

分类号	<u>TP399</u>	密级	<u>公开</u>
UDC	<u>004</u>	学位论文编号	<u>D-10617-30852-(2021)-02196</u>

重庆邮电大学硕士学位论文

中文题目	<u>基于 WebRTC 回声消除的音频系统</u>
	<u>研究与实现</u>
英文题目	<u>Research and Implementation of</u>
	<u>Audio System Based on</u>
	<u>WebRTC Echo Cancellation</u>
学 号	<u>S180231971</u>
姓 名	<u>沈励芝</u>
学位类别	<u>工程硕士</u>
学科专业	<u>计算机技术</u>
指导教师	<u>龙昭华 教授</u>
完成日期	<u>2021 年 5 月 28 日</u>

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的
研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含他
人已经发表或撰写过的研究成果，也不包含为获得 重庆邮电大学 或其他单位
的学位或证书而使用过的材料。与我一同工作的人员对本文研究做出的贡献均已
在论文中作了明确的说明并致以谢意。

作者签名：沈励芝

日期：2021 年 5 月 28 日

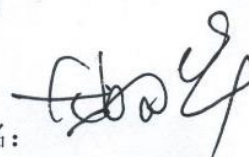
学位论文版权使用授权书

本人完全了解 重庆邮电大学 有权保留、使用学位论文纸质版和电子版的
规定，即学校有权向国家有关部门或机构送交论文，允许论文被查阅和借阅等。
本人授权 重庆邮电大学 可以公布本学位论文的全部或部分内容，可编入有关
数据库或信息系统进行检索、分析或评价，可以采用影印、缩印、扫描或拷贝等
复制手段保存、汇编本学位论文。

(注：保密的学位论文在解密后适用本授权书。)

作者签名：沈励芝

导师签名：



日期：2021 年 5 月 28 日

日期：2021 年 5 月 28 日

摘要

近年来,音视频即时通信技术广泛应用于医疗、军事、教育、安防控制等应用领域。然而,随着音视频即时通信在各种场景下的广泛应用,现有的技术呈现出许多不足,例如音视频不同步、存在回声干扰以及传输质量难以保证等。其中,回声干扰在很大程度上影响语音通信的质量从而导致通话质量受损,是亟待解决的重要问题。为了消除回声干扰,研究者们对其进行了广泛而深入的研究,现如今已取得较大的进展。为更好的推动数字化校园建设,本文以消除音视频即时通信中的回声为目标,重点对不稳定的声学回声消除进行研究。本文主要工作如下:

1. 本文对 WebRTC 的相关理论进行了研究,主要包括其发展现状、应用领域、整体框架和核心技术,深入研究 WebRTC 七大组件、开源 API 和 WebSocket 协议等。重点分析 WebRTC 回声消除模块的实现原理,深入研究 WebRTC 自适应滤波回声消除算法中的最小均方(Least Mean Square, LMS)和递归最小二乘(Recursive Least Square, RLS)两类算法并分析各类算法的优缺点。为本文所提出的方法做一定的理论准备。

2. LMS 算法计算复杂度低、易实现,但它收敛效果不佳。针对此问题,本文在最小均方算法基础上对归一化最小均方(Normalized LMS, NLMS)算法进一步研究并进行改进。通过仿真实验平台验证以及实验结果表明,改进后的归一化最小均方算法相对于最小均方算法和归一化最小均方算法具有收敛速度快、稳态误差低和跟踪能力强的优点。

3. 设计并实现基于 WebRTC 回声消除的课堂会议系统。系统的开发工作包括架构设计、服务器端配置部署和客户端设计实现。本文选取 SFU 作为系统的通信架构。服务端工作包含三大模块部署,分别为流媒体模块、信令模块和网络地址转换穿透模块。客户端工作包括环境搭建、通信原理研究及各功能模块实现。

4. 在多用户使用的情况下,本文对系统功能模块进行测试并收集有关数据进行性能分析及回声消除分析。测试结果表明,本系统具有跨应用平台、稳定和低功耗的优点。最后通过实际的课堂场景应用表明本系统具有实际的应用价值。

关键词: WebRTC, 自适应滤波, 回声消除, 课堂会议

Abstract

Recently, audio and video instant messaging technology has been widely used in medical, military, education, security control and other application fields. However, with the widespread application of audio and video instant messaging in various scenarios, the existing technology presents many deficiencies, such as unsynchronized audio and video, echo interference, and difficulty in guaranteeing transmission quality. The presence of echo affects the quality of voice communication, to a large extent, resulting in loss of call quality, which is an important issue to be solved urgently. In order to eliminate echo, researchers have carried out extensive and in-depth research on it. Nowadays, great progress has been made. In order to better promote the construction of digital campus, this thesis aims to eliminate the echo in audio and video instant messaging, and focuses on the research of unstable acoustic echo cancellation. The main works of this thesis are as follows:

1. This thesis studies the related theories of WebRTC, including its development status, application fields, overall framework and core technology, and in-depth study of the seven major components of WebRTC, open source API and WebSocket protocol. Thesis focus on the analysis of the implementation principle of the WebRTC echo cancellation module, and in-depth study of the Least Mean Square (LMS) and Recursive Least Square (RLS) algorithms in the WebRTC adaptive filter. Analyzing the advantages and disadvantages of the two various types of algorithms. Make certain theoretical preparations for the method proposed in this thesis.

