基于 WebRTC 的实时视音频通信研究综述

张向辉[1] 黄佳庆[1] 吴康恒[2] 雷志斌[2]
(华中科技大学电子与信息工程系智能互联网技术湖北省重点实验室 武汉 430074)[1]
(香港应用科技研究院 香港新界)[2]

摘 要 实时网页通信(WebRTC)无需在网页上安装插件或客户端即可实现实时视音频通信,具有跨平台、开发成本低和适用范围广等优点,已被 W3C 纳为草案,正在带来多媒体通信的一场革命。首先介绍基于 WebRTC 实时视音频通信的总体架构;接着详细阐述其相关关键技术,包括信令机制、网页应用和浏览器底层 WebRTC,后者还包括视、音频编译码模块和传输模块;然后从两用户和多用户的角度对 WebRTC 实时视音频通信应用的实现细节进行比较;最后讨论值得进一步研究的开放问题。

关键词 WebRTC,P2P,WebSocket,视音频通信,视频会议
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.001

Survey on Real-time Video and Audio Communication Based on WebRTC

ZHANG Xiang-hui[1] HUANG Jia-qing[1] WU Kang-heng[2] LEI Zhi-bin[2]
(Hubei Provincial key Laboratory of Smart Internet Technology, Department of Electronics & Information Engineering, Huazhong University of Science & Technology, Wuhan 430074, China)[1]
(Hong Kong Applied Science and Technology Research Institute (ASTRI), HongKong_New Terroteris, China)[2]

Abstract WebRTC (Web real-time communication) can achieve real-time video and audio communication on the Web without installation of any plugin or client software. It is accepted as W3C draft, because of low development cost and wide application range. WebRTC is bringing us an innovation era on multimedia communications. The overall framework of WebRTC-based real-time video and audio communication was introduced first, and then the related key technologies of WebRTC were elaborated, including signaling mechanism, Web app and WebRTC of low layer of browser. The latter also includes video codec module, audio codec module and transmission module. Next, implementation details of WebRTC real-time video and audio communication applications were compared from the point of view of two users and multiple users. Finally, open issues that are worthy of further study were discussed.

Keywords WebRTC, P2P, WebSocket, Video and audio communication, Video conference

1 引言

网页实时通信 WebRTC[1](Web Real-Time Communication)可使支持 HTML5 的浏览器通过调用 JavaScript(JS) API 开发功能丰富且高质量的实时通信应用,尤其是可使网页具备视频对话或音频对话等实时通信功能,其于 2011 年 6 月开源,并在 Google、Mozilla 基金会、Opera 支持下被包括进万维网联盟的 W3C 推荐标准,W3C 中定义了 PeerConnection、DataChannel[2] 和 MediaStream[3] 等 API 的细节。WebRTC 既不需要插件,也不需要下载或安装客户端,能给网页开发者提供功能多样、实时多媒体应用的接口,从而在不同的浏览器、多样性的平台之间[4,5]搭建一个健壮、互通的实时通信平台。

本文详述 WebRTC 视音频通信总体架构和关键技术,阐明 WebRTC 的基本概念;然后从用户数量的角度,总结现有的网页应用的优缺点和详细原理,并指出目前已有的支持多用户视音频通信的 WebRTC 网页应用采用非 P2P 组网方式,可扩展性受到限制。这里的 P2P[6,7],不是指点到点的通信,而是与不同于客户端-服务器模型相对的、无中心服务器、依靠许多对等用户(Peers)交换信息的互联网体系,即对等网络。

2 WebRTC 视音频通信架构和关键技术

基于 WebRTC 的视音频通信总体架构如图 1 所示,主要包括 3 个大类:信令机制、浏览器底层的 WebRTC、网页应用(Web App)。信令机制用于协调浏览器之间建立连接的过

到稿日期:2014-01-17 返修日期:2014-06-10 本文受国家自然科学基金(61271227)资助。
张向辉(1990-),女,硕士生,主要研究方向为 WebRTC;黄佳庆(1972-),男,副教授,主要研究方向为网络通信、网络编码,E-mail:jqhuang@hust.edu.cn(通信作者);吴康恒(1978-),男,博士,主要研究方向为网络媒体与安全;雷志斌(1968-),男,博士,主要研究方向为网络媒体与安全。

