN 服务器相当于中继服务器,客户端获取 TURN 的公网 IP 地址,客户端之间的通信均通过 TURN 服务器进行中转,这对于 TURN 服务器的性能提出了很高的要求。但基于 TURN 服务器是对 STUN 服务器的有效补充,可以满足性能要求。

Web 应用以 HTTP 绑定的方式登录并访问 Openfire 服务器,但 Web 客户端和 Openfire 服务器处于不同域,涉及浏览器 JavaScript 跨域问题。这就需要 Web 客户端调用 JavaScript 的 80 端口,并且是同一子域。使用 Apache 做反向代理,转发 HTTP-bind 请求到 XMPP 服务器的 HTTP-binding 端口。

基于安全性的考虑,服务平台将 Openfire 服务器部署在内网,并同时配置内网 IP 地址和外网 IP 地址,使内网用户和外网用户均可访问。STUN 服务器必须部署在公网,并且内置两个网卡,向外部提供两个公网 IP 地址和两个端口,进行 NAT 类型的检测和打洞。TURN 服务器也部署在公网,向外部提供一个公网 IP 地址和端口。Apache 反向代理服务器与 Openfire 服务器部署在同一台机器上。

5 结束语

本文总结了 WebRTC 的主要优点和技术特点,对分别利用 WebRTC 技术提供的核心库和 Web API 设计和开发具有实时音视频通信功能的移动客户端应用和 Web 应用进行了技术可行性分析,并详细描述了笔者基于 WebRTC 技术开发移动客户端应用、Web 应用和服务平台的参考设计和实现。

在不同的移动网络环境(3G 网络和 Wi-Fi 网络)和不同的 Android 智能手机上对所开发的 WebRTC 移动客户端进行测试,发现 WebRTC 核心库对网络时延、噪音抑制和回声消除都有较好的处理,能在不同网络环境中获得较好的语音质量。在固定宽带和 PC 上测试 WebRTC 的 Web 应用,除了较好的声音质量,VP8 也能很好地处理视频信息。当然,还需要在更多环境和条件下对 WebRTC 技术进行测试,对测试结果进行系统客观的分析,更好地评价 WebRTC 技术在音视频通信服务中的技术成熟度。

WebRTC 移动客户端和 Web 应用还有很大的改进空间。比如,客户端可以根据不同的网络环境动态调整编解码,在 iSAC、iLBC 和 Opus 编解码中自动选择适合当前网络环境的最优编解码和技术参数。视频编解码 VP8 和 H.264 在不同网络环境下的性能对比也值得进一步的深入研究。WebRTC 移动客户端目前只支持 Android 移动平台,可以将 WebRTC 核心库移植到其他移动平台上,客户端可以支持更多的操作系统。也可以将 WebRTC 核心库和移植部分封装为 SDK,供移动应用开发者开发各种具有实时通信能力的移动客户端。Web 应用同移动客户端之间的互联也值得进一步的研究和实现。在 WebRTC 技术中,目前还没有进一步挖掘的能力和对数据流的支持,包括 Web API 中的数据信道的使用和 WebRTC 核心库中对应但还未出现的数据流引擎。

参考文献

1 WebRTC. http://www.webrtc.org

2 W3C HTML5. http://www.w3.org/TR/html5/

3 IETF RTCWeb JSEP.JavaScript session establishment protocol. http://datatracker.ietf.org/doc/draft-ietf-rtcweb-jsep/,2013

4 IETF RFC5245 ICE.Interactive Connectivity Establishment,2010

5 IETF RFC5389 STUN.Session Traversal Utilities for NAT,2008

6 IETF RFC5766 TURN.Traversal Using Relays around NAT,2010

7 W3C WebRTC Workgroup. http://www.w3.org/2011/04/webrtc-charter.html

8 W3C WebRTC API. http://dev.w3.org/2011/webrtc/editor/webrtc.html

9 IETF RTCWeb Workgroup. http://tools.ietf.org/wg/rtcweb/charters

10 3GPP TR 23.701.Study on the Support of WebRTC IMS Client Access to IMS,2013

11 IETF RFC3951 iLBC.Internet Low Bit Rate Codec,2004

12 IETF RFC3592 RTP iLBC. RTP Payload Format for Internet Low Bit Rate Codec Speech, 2004

13 IETF RFC6716 Opus.Definition of the Opus Audio Codec,2012

14 WebM project. http://www.webmproject.org/

(下转第 36 页)