2. The LMS algorithm has low computational complexity and is easy to implement, but its convergence effect is not good. To solve this problem, thesis further researches and improves the normalized least mean square (Normalized LMS, NLMS) algorithm on the basis of the LMS algorithm. Verification by the simulation experiment platform and experimental results show that the improved NLMS algorithm has the advantages of fast

convergence, low steady-state error and strong tracking ability compared to the LMS algorithm and the typical NLMS algorithm.

3. Design and implement a classroom conference system based on WebRTC echo cancellation. The development process of the system includes architecture design, server-side configuration and deployment, and client-side design and implementation. This thesis selects SFU as the communication framework of the system. The server-side work includes the deployment of three modules, namely the streaming media module, the signaling module, and the network address translation penetration module. The client's work includes environment construction, communication principle research and implementation of various functional modules.

4. Under the case of multi-user circumstances, this thesis tests the system function modules and collects relevant data for performance analysis and echo cancellation analysis. The test results show that the system has the advantages of cross-application platform, stability and low power consumption. Finally, the actual classroom scene application shows that the system has practical application value.

Keywords: WebRTC, adaptive filter, echo cancellation, classroom conference

目录

图录	VII
表录	IX
第 1 章 引言	1
1.1 研究背景及意义	1
1.2 研究现状	3
1.2.1 WebRTC 技术研究现状	3
1.2.2 回声消除算法研究现状	3
1.3 论文研究内容	5
1.4 论文组织结构	5
第 2 章 WebRTC 相关基础理论	7
2.1 WebRTC 架构及核心技术	7
2.1.1 整体架构	7
2.1.2 WebRTC 组件介绍	8
2.1.3 Javascript 接口	9
2.1.4 WebSocket	11
2.2 WebRTC 回声消除技术	13
2.2.1 回声消除原理	13
2.2.2 自适应滤波器	14
2.2.3 自适应滤波算法的性能指标	16
2.3 本章小结	17
第 3 章 WebRTC 自适应回声消除算法分析及改进	18
3.1 自适应回声消除模块	18
3.2 自适应回声消除常见算法介绍	19
3.2.1 LMS 算法与 RLS 算法	20
3.2.2 NLMS 算法	27
3.3 改进的 NLMS 算法	28
3.3.1 理论研究	28

3.3.2 实验结果及分析	30
3.4 本章小结	32
第 4 章 基于 WebRTC 回声消除的课堂会议系统设计与实现	33
4.1 需求分析	33
4.1.1 功能性需求分析	33
4.1.2 非功能性需求分析	35
4.1.3 WebRTC 回声消除接口分析	36
4.2 总体设计	37
4.2.1 系统通信模型	37
4.2.2 系统总体设计	40
4.2.3 服务端设计与实现	41
4.2.4 客户端设计与实现	45
4.3 功能模块设计与实现	49
4.3.1 用户注册登录模块	49
4.3.2 音视频课堂会议模块	50
4.3.3 课堂交流讨论模块	53
4.3.4 课程管理模块	54
4.3.5 互动式课堂签到模块	55
4.4 软件测试	56
4.4.1 测试环境	56
4.4.2 系统效果展示	56
4.4.3 系统功能测试	59
4.4.4 客户端性能测试	63
4.4.5 系统回声消除测试	65
4.5 本章小结	67
第 5 章 总结与展望	68
5.1 主要工作与创新点	68
5.2 展望	68
参考文献	70

致谢	76
攻读硕士学位期间从事的科研工作及取得的成果	77

图录

图 2.1 WebRTC 整体架构	7
图 2.2 获取本地媒体流	9
图 2.3 约束条件设置	10
图 2.4 创建 TCP 连接	11
图 2.5 建立 WebSocket 连接	12
图 2.6 WebSocket 全双工通信模型	13
图 2.7 回声消除原理图	14
图 2.8 自适应滤波器结构图	15
图 3.1 WebRTC 回声消除模块框架图	18
图 3.2 LMS 算法实现效果展示	22
图 3.3 RLS 算法实现效果展示	24
图 3.4 不同步长对 LMS 算法的影响	25
图 3.5 不同遗忘因子对 RLS 算法的影响	26
图 3.6 不同遗忘因子对 RLS 算法增益向量的影响	26
图 3.7 不同步长对 NLMS 算法的影响	28
图 3.8 误差曲线对比	31
图 3.9 跟踪性能学习曲线对比	31
图 4.1 系统结构原理图	33
图 4.2 AEC 处理过程	37
图 4.3 Mesh 架构图	38
图 4.4 MCU 架构图	39
图 4.5 SFU 架构图	39
图 4.6 系统架构图	40
图 4.7 系统的整体功能模块	41
图 4.8 Janus 整体架构图	41
图 4.9 coturn 服务器配置图	45