程；浏览器底层的 WebRTC 包括视频编解码模块、音频编解码模块、传输模块；网页应用是调用 WebRTC 的 JS API 实现的网站。

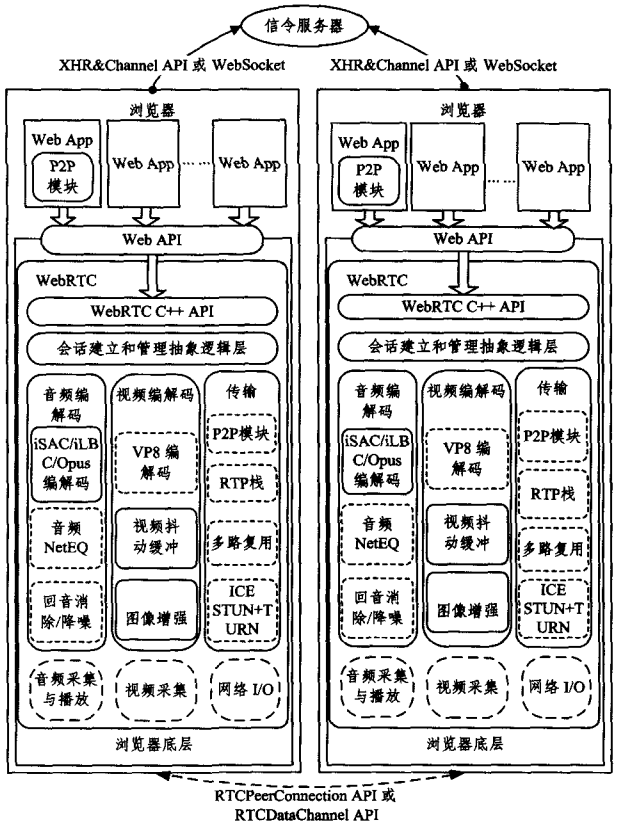

图 1　基于 WebRTC 的视音频通信总体架构图

2.1　信令机制

WebRTC 信令的作用是：在客户端之间传递控制信息，通过控制信息处理客户端之间的发现、连接建立、连接维护和连接关闭等任务。启动一个 WebRTC 会话需要以下几个信息：本地客户端的初始化信息；远程客户端的初始化信息；远程参与建立连接的信息，主要是 IP 和端口。

JSEP(JavaScript Session Establishment Protocol)[8]，即是 JavaScript 会话建立协议。它是这样的信令的处理过程：呼叫方发送请求(Offer)；被呼叫方接受该请求；被呼叫方反馈应答(Answer)；呼叫方接受应答。WebRTC 中定义了一系列的 JS API，从而简化了 JSEP 实现过程，浏览器仅需要按照规范调用这些 API，即可完成交换请求和应答消息，从而创建会话。

WebRTC 未指定信令实现的方法和协议标准，但所有的 WebRTC 应用都需要一个机制来交换两个通信终端的初始化媒体信息，即都需要进行 JSEP 信令过程来交换 SDP(Session Description Protocol)。会话描述协议 SDP 描述的是流媒体的初始化参数[9]。WebRTC 应用的开发者就可以自主选择适合的协议，主要有如下两种机制。

2.1.1　XHR&Channel API 信令机制

XHR(XMLHttpRequest)和 Channel API 共同可作为一种信令机制，使用这种机制时信令服务器是托管在 GAE (Google App Engine)上的。XHR 是一种 HTTP 请求协议，是提供给浏览器脚本语言的 API，它用于把 HTTP 或 HTTPS 请求发送到服务器，加载服务器响应数据，并将其返回到浏览器脚本。GAE 是一个开发、托管网络应用程序的平台，也是一个云端的服务器，Channel API 是 GAE 中为实时通信服务器推送提供的接口[10]。

该信令机制原理：客户端通过 XML 向服务器发送一个请求，服务器拿到这个请求后，即为每一个客户端创造一个客户 ID，通过这个 ID 关联一个永久的通道(Channel)，同时客户端也会创建一个套接字(Socket)来监听这个通道。一旦有客户端加入退出或客户端的信息有变化，此客户端就会发出 POST 请求到服务器，服务器即刻通过通道发送信息给需要的客户端。这个通道是单向的，即是从服务器到客户端永久存在的一个渠道，之所以要一直存在，是为了确保客户端信息更新的即时性，从而达到即时通信的效果。Channel API 实质上是 Iframe 流[13]的 Comet 方式，Comet 在 2.1.2 节中会详细介绍。

2.1.2　WebSocket 信令机制

WebSocket 是基于 HTML5 的一种浏览器与服务器间进行全双工通讯的网络技术，属于 HTML5 的一项新的特性，其出现使得浏览器对套接字的支持成为可能，从而在浏览器和服务器之间提供了一个基于 TCP 连接的双向通道。WebSocket 通信协议于 2011 年被 IETF 定为标准[11]，WebSocket API 被 W3C 定为标准草案[12]。

WebSocket 技术为通过因特网进行的双向通信提供了一个新的 JS API 和协议。WebSocket 协议绕过了速度较慢的基于文档的 HTTP 协议，便于直接处理固定数据格式。在 WebSocket API 中，浏览器和服务器只需执行一个握手(Handshaking)操作，它们之间就会形成一条可以互相传送数据的快速全双工通道。WebSocket 的头信息很小，只有 2 字节[11]，可以节约带宽资源，降低主机压力。

下面从延迟性和吞吐量两方面比较实时网页应用 3 种方案：轮询(Polling)、Comet 和 WebSocket，如图 2 所示，以进一步说明 WebSocket 的优越性。

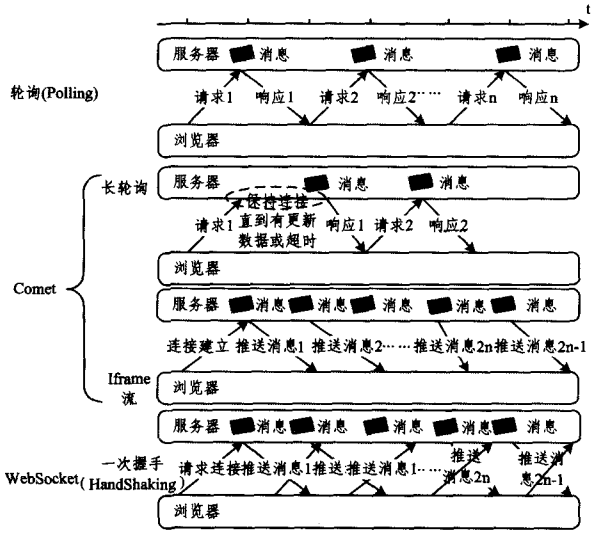

图 2　轮询、Comet、WebSocekt 延迟性比较

(1)延迟性

轮询(Polling)是由浏览器定时向服务器发送请求，服务器接到请求后马上返回最新的数据给作为客户端的浏览器，并关闭连接。这种传统的模式很明显的缺点是浏览器需要不断地向服务器发出请求，请求周期长，实时性不好。

Comet是一种用于网页的推送技术[13]，能使服务器实时地将更新的信息传送到客户端，而无需客户端发出请求，目前有两种实现方式：长轮询和 Iframe 流[13]。第一种是基于 AJAX（Asynchronous JavaScript and XML）[14] 的长轮询（Long-polling）方式：客户端向服务器发送 AJAX 请求，连接上服务器后，客户端与服务器一直保持连接，直到有更新数据或超时，立刻向客户端返回，连接断开，客户端处理完后，立刻重新发送 AJAX 请求。第二种是基于 Iframe 流方式，也叫 Comet 的长连接方式：客户端向服务器发送 AJAX 请求，服务器端接到这个请求后作出回应并不断更新连接状态以保证客户端和服务器端的连接不过期，客户端与服务器一直保持连接。当服务器有更新消息时立刻发送到客户端，除非在通信出现错误时，连接不会断开，这种方式和 WebSocket 的实现方式十分类似，Comet 虽然可达到双向通信，但依然需要发出请求。

WebSocket 可取代轮询和 Comet 技术，使客户端浏览器具备像 C/S 架构下桌面系统的实时通信能力。浏览器通过 JS 向服务器发出建立 WebSocket 连接的请求，即一次握手连接建立以后，客户端和服务器端即可直接交换数据。

图 2 中，通过对比以上 3 种技术可直观了解 WebSocket 能较大地降低延迟性。轮询的方式是一种半双工传输，当一个响应结束后，因连接关闭需要重新等待定时再次建立连接，所以会造成较大延时。Comet 的长轮询方式和传统轮询效率相似，不同于服务器接到请求后立刻返回，而是保持连接直到有数据更新再返回。而 Comet 长连接和 WebSocket 中，一旦连接建立，该双向通道会持续保持连接，不需要等待浏览器的定时请求，效率提高一倍。但 Comet 长连接方式会占用较多信道资源，由下一节可知其在吞吐量上不占优势。

（2）吞吐量

轮询方式下浏览器需要不断地向服务器发出请求，这样会占用较多带宽和服务器资源。基于 Iframe 流的 Comet 技术一直占用信道，导致过量消耗服务器带宽和资源，实质是轮询的一个改进。HTTP 请求和响应头信息总开销包含几百个字节，且还不包括任何数据。有时头信息非常长，里面包含的数据可能只是一个很小的值。相比之下，WebSocket 服务器与客户端之间交换的标头信息很小，约 2 字节，即使长连接，也能节省服务器资源和带宽并达到实时通信。由此可见，WebSocket 相比 Comet 长连接在吞吐量上占优势。

目前看到的 WebRTC 实现中，WebSocket 是应用最为广泛的信令机制，作为目前使用较多的服务器端 JS 平台，Node.js 可以提供该协议的一种实现。虽然 Node.js 本身没有原生的 WebSocket 支持，但有多种第三方的 npm 包实现，其中较为成熟的是 ws[15] 和 socket.io[16] 两个模块。目前主要的浏览器如 Chrome、FireFox、Opera、Safari 均已支持 WebSocket，而微软 IE10 及以上版本也支持 WebSocket。

2.2 浏览器底层的 WebRTC

2.2.1 音频编解码

音频编解码模块的目标是实现从声卡采集到的声音到可供网络传输的数据的框架，是在 Google 收购 GIPS（Global IP Solutions）公司后开源的。GIPS 是一家位于瑞典斯德哥尔摩的专门从事处理 IP 语音技术的高科技公司，掌握着 VoIP 技术的多项专利，广泛应用于北电网络、Skype、QQ、WebEx、AOL、EarthLink 等产品中[1]。源端声卡上数据采集后，经过编码再到网络上传输，接收端进行逆向过程。

它支持 iSAC、iLBC、Opus 3 种编解码器。iSAC（Internet Speech Audio Codec）[17] 是一种用于 VoIP 和流媒体的宽带和超宽带音频编解码器，iSAC 采用 16 kHz 或 32 kHz 的采样频率和 12 到 52kbps 的可变比特率。iLBC（Internet Low Bitrate Codec）[18] 是一种用于 VoIP 和流媒体的窄带音频编解码器[19]。采用 8 kHz 的采样频率，帧大小为 20 毫秒时比特率为 15.2kbps，帧大小为 30 毫秒时比特率为 13.33kbps。Opus[20] 是由互联网工程任务组（IETF）开发的有损音频压缩格式，特别适用于在因特网上的交互式实时应用，支持固定比特率和 6kbps 到 510kbps 可变比特率，帧大小从 2.5 毫秒到 60 毫秒，采样率从 8kHz（4kHz 带宽）到 48kHz（20kHz 全频，可再现人类听觉系统的整个范围）。音频 NetEQ 包括抖动缓冲及丢包补偿模块以提高音质，并把延迟减至最小，从而提高音频质量。回声消除/降噪是基于信号处理的一个功能，在实时采集过程中，存在由麦克录制时导致的回声和由一些背景等声音导致的噪音，该模块可消除回声，降低噪音。

2.2.2 视频编解码

视频编解码模块是一个实现从摄像头数据采集到网络数据传输，以及从网络传输的数据到屏幕显示的解决框架。VP8[21] 是 Google 开发的视频编解码技术。视频抖动缓冲技术用于降低视频抖动和丢包对视频质量带来的影响。图像增强技术，针对每一帧的图像进行处理，这其中包括用降噪、颜色增强等技术手段来保证更好的视频质量以达到视频图像处理的目的。