图 4.10 coturn 服务器配置成功显示图 45

图 4.11 depot_tools 工具配置成功显示 46

图 4.12 基于 WebRTC 的用户音视频通信呼叫过程 48

图 4.13 用户注册登录模块流程图 50

图 4.14 音视频课堂会议模块流程图 51

图 4.15 回声消除执行过程 52

图 4.16 课堂交流讨论模块流程图 53

图 4.17 课程管理模块流程图 54

图 4.18 互动式课堂签到模块流程图 55

图 4.19 用户注册登录模块效果展示 56

图 4.20 音视频课堂会议模块效果展示 57

图 4.21 课程管理模块效果展示 57

图 4.22 课堂交流讨论模块效果展示 58

图 4.23 互动式课程签到模块效果展示 58

图 4.24 课堂通 APP 电流、电压、温度及内存占用测试对比展示图 64

图 4.25 六台设备接入回声测试界面 65

图 4.26 远端信号输入波形 66

图 4.27 远端处理后的回声信号 66

图 4.28 近端信号输入波形 66

图 4.29 近端处理后的信号波形 66

图 4.30 双讲模式下近端参考信号波形 67

图 4.31 双讲模式下处理后的信号波形 67

表录

表 3.1 LMS 算法步骤 21

表 3.2 RLS 算法步骤 24

表 3.3 LMS 算法与 RLS 算法对比 27

表 3.4 改进的 NLMS 算法步骤 30

表 4.1 Android 端实现音视频通信的主要类和方法 47

表 4.2 服务器配置信息 56

表 4.3 手机端测试设备配置信息 56

表 4.4 用户注册模块功能测试用例表 59

表 4.5 用户登录模块功能测试用例表 60

表 4.6 音视频课堂会议模块功能测试用例表 60

表 4.7 课堂交流讨论模块功能测试用例表 61

表 4.8 课程管理模块功能测试用例表 62

表 4.9 互动式课堂签到模块功能测试用例表 63

第 1 章 引言

1.1 研究背景及意义

近年来随着互联网产业的不断发展以及在国家政府对互联网行业大力支持的背景下，与最早的 1996 年即时通信(Instant Messaging, IM)业务不同^[1]，如今的即时通信已经发展成为综合化交流平台^[2]，主流的发展方向包括医疗、军事、教育、安防控制等。与此同时，伴随着通信技术的飞速发展，实时音视频的需求也在不断增多。相比早期的书信，电话，电子邮件等方式，实时音视频满足了无障碍交流的需求。因此，多种类型的即时通信音视频系统应运而生，主要分为硬件即时通信音视频系统、软件即时通信音视频系统以及网页端即时通信音视频系统。硬件即时通信音视频系统需要嵌入式环境支持，软件辅助的方式实现，虽然稳定性高，但是花费成本大、搭建环境复杂、不方便使用。软件即时通信音视频系统针对不同场景具有不同功能而倍受用户广泛关注，各大软件的使用率也是位居世界榜首，例如微信、QQ 等。网页端即时通信(Web Real Time Communication, WebRTC)音视频系统因无需考虑软硬件环境的优点，受到广大用户的喜爱。也因它无需安装任何插件和拥有开源 API 供开发者调用的特点，更是迅速占领了各大 Web 市场。

同时随着智能移动终端的发展，各种实时音视频、视频直播互动等移动终端的 APP 和小程序不断涌现出来。无论是哪种终端的通信方式都从最初的仅支持发送消息、文件或者购物等单一功能模式逐步向各种功能一体化的形态发展，实时交互的要求更高。Android 系统作为移动终端市场的主要操作系统之一。各种国产品牌的手机大多仅支持 Android 系统，如华为、小米、Oppo 等。WebRTC 技术为 Android 开发平台提供了原生代码库，但目前基于 WebRTC 技术的开发大多集中在网页端，针对移动端开发相对较少，因此本文重点研究并实现了基于 Android 的 WebRTC 回声消除的音频系统。

最早的视频会议是诞生于 1964 年由美国贝尔实验室采用模拟技术研发出的可视电话。但是由于效果不佳，几乎不被研究者看好，随着视频会议相关协议和标准不断发展和完善，该技术被广泛应用于政府、金融、环境、教育等领域。在此期间，视频会议经历了三个阶段：模拟技术会议阶段、专用网数字会议阶段及基于 IP 网