2.2.3 传输模块

传输模块主要包括 NAT 机制和传输机制。

（1）NAT 机制

WebRTC 对内网上的主机建立连接需要 NAT（Network Address Translation）[22-24] 即网络地址转换。WebRTC 直接采用了 Libjingle 中关于传输部分的组件。Libjingle 是 Google 公司开发的实现 P2P 传输的 C++ 开源库，Google Talk 即是基于这个库开发的。通过 Libjingle 可以建立一个直通的网络连接，可以无需关心中间的 NAT、防火墙、中继服务器和代理、会话建立的细节，直接进行数据的交换。

Libjingle 中采用的是 ICE（Interactive Connectivity Establishment）[22] 这种综合性的 NAT 穿越的框架。如图 3 所示，ICE 综合运用基于 UDP 的 STUN（Session Traversal Utilities for NAT）[24] 和基于 TCP 的 TURN（Traversal Using Relays around NAT）[23] 协议来实现。使用 ICE 的应用程序需有一个 ICE 代理（ICEAgent）来负责 ICE、STUN、TURN 以及其它模块的交互，发现本地设备的网络的拓扑结构的信息，找到一条路径或者通过该路径通信。每个代理都有一些可以用于与远端主机代理通信的候选（candidates）传输地址。在网络中，每个代理可有自己的 STUN 服务器[24]，也可共用一个。本地计算机收集到所有的候选传输地址，将它们按优先级高低进行排序，然后通过信令通道发送给远端计算机。这些候选传输地址作为 SDP 请求的属性被传输。当远端计算机收到请求后进行相同的收集过程，并且把自己的候选传输

地址作为响应消息发给请求者。这样，每个代理都有一个完整的包含了双方候选传输地址的列表，即可进行后续数据传输。

WebRTC 中这一过程由浏览器底层实现，每个 PeerConnection API 生成一个 ICE 代理[25]，当一个新的 ICE 候选连接可用时，ICE 代理将通过一个回调函数通知应用程序。当所有的候选地址搜集齐之后，回调函数也会通过触发来通知应用程序已经完成搜集工作。

STUN 需要一个公网 IP 的 STUN 服务器，使防火墙后的客户端找出自己的公网地址，这可以完成 92%的穿越。大约 8%的概率无法通过上述方式成功建立连接时就需采用中继服务器（见图 3 中 S3），即采用 TURN[26] 协议。

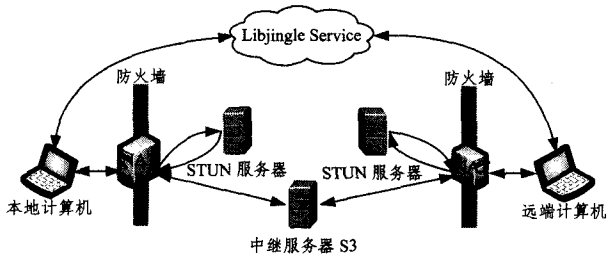

图 3　ICE—Libjingle 中的 NAT 策略

(2)传输机制

WebRTC 的传输模块重用了 Libjingle 的部分组件，WebRTC 技术在浏览器内部集成了一系列的网络传输及会话控制相关协议。传输及会话控制的实现重用了 Libjingle 项目的部分代码。RTP 栈是 RTP（Realtime Transport Protocal）实时传输协议[27]，该协议为实时媒体应用提供了一个传输音频和视频数据的方案。

以上，音频编解码、视频编解码和传输模块包含了视频音频处理技术，即 NAT 技术，但未包含 P2P[6] 技术。具体的实现细节随着 WebRTC 项目的建立现在都已经开源了，开发者可将其移植到桌面客户端和移动客户端等软件开发中，也可以在移植到视音频通信系统等硬件开发中。而对网页开发者来说，这些 WebRTC 的内部处理是高度封装的，网页开发者通过调用 WebRTC 的 API，借助浏览器来实现媒体通信，即不用关心它底层是怎样实现的。

2.3　网页应用(Web App)

网页应用程序，即调用 WebRTC 的 API 实现的网站，属于 WebRTC 的顶层开发。如图 1 所示，开发者应用 WebRTC 提供的 API，可以实现网页上的视音频通信，另外有些网页应用嵌入了 P2P 功能，实现了基于 WebRTC 的 P2P[6] 下载，这一节着重介绍网页应用程序中的 P2P 模块，网页视音频通信的例子将会在第 3 节详细展开。

2.3.1　网页应用中的 P2P 模块

WebRTC 网页应用中的 P2P 主要包括伙伴关系管理和数据调度。

(1)伙伴关系管理

P2P 网络是在原有物理网络的基础上形成的一种逻辑结构的网络，用来表示节点之间的逻辑关系。最简单的节点管理策略是集中目录式，每个节点加入到 P2P 系统中时，都需要向目录服务器注册，中央服务器保存了所有节点上传的文件索引和存放位置的信息，最后节点连接到文件所有者的节点主机上，与此主机建立连接并传输文件。节点之间会周期性地交换缓存映像（BufferMap）[28]。缓存映像可通过 bool 类型的数组来实现，如果其中某位的值为 1，则表明该位对应的分段有效，即该分段已经存储在缓存中；反之，如果为 0，则表明改位对应的分段无效。定期交换缓存映像可由 DataChannel API 来实现[29]。节点的加入、节点的离开、节点的伙伴关系管理等都是 P2P 系统中所必须解决的节点管理问题。