络视频会议阶段。20 世纪 60 年代，第一代会议系统采用全模拟技术，在两个终端之间进行数据传输，此时传送的效果只能是黑白图像且还要占用很大的带宽。到 70 年代中期，数字图像传输和语音编解码技术有了显著的进步，使得模拟视频会议向数字视频会议转型，进一步在某些地区开始形成了专用电话网。这个时期由于各地使用的标准不一，主要通过卫星、光纤等专用网络来连接视频会议系统，尽管图像质量和可靠性都得到了提升，但是设备费用高且难以实现国际视频会议。90 年代开始，随着互联网技术的高速发展，网络带宽的不断提升，基于 Internet 的硬件方式视频会议和软件方式视频会议都得到了广泛应用。硬件方式视频会议带动了一些经济效益的产生，典型受大众深爱的企业有思科、华为、飞视美等。软件方式视频会议更是因其价格低廉、功能独特、使用方便等特点备受广大开发商和使用者的喜爱。此阶段视频会议的发展离不开协议标准的建设，典型的有音视频编解码协议，如 iSAC、iSAB、H.323、H.264 协议等，信令控制协议，如 SIP(Session Initiation Protocol)等。

传统的音视频会议系统投入成本高且使用场景十分有限，会议质量受硬件设备及网络带宽影响较大。相比较而言，本文设计的课堂会议系统具有低成本且方便用户使用的优点，在满足课堂交流需求的同时具有良好的实时协同交流特点，能够显著提高课堂效率，具有广泛的应用空间。实时音视频会议的重点是保障实时的音频和视频的传输质量，以及多用户实时交互时的响应速度。在 WebRTC 开源框架中，首先是通过音视频采集模块对音视频数据进行采集，再经过网络传输至接收方，接收方接收数据后，对音视频数据作同步处理进行播放，从而确保音视频通信质量。在对音视频进行采集的过程中，音频回声是一个困扰开发者以及用户的问题，所以对音频数据进行回声消除处理是一个重要的研究点。随着 WebRTC 技术的广泛应用，不少专家学者提出了 WebRTC 回声消除改进算法，但仍有一些不足，如无法达到可靠平衡收敛效果和稳定误差的效果。为此，本文将重点研究基于 WebRTC 的自适应回声消除算法并对其改进，使得提升算法收敛速度的同时，能够保证稳定误差的性能，同时设计并实现基于 WebRTC 回声消除的音频系统。

1.2 研究现状

1.2.1 WebRTC 技术研究现状

2010 年, Google 收购了 Global IP Solutions 公司, 并因此获得了该公司名下的 GIPS 技术, 随后将其更名为 WebRTC 技术^[3, 4]。2011 年 6 月 Google 将此技术开源, 并同多家企业联合支持将它纳入万维网联盟的 W3C(The World Wide Web Consortium)推荐标准。随后经历了六年时间, W3C WebRTC 1.0 标准草案正式定稿。W3C 中定义了 PeerConnection、DataChannel 和 MediaStream 等 Javascript API 的细节^[5, 6]。目前 Opera、Safari、Chrome 和 Firefox 等各大浏览器厂商基于此标准广泛使用 WebRTC 技术。据全球网络实时通信市场研究报告显示: 2017-2021 年期间, WebRTC 全球使用率以 34.37% 增长率快速增长。

WebRTC 是一项无需插件且无需客户端下载的技术, 是 HTML5 标准之一。这种技术使用户终端之间有了对等通信的能力, 并且在延时、花费成本和系统安全性各方面都有了明显的优势。随着浏览器供应商对此技术的大力支持, 这些新标准、开源应用程序编程接口和协议正在对全世界的互联网发展产生重大影响。采用 WebRTC 技术开发实时通信, 最近几年受到很多关注和宣传^[7], 较少讨论的是基于移动终端平台开发适应各高校场景下的实时音视频通信应用, 其中典型的课堂通信系统通常需要支持两方电话或多方会议场景。

总的来说, WebRTC 有开源免费, 拥有用户终端音视频数据读取能力, 可在两个终端用户实例之间创建一个端到端的安全媒体路径和易于开发维护等实用性^[8]。尽管国家为建立 IT 网络基础设施, 教室和实验室做出了不少努力, 但就高校的网络环境而言, 拥塞和互联网访问问题仍然存在, 对教育系统的有效性产生了负面影响。利用 WebRTC 技术开发课堂会议应用系统, 提升课堂效率的同时, 解决高校的实时通信难题, 降低通信费用, 提高工作效率。