(2)数据调度

P2P 的数据调度包括父节点选择和数据块选择两方面。每个节点独立地选择它的伙伴，然后通过消息传播机制，告诉它的伙伴它所拥有的数据，每一个节点从它的邻居节点请求缺失的数据块，同时也传递可用的数据块到它的邻居节点。文件被分割为大小相等的数据块并编号，节点在文件下载初期，即节点为新加入网络的节点时节点自身没有任何文件分片，会随机产生一个需要下载的文件片的 ID 号，然后在邻居节点中选择具有该文件分片 ID 号的一个节点，并向该节点发送请求文件分片。总之数据调度要解决的问题是确定从哪个伙伴处请求哪个数据块。

2.3.2　P2P 的 WebRTC 网页应用实例

采用 P2P 技术的网页应用可提升吞吐量，增强可扩展性。目前支持 P2P[6] 下载的网页应用为 PeerCDN、SwarmCDN 和 ShareFest，但暂无 P2P 流媒体直播点播等 WebRTC 实时视音频通信。

(1)PeerCDN

CDN(Content Deliverry Network)就是通过在现有的网络中增加一层新的网络架构，将网站的内容发布到最接近用户的网络“边缘”，使用户可就近取得所需内容，缓解网络拥堵[30]。采用了 P2P 的 CDN 内容分发减少了服务器的压力，直观上提高了访问者的页面响应速度，因其利用了其他节点的闲置上行带宽。在视频通信中，带宽好的节点多承担分发的任务，从而均衡网络负载。

PeerCDN 是基于 P2P 的 CDN，能够借助于一些 JS 代码在浏览器之间分发网站数据，其中包含 WebRTC API。目前 PeerCDN 仅支持静态文件下载。静态文件不同于流媒体和实时视频。静态文件保存在服务器上，更新的频率较低。流媒体是供应商用一个视频传送服务器把节目作为数据包来传输。两者都需要带宽和处理能力强大的服务器来发送数据。

PeerCDN 现在已被 Yahoo 收购，不再提供试用，下一步很可能作为商用。

(2)SwarmCDN

SwarmCDN 与 PeerCDN 极为相似，这个架构是为了解决如下问题：通常浏览器在访问一个带有大量图片或者视频页面时，直接从服务器加载这个页面和视频是比较漫长的，这对于带宽不充足的用户的网络需求就带来了一定的困难。该架构是采用 P2P 技术来解决这个问题。

这是一种分布式大数据传输系统。这种系统的主要目的是从离用户最近的资源里分别加载大数据，以实现这种快速浏览的用户体验，弥补带宽不足这一缺陷。假设当前要打开

一个页面，那么就会去寻找附近已经打开或者正在打开这个页面资源的用户，分别从中获取需要的页面资源，最后结合到一起，从而提高页面加载的速度。

(3)ShareFest

sharefest.me 是一个基于 WebRTC DataChannel API 的文件传输网站。该网站在 github 上有开源代码，可以在本地部署。它是网页版的 BitTorrent[31]，共享文件的过程更加简单。如图 4 所示，浏览器 A 选择共享本地的一个文件，则会生成一个随机的会话房间即随机 URL，此时 A 充当临时的“文件服务器”，其它浏览器请求这个 URL 就可以从 A 下载该文件。当请求者数量增多时，下载过这个文件的主机相应页面只要没有关闭，就可以作为文件源供他人下载，从而大大提高文件共享的效率。

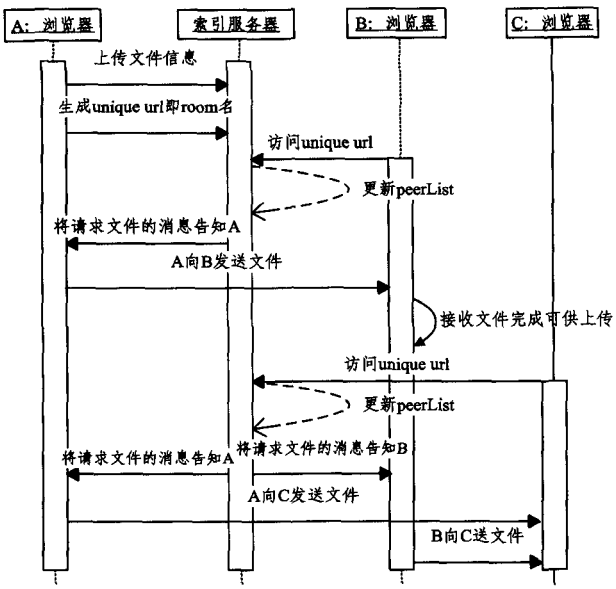

图 4　ShareFest 文件共享流程

3　基于 WebRTC 的实时视音频通信

基于 WebRTC 的实时视音频通信最初的实现仅支持两用户(详见 3.1 节)。随后可以支持 3 人及以上用户的视音频通信(详见 3.2 节)，网络拓扑结构采用全连接(Full Mesh)的方式。

3.1　两用户 WebRTC 实时视音频通信

Google 提供的是两用户的视频通信的示例，分别为 AppRTC 和 CodeLab。

3.1.1　AppRTC

AppRTC 是写入到 WebRTC 源码中的示例，只支持一对一的视频通话，是用原生 WebRTC API 所写的，且最早在跨浏览器——Google chrome 和 Firefox 之间实现视频会话。它采用 XHR 和 Channel API 作为信令机制。其开源代码包含在 WebRTC 开源工程中，其中 apprtc.py 是服务器文件，托管到 GAE 上，adapter.js 则屏蔽了不同浏览器之间 API 的差异。

3.1.2　CodeLab

CodeLab 是 Google 于 2013 年 4 月创立的一个逐步引导学习 WebRTC API 的例子，并不是一个成熟的项目。通过这个例子可以逐步学习如何通过 WebRTC 提供的 API，获取摄像头麦克风视音频数据，通过浏览器建立点到点的连接进行视频通话，以及传递其他任意非视音频的数据，建立基于 Node.js 的信令服务器，从而实现一个完整的视频通话。CodeLab 只支持两个用户，当第三个用户进入时，页面会提示会话已满。

3.2　多用户 WebRTC 实时视音频通信

根据 Google 提供的 WebRTC API，有开发者实现了 3 人及以上的视音频通信。目前多用户视音频通信采用全连接方式[32]。全连接方式的优点是组网实现简单，缺点是代价较高，如图 5 所示，n 个节点需要建立 $n(n-1)/2$ 个连接，初始化时每个节点与其他节点通过信令服务器建立连接。

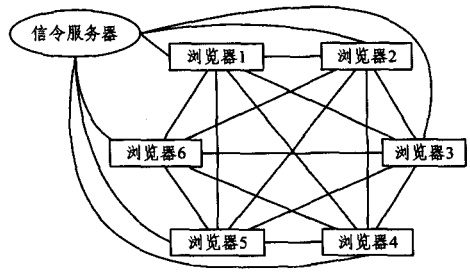

图 5　基于全连接的 WebRTC 多用户视音频通信

3.2.1　SimpleWebRTC

SimpleWebRTC 是最早的开源 WebRTC 多用户视频会议，从起初的 conversat.io 到现在相当成熟的 talky.io，其包含群组视频、屏幕共享、群组加密。

基于 SimpleWebRTC.js 库，主要用 PeerConnection API 来建立会话，从而在浏览器之间交换视音频数据，根本目的是建立一个点到点的连接，SimpleWebRTC.js 库屏蔽了这个过程中 WebRTC API 调用的细节。这个示例需要额外的 HTTP 服务器，如 Apache。

通过 SimpleWebRTC.js 可以看到建立浏览器之间的视频通话需要经过以下步骤：在本地和远端页面建立〈video〉元素，获得本地视频图像，主要采用 MediaStream API；与访问同一个 URL 即同一个会话的远端用户建立连接，主要用到 PeerConnection API；将本地视音频流通过建立的链接发送出去；接收远端来的数据流，在本地的〈video〉元素中播放。其中最为关键的是建立连接的信令部分。信令部分不限定，因而可有多种协议选择。其中之一的开源信令服务器 SignalMaster 基于 WebSocket 协议。

3.2.2　ChatRoom

ChatRoom 是基于 WebRTC.io 实现的开源视频聊天室。与 SimpleWebRTC 不同的是，其本身已包含 HTTP 服务器。ChatRoom 完全在 Node.js 上开发，采用了目前主流的基于 node 的网络框架—Express[15]。

结束语　本文详细阐述了 WebRTC 应用的架构和实现细节；随后以 AppRTC 和 CodeLab 为例介绍了两用户 WebRTC 视音频通信；接着以 SimpleWebRTC 和 ChatRoom 为例介绍了 WebRTC 多用户视音频通信的现状，以说明目前 WebRTC 并非不支持多用户的视频通话，只是 Google 目前不建议应用在多用户的场景。对于小规模的如 10 人以下的多人视频，这样的解决方案勉强可以满足性能上的基本需求。

2.3节中以PeerCDN、SwarmCDN、ShareFest为例介绍了基于WebRTC的P2P下载的发展现状，由此可以看出，除视频通话外，WebRTC在CDN与文件下载方面已经实现。表1总结了本文各种WebRTC实例的比较。

表1 基于WebRTC网页应用比较

	AppRTC	CodeLab	SimpleWebRTC	OpenTok	ChatRoom	PeerCDN	SwarmCDN	ShareFest
用户数目	2	2	≥3	≥3	≥3	≥3	≥3	≥3
信令机制	Channel API	WebSocket	WebSocket	多种方案	WebSocket	—	—	WebSocket
视频编解码	VP8	VP8	VP8	VP8	VP8	—	—	—
音频编解码	iSAC/iLBC	iSAC/iLBC	iSAC/iLBC	iSAC/iLBC	iSAC/iLBC	—	—	—
穿越	ICE	ICE	ICE	ICE	ICE	ICE	ICE	ICE
Web API	MediaStream PeerConnection	MediaStream PeerConnection	MediaStream PeerConnection	MediaStream PeerConnection	MediaStream PeerConnection	PeerConnection	PeerConnection	DataChannel
是否开源	是	是	是		是	否	否	是

WebRTC暂未统一为网页标准，但随着广大浏览器逐步集成，WebRTC作为一种便捷的沟通方式会得到越来越多的研究和应用，也会在更多的情景下采用WebRTC[33]。改进多用户视音频通信网络拓扑，降低网络带宽压力以提高通话质量是未来基于WebRTC多人视音频通信领域的重点和难点，因此今后的研究方向及可能面临的挑战主要有：

(1)基于WebRTC的P2P流媒体的方案研究。除简单一对一视音频之外，WebRTC可有更广阔的用途，例如一个公共事件的直播有一个或几个发言人和成百上千的观看者。目前WebRTC应对多人视频通信时采用全连接方式，随着用户数量的增加，每个节点的连接数随之增加，对上行下行带宽的压力也会越来越大，该结构的缺点将会越来越突出，因此这种方式无法应对成百上千个节点的情景。WebRTC已在浏览器中实现对等内容分发[34]。从前文中看到，目前实现了基于WebRTC的P2P下载，但尚未实现基于WebRTC的P2P流媒体传输。P2P模块的实现可以在网页应用层通过网页开发实现，也可以在WebRTC的架构内部即浏览器的底层传输机制中嵌入P2P的功能模块，进而在现有API的基础上，封装好支持P2P数据分发功能的API以供网页开发者调用。研究支持P2P多用户WebRTC是未来需要进一步研究的内容。