1.2.2 回声消除算法研究现状

在信息技术全球化的浪潮下, 人们对实时音视频传输质量的要求也越来越高, 回声问题也是当前的重点研究课题。在网络频繁切换状态下, 实时语音质量是终端

用户关注的问题。线路回声是由于二四转换阻抗不匹配导致的,通过硬件设备的处理能够有明显的抑制效果,本文不将其纳入重点研究范围。声学回声的产生是由于部分声音信号直接或间接地再一次反馈到接收方,致使语音信号受到干扰,影响音频质量。现有的声学回声消除技术主要有麦克风阵列技术、回声抑制器技术和自适应滤波器技术^[9]。但前两者只是通过硬件设备对语音信号进行处理,成本花销大且效果不佳。现在常用的回声消除技术是采用自适应滤波器,因此本文对其重点研究。传统的声学回声消除技术是从国外二十世纪七十年代的早期算法中发展而来的,这种方法的成本消耗大。

算法性能的好坏直接决定自适应滤波器性能,对回声消除有着重要作用。针对回声具有稀疏性,不少研究者通过研究稀疏算法来提高自适应系统的效率。如文献[10]指出采用P范数约束来修改经典LMS算法的代价函数,将经典的P范式概念分解为一个新的非均匀范式定义,以实现范数约束的定量调整。文献[11]指出用泰勒级数近似值得出瞬时行为,建立参数选择规则以更新代价函数。文献[12]提出每个抽头位置的自适应增益随长度的不同而变化且与权重的绝对值成比例动态调整。但这些算法一般只适用于回声路径稀疏或高度非稀疏的情况下。

现在主流的回声消除算法主要分为递归最小二乘自适应算法和最小均方自适应算法。递归最小二乘算法的收敛速度快,但是算法结构复杂,计算复杂性高^[13]。最小均方算法由于其简单性和鲁棒性成为最受欢迎的自适应回声消除算法之一^[14]。因此现在主流的回声消除算法是在LMS算法上衍生出来的。但在声学回声消除时延不确定性的环境中,自适应滤波器的长度必须不断增加,这会导致计算量大和收敛速度慢等问题^[15]。最早的LMS算法是由Windrows和Hoff最先提出来,因其具有原理简单、计算复杂度低、易实现的特点被许多研究者采用,并在此基础上针对LMS算法收敛速度慢的特点进行研究并提出很多改进的算法。文献[16]提出基于辛赫函数并结合最大相关熵规则,加快了算法的收敛速度,但该算法的主要改善对象是低频振荡模式的自适应滤波器。文献[17]中N. Iqbal等人提出变换域与稀疏性感知相结合的组合滤波器算法。文献[18]中M. Salman等人针对脉冲噪声环境下提出一种变步长LMS算法,在该算法的代价函数中使用了反三角函数约束。文献[19]指出S. Rahman等人针对变换域提出用于非平稳过程的离散小波变化的LMS算法。文献[20]中针对平稳信号提出一种基于步长因子和输入信号之间非线性关系函数

的改进 LMS 算法。本文将重点介绍最受欢迎的自适应滤波器算法之一：归一化最小均方算法。它之所以备受研究者关注，是因为它原理简单且易于实施。该算法的稳定性由步长参数控制。众所周知，在稳定性条件下，步长因子的增大一方面反映了算法具有快速收敛和良好的跟踪能力，另一方面又反映了算法具有较低的稳定失调能力。为了平衡这个矛盾的要求，需要更加精准地控制步长因子。尽管这个问题的表述很简单，但是要找到一个效果好且可靠的解决方案并不容易。

1.3 论文研究内容

本文以音频系统为导向，主要对 WebRTC 技术进行研究，分别从两个方面进行了创新及研究：一是研究 WebRTC 框架并分析其关键技术，接着深入研究 WebRTC 自适应回声消除算法并对其进行了改进；二是设计与实现基于 WebRTC 回声消除的音频系统。详细的研究内容如下。

首先，介绍本文的基础理论背景知识，对实现 WebRTC 音频系统的实时通信技术作重点分析，为第四章的系统设计做铺垫。

接着对 WebRTC 框架中回声消除模块深入研究，重点聚焦至 LMS 算法和 RLS 算法研究。通过理论研究及对比实验研究分析，RLS 算法以牺牲计算复杂度为代价，引入了迭代矩阵，因此获得了比 LMS 算法更优的收敛速度，但是其实现原理更加复杂^[21]。针对实用性方面考虑，本文在 LMS 算法基础上进一步地研究文献[22]和文献[23]中介绍的 NLMS 算法，针对算法存在收敛速度慢的不足之处，提出改进的 NLMS 算法并通过仿真平台 Matlab 进行验证。

最后，结合实际的课堂应用场景，研究并实现基于 WebRTC 回声消除的音频系统，主要工作包括架构选取、服务器端及客户端设计。服务器端负责保证客户端间的实时通信、媒体流传输及存储；客户端设计主要包括音视频媒体流的采集及系统功能模块实现。在设计开发过程中，通过实际运用，对该系统的所有功能进行不断测试及改进，使系统的性能达到稳定状态。

1.4 论文组织结构

本文以国内外即时通信技术为课题背景，详述 WebRTC 技术的发展历程以及

目前国内外市场的应用情况，研究基于 WebRTC 的回声消除现状并将在后文重点研究。本文了解到目前针对教育方向音视频会议系统开发尚有缺口，并明确了目前中小学以及各高校线上课堂会议的需求，最终实现基于 WebRTC 开源框架的音频系统，并通过实际教学场景应用对该系统进行测试。

第1章是引言，主要介绍即时通信的研究现状，明确 WebRTC 开源框架的优缺点，阐述 WebRTC 技术的应用领域。进一步地，介绍 WebRTC 回声消除算法的研究现状。最后对本文的研究内容及组织结构进行说明。

第2章是相关技术知识介绍。主要分为两部分，首先主要对 WebRTC 技术框架原理进行分析，包括整体架构分析、WebRTC 组件介绍、音视频会议系统接口介绍及协议介绍，为后续实现基于 WebRTC 回声消除的课堂会议系统提供理论基础。第二部分是对 WebRTC 回声消除技术的理论知识进行研究，为第3章自适应滤波回声消除算法奠定理论基础。

第3章在第2章的基础上，进一步地对 WebRTC 自适应回声消除原理进行深层次研究，是论文的主体部分。本章首先对回声消除模块的工作原理进行介绍，重点研究自适应滤波回声消除模块，进一步对多种自适应回声消除算法进行原理分析及仿真平台实验验证：最小均方算法、递归最小二乘算法和 NLMS 算法。随后，本文基于 NLMS 算法研究进行改进，改善因无法更精准的调整步长因子和未考虑背景噪声所导致算法收敛速度慢的问题，并通过仿真实验验证改进后的算法性能更优。

第4章是系统的设计实现以及效果展示。在前三章的基础上，结合项目的需求，设计实现基于 WebRTC 回声消除的课堂会议系统，并对系统进行功能测试，通过实际运用场景进行性能测试及回声消除测试，验证系统的稳定性及回声消除有效性，是论文的主体部分。本系统的功能主要包括用户注册登录、音视频课堂会议，课堂交流讨论、课堂管理以及互动式课堂签到等。

第5章是总结与展望，对本文工作进行总结，包括本文的主要工作及本文工作不足之处，并在此基础上提出展望。

第 2 章 WebRTC 相关基础理论

2.1 WebRTC 架构及核心技术

2.1.1 整体架构

实时通信 RTC 有很多好处，但是由于昂贵的音视频环境部署类的问题，给研究界带来了几项挑战^[24]。网页即时通信 WebRTC 提供了实时通信系统的核心技术，它的原理是独立封装音视频捕获及处理模块、网络传输模块及会话控制等协议模块供开发人员调用。它无需安装任何插件也无需安装客户端就能带给用户极度轻便的体验。WebRTC 是一个对等开源框架，被认为是标准、协议和 Javascript 的集合^[25]，WebRTC 的整体架构如图 2.1 所示。