(2)集成MCU的WebRTC多用户视音频研究。对于成百上千数量级的用户，一个更好的选择是使用多点控制单元MCU (Multipoint Control Unit)，在大量参与者之间分发流媒体。MCU是一个起到桥梁作用的服务器，可以应对不同的分辨率、编解码器和帧速率的多人视音频通信，可做选择性流的转发、混合音频和视频流[35]。目前有几种开源MCU软件可供选择，例如，Licode (以前被称为Lynckia)为WebRTC提供了一个开源的MCU，OpenTok是在Mantis框架下通过西班牙电信公司Telefonica的云计算平台担当类似MCU的角色实现云端的WebRTC多人视音频通信，可惜这一技术并未开源。未来研究开发具备MCU的多人WebRTC多人视音频也是WebRTC研究的主要方向之一。

(3)移动终端的WebRTC视音频通信。随着移动互联网的迅速发展，移动终端设备的WebRTC视音频通信研究不可忽视。由于WebRTC中视音频编解码会占用较多的CPU资源，视音频数据的传输也需要较大的网络带宽和流量，而手机等移动终端的处理能力和网络相对有限，因此如何协调移动终端的处理能力和WebRTC视音频通信的需求也是未来需要进一步研究的内容。

参考文献

[1] Jennings C, Hardie T, Westerlund M. Real-time commu-nications for the web[J]. Communications Magazine, IEEE, 2013, 51(4): 20-26

[2] Bergkvist A, Burnett D C, Jennings C, et al. WebRTC 1.0: Real-time communication between browsers, W3C Working Draft 10 September 2013 [S/OL]. http://www.w3.org/TR/webrtc/

[3] Burnett D C, Bergkvist A, Jennings C, et al. Media Capture and Streams, W3C Working Draft 03 September 2013[S/OL]. http://www.w3.org/TR/mediacapture-streams/

[4] Singh K, Krishnaswamy V. A Case for SIP in Java-Script[J]. Communications Magazine, IEEE, 2013, 51(4): 28-33

[5] Amirante A, Castaldi T, Miniero L, et al. On the seamless interaction between WebRTC browsers and SIP-based conferencing systems[J]. Communications Magazine, IEEE, 2013, 51(4): 42-47

[6] 刘亚杰. P2P流媒体内容分发关键技术研究 [D]. 长沙：国防科学技术大学，2005

[7] 张国印，李军，王向辉，等. 一种基于移动P2P改进的Gossip算法[J]. 计算机科学，2013，40(9)：103-105

[8] Uberti J, Jennings C. Javascript Session Establishment Protocol, draft-ietf-rtcweb-jsep-03, August 29 2013 [S/OL]. https://tools.ietf.org/html/draft-ietf-rtcweb-jsep-03

[9] Handley M, Perkins C, Jacobson V. RFC 4566: SDP: session description protocol[S]. IETF, July 2006

[10] Ciurana E. Developing with Google App Engine[M]. Apress, 2009: 11-32

[11] Fette I, Melnikov A. RFC 6455: The WebSocket Proto-col[S]. IETF, December 2011

[12] Hickson I. The WebSocket API, W3C Working Draft 09 August 2012[S/OL]. http://www.w3.org/TR/2012/WD-websockets-20120809

[13] 孙清国，朱玮，刘华军. Web应用中的服务器推送技术研究综述[J]. 计算机系统应用，2008，17(11)：116-120

[14] Garrett J J. Ajax: A new approach to web applica-tions [EB/OL]. 2005. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

[15] Void B Y. Node.js开发指南[M]. 北京：人民邮电出版社，2012：80-97

[16] Ornbo G. Sams Teach Yourself Node.js in 24 Hours[M]. 傅强，陈宗斌，译. 北京：人民邮电出版，2013：148-185

[17] Jones P, Shabestary T, Alvestrand H, et al. RTP Payload Format for the iSAC Codec[S/OL]. 2013. https://tools.ietf.org/html/draft-ietf-avt-rtp-isac-04

[18] Duric A, Andersen S. RFC 3952: Real-time Transport Protocol (RTP) Payload Format for internet Low Bit Rate Codec (iLBC) Speech[S]. IETF, 2004

(下转第32页)

差的增加是由于相应的定位范围急剧增加所致。然而当节点密度到达某一个点后，大部分能定位的节点获得定位。随着节点密度的进一步增加，未定位的节点获得更多的参照节点，有更多的选择来计算位置，因此定位误差会大大减少。而 LOMPL 中的逐一迭代定位稍精准于 SLMP 的累加迭代定位，平均误差差距不大。

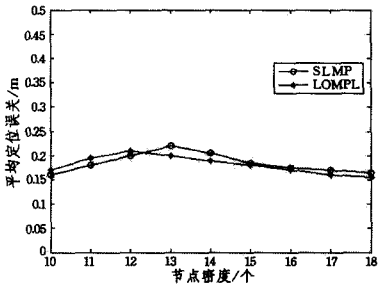

图 3　平均定位误差随节点密度变化的情况

结束语　针对大规模水声传感器网络，本文首先对水面网关部署的优化问题进行了研究，采用最低数量的网关节点进行最优位置的部署以确保网络的良好性能，并实现了多种目标的约束优化。随后提出了一种新的有移动预测定位的定位规划 LOMPL。运用水下移动模式的空间关联特点，使节点之间能够获得移动信息，从而预测出其位置。仿真结果显示了 LOMPL 能够在保持一个相当高的定位范围和定位精度的同时极大地减少通信能耗。下一步将在应用领域实现进一步拓展，增加更多的干扰模型，规划 MAC 协议的影响，进一步完善 LOMPL 所带来的定位优势。