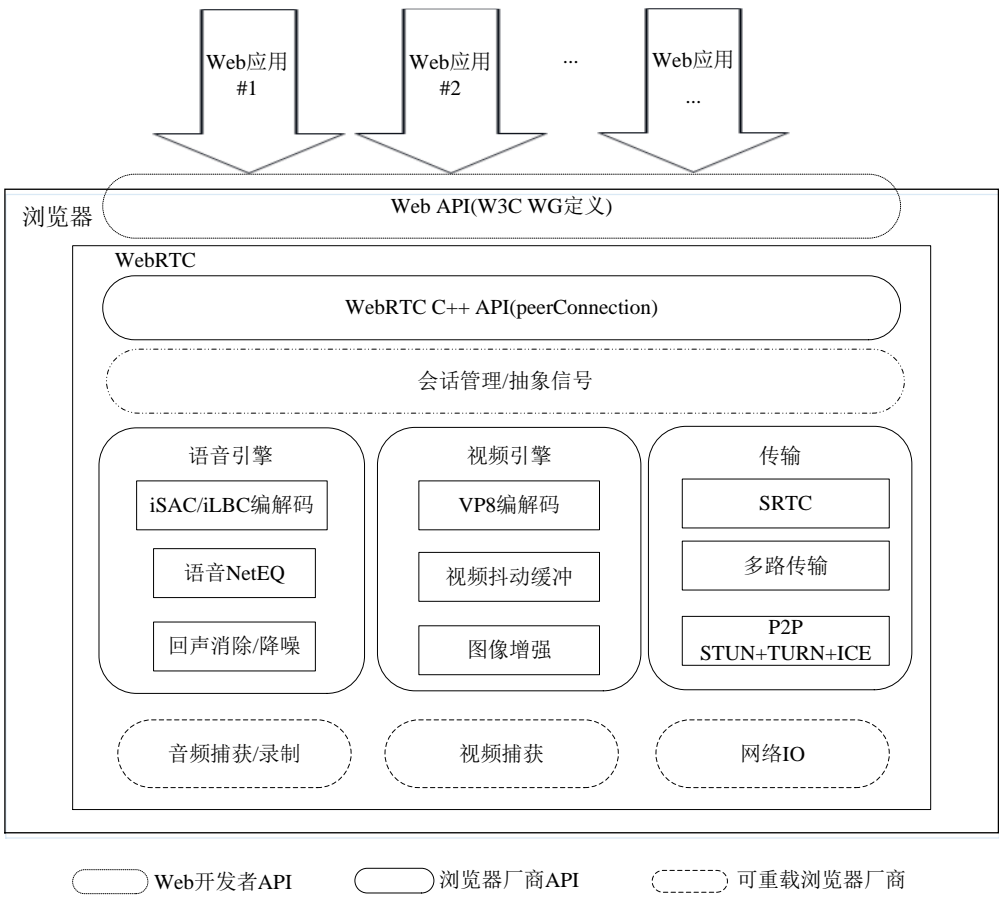


图 2.1 WebRTC 整体架构

WebRTC 架构模型包括 Web 应用、浏览器和 Web API 三大模块^[26]，下面将按

照从上至下的顺序依次介绍这些模块。

2.1.2 WebRTC 组件介绍

1. Web 应用(Your Web App)

开发人员可以直接调用 Web API 开发某些应用，如音视频应用、电子商务应用及娱乐资讯运用等。

2. Web API

包含用于在网络上进行实时通信的基本组件。有了这些组件，开发人员就可以通过访问需要的 API 对应地创建 Web 应用程序。

3. WebRTC C++ API

WebRTC C++ API 是面向浏览器厂商的本地 C++ API。虽然各个浏览器厂商的实现方式不同，但都是严格按照 W3C 标准执行。

4. 会话管理/抽象信号

会话管理/抽象信号是一个抽象的会话层，用来提供建立会话和会话管理，以便把协议的会话实现留给应用开发者。由于 WebRTC 提供实时的音视频传输，采用了 RTP(Real-time Transport Protocol)协议栈，即 RTP/RTCP(Real-time Transport Control Protocol)等相关的协议，从而提供实时传输功能和流量控制作用。

5. 语音引擎

语音引擎的主要任务是回声消除(Acoustic Echo Canceller, AEC)和降噪(Noise Reduction, NR)。回声消除是一种改善声音质量，消除产生的回声或防止其发生的方法。降噪是从信号中去除噪声的过程。音频机制主要分为 iSAC 和 iLBC 两大类编解码器。iSAC 编解码器代表由 Global IP Solutions 开发的宽带音频编解码器。该编解码器适用于音频流，并于 2011 年 6 月成为 WebRTC 技术的一部分。Global IP Solutions 还是 iLBC 编解码器的创建者。该窄带音频编解码器适用于 IP 上的语音通信。

6. 视频引擎

视频引擎负责图像增强和视频抖动缓冲。VP8 编解码器是视频引擎的主要视频压缩格式，目前得到 Chrome, Mozilla 和 Opera 的支持^[27]。图像增强是通过使用某种软件进行操作来提高数字图像质量的过程，包括明暗度检测、降噪处理等功能

[28, 29]。视频抖动缓冲器分为抖动和缓冲两部分。抖动模块主要提供稳定渲染视频并显示的效果。缓冲模块主要处理异常情况，如媒体流出现丢包、乱序、延迟到达的情况。

7. 传输机制

传输机制负责 SRTP (Secure Real-time Transport Protocol, 安全实时传输协议), P2P (Peer-to-Peer) 和 ICE 交互式连接建立。

2.1.3 Javascript 接口

WebRTC API 的主要有三大组件：**MediaStream**：允许 Web 浏览器访问摄像头和麦克风；**RTCPeerConnection**：设置音频或视频通话；**RTCDataChannel**：允许浏览器通过对等连接发送数据。

1. MediaStream

在获取媒体流的过程中，首先需要访问用户设备并获取相应的权限，如获取音频设备的麦克风和获取视频设备的摄像头。用户对授权权限进行管理，之后不再进行询问，其中 `getUserMedia()` 方法是访问本地输入设备的主要方式。输入设备选择由 `mediaStream` 对象进行控制，`mediaStreamTrack` 对象可以用来表示同一设备的不同音视频轨道。如图 2.2 是检测浏览器是否支持获取输入设备的相关媒体流的部分重要代码。

```
1 navigator.mediaDevices.getUserMedia = navigator.mediaDevices.getUserMedia ||
2 navigator.mediaDevices.webkitGetUserMedia ||
3 navigator.mediaDevices.mozGetUserMedia ||
4 navigator.mediaDevices.msGetUserMedia;
5 if (navigator.getUserMedia) {
6     // 支持
7 } else {
8     // 不支持
9 }
10 navigator.mediaDevices.getUserMedia({ video: true, audio: true }).then( ).catch( );
```