参 考 文 献

[1] 何明，梁文辉，陈国华，等. 水下移动无线传感器网络拓扑[J]. 控制与决策，2013，28(12)：1761-1770

[2] Petrioli C，Petroccia R，Spaccini D. Time synchronization and localization for underwater acoustic sensor networks with the SUNSET framework[C]// Proceedings of the 8th ACM international workshop on Wireless network testbeds，experimental evaluation & characterization. ACM，2013：99-100

[3] Lloret J. Underwater Sensor Nodes and Networks[J]. Sensors，2013，13(9)：11782-11796

[4] Seah W K G，Tan H P. Multipath virtual sink architecture for wireless sensor networks in harsh environments[C]// Proceedings of the first international conference on integrated internet ad hoc and sensor networks. ACM，2006：19

[5] Ibrahim S，Cui J H，Ammar R. Efficient surface gateway deployment for underwater sensor networks[C]// IEEE Symposium on Computers and Communications，2008 (ISCC 2008). IEEE，2008：1177-1182

[6] Cui J H，Kong J，Gerla M，et al. The challenges of building mobile underwater wireless networks for aquatic applications [J]. Network，IEEE，2006，20(3)：12-18

[7] Guan J，Huang J，Lu J，et al. Topology structure design for underwater acoustic Wireless Sensor Network [C] // TENCON 2013-2013 IEEE Region 10 Conference (31194). IEEE，2013：1-4

[8] Pan Quan，Zhao Chun-hui，Liu Liu. A probability hypothesis density filter with Singer model for maneuver target tracking[C]// Control Conference (CCC). 2013：4778-4782

[9] Climent S，Sanchez A，Capella J V，et al. Underwater Acoustic Wireless Sensor Networks：Advances and Future Trends in Physical，MAC and Routing Layers [J]. Sensors，2014，14(1)：795-833

[10] Peng Zheng，Cui Jun-hong，Shi Zhi-jie，et al. Scalable Localization with Mobility Prediction for Underwater Sensor Networks [J]. IEEE Computer Society，2011，1(17)：335- 348

[11] Beerens S P，Ridderinkhof H，Zimmerman J T F. An analytical study of chaotic stirring in tidal areas [J]. Chaos，Solitons & Fractals，1994，4(6)：1011-1029

(上接第 6 页)

[19] Andersen S，Duric A，Astrom H，et al. RFC 3951：In-ternet low bit rate codec (iLBC)[S]. IETF，2004

[20] Valin J M，Vos K，Terriberry T. RFC 6716：Definition of the opus audio codec[S]. IETF，2012

[21] Bankoski J，Wilkins P，Xu Y. Technical overview of VP8，an open source video codec for the Web [C]// 2011 IEEE International Conference on Multimedia and Expo (ICME). Barcelona，2011：1-6

[22] Rosenberg J. RFC 5245：Interactive connectivity estab-lishment (ICE)：A protocol for network address translator (NAT) traversal for offer/answer protocols[S]. IETF，2010

[23] Petit-Huguenin M. RFC 5928：Traversal Using Relays around NAT (TURN) Resolution Mechanism[S]. IETF，2010

[24] Rosenberg J，Mahy R，Matthews P. RFC 5389：Session traversal utilities for NAT (STUN)[S]. IETF，2008

[25] 竹洪涛. 一种基于 SIP 和 WebRTC 的实时可视对讲方案设计[D]. 成都：西南交通大学，2013

[26] Mahy R，Matthews P，Rosenberg J. RFC 5766：Traversal using relays around NAT (TURN)：relay extensions to session traversal utilities for NAT (STUN)[S]. IETF，2010

[27] Perkins C，Westerlund M，Ott J. Web Real-Time Com-munication (WebRTC)：Media Transport and Use of RTP[S/OL]. 2012. https：//tools. ietf. org/html/draft-ietf-rtcweb-rtp-usage-06

[28] 郭远威. P2P 流媒体关键算法的研究[D]. 长沙：中南大学，2012

[29] Becke M，Rathgeb E P，Werner S，et al. DataChannel considerations for RTCWeb[J]. Communications Mag-azine，IEEE，2013，51(4)：34-41

[30] 张春红，裘晓峰，弭伟，等. P2P 技术全面解析[M]. 北京：人民邮电出版社，2010：149-164

[31] Pouwelse J，Garbacki P，Epema D，et al. The BitTorrent P2P file-sharing system：Measurements and analy-sis[M]// Peer-to-Peer Systems IV. Springer Berlin Hei-delberg，2005：205-216

[32] Johnston A B，Burnett D C. WebRTC：APIs and RTCWEB Protocols of the HTML5 Real-Time Web[M]. Digital Codex LLC，2012：9-11

[33] Johnston A，Yoakum J，Singh K. Taking on WebRTC in an Enterprise[J]. Communications Magazine，IEEE，2013，51(4)：48-54

[34] Nurminen J K，Meyn A J R，Jalonen E，et al. P2P media streaming with HTML5 and WebRTC[C]// IEEE International Conference on Computer Communications. Turin：IEEE，2013：63-64

[35] Xu Y，Yu C，Li J，et al. Video telephony for end-consumers：Measurement study of Google+，iChat，and Skype[C]// Proceedings of the 2012 ACM conference on Internet measurement conference. ACM，2012：371-384