图 2.2 获取本地媒体流

`getUserMedia()` 方法内置 3 个参数：分别是对象，成功和失败回调函数。其中对象里面包含捕获对象，表示要获取的媒体设备，即要求获取的是摄像头还是麦克风，更加精准的媒体配置信息可以进一步设置。如图 2.3 是约束条件的部分重要代码。

```
1 navigator.mediaDevices.getUserMedia({
2   // 以下就是约束条件
3   video: true,
4   audio: true
5 })
6   .then(createConn )
7   .catch(
8     console.log(`getUserMedia() error: ${e.name}`);
9   );
10
11 // 或者约束条件可以提出来单独设置
12 let constraints = {
13   video: true,
14   audio: true,
15   ...
16 }
17 navigator.mediaDevices.getUserMedia(constraints, onSuccess, onError)
```

图 2.3 约束条件设置

2. RTCPeerConnection

在 WebRTC 框架中，媒体流的传输不经过服务器，直接在客户端之间传输。但是客户端之间先需要通过服务器进行信令连接。由于客户端和浏览器一般情况下都是处于 NAT（Network Address Translation，网络地址转换）或防火墙之后^[30]，其 IP 地址属于单位私有地址，在非同一局域网的情况下，客户端相互之间是无法发现彼此的，因此也无法直接建立 P2P 信道。那么要实现 P2P 的媒体通信，首先需要客户端与信令服务器建立连接之后，通过信令服务器的转发来实现对等端协商数据信息交换。当然，信令的作用并不止于此，基于 WebRTC 通信中，信令的作用有主要以下四个方面：(1)进行媒体协商（交换媒体信息、候选地址信息和媒体加密信息）；(2)标识和保证会话者身份；(3)控制媒体会话的相关信息；(4)提供双方会话管理机制，如主动请求和终止会话。RTCPeerConnection 接口屏蔽掉内部的复杂性，封装了大量的编解码、通信协议接口来实现整个实时通信过程。

3. RTCDataChannel

RTCDataChannel 表示一个数据通道，并且允许通信过程中的对等双方进行任意数据交换。它依赖于 RTCPeerConnection 对象传输自定义数据，如文字、文件和图片等，其本身不能单独使用。即在使用 RTCDataChannel 时必须先创建 RTCPeerConnection 对象,然后创建 RTCDataChannel 对象。RTCDataChannel 的具体使用过程类似于 WebSocket，接收端主要包含 onmessage()方法监听数据，on()方法来接收事件和 send()方法发送数据。使用 RTCDataChannel 传输数据时，分为可靠模式和不可靠模式，可靠模式保证数据按序到达并且保证数据传送成功，但是这种方式开销大且传输速率慢；不可靠模式不保证数据按序到达且传输不一定发送成功。显然不可靠模式的效率会更高，在不丢包的情况下，是值得考虑的。

2.1.4 WebSocket

接下来将介绍视频会议的另一个重要组成部分信令服务器，它所采用的主要技术之一就是 WebSocket。WebSocket 通信协议于 2008 年被提出，2011 年被 IETF（The Internet Engineering Task Force，国际互联网工程任务组）定为标准 RFC 6455，并由 RFC 7936 补充规范，随后被纳入 HTML5 规范中^[31]，它在客户端上实现的 WebSocket API 也被 W3C 制定为接口标准。

WebSocket 协议依赖 HTTP（HyperText Transfer Protocol，超文本传输协议）发起请求来完成“握手”。但是 WebSocket 从根本上解决了 HTTP 不能解决的问题，采用非常简短的请求头信息来发送信息，有效数据占比高。同时 WebSocket 没有同源限制并且它可以传输各种各样格式的消息，例如二进制文件、文本信息或者自定义对象消息。

一般来讲，Websocket 默认端口是 80 端口。随着用户对网络安全性要求不断提高，引入 TLS（Transport Layer Security，安全传输层协议）进行安全加密传输，默认端口为 443。建立 WebSocket 连接的步骤如下：

1. 建立 TCP 连接

因为 WebSocket 是基于 TCP 协议的，所以先要建立一个 TCP 连接。这里创建一个 TCP 服务，监听 7000 端口。部分重要代码如图 2.4 所示。

```
1 var net = require('net');
2 net.createServer(function(socket) {
3   console.info('tcp client connected');
4   socket.on('data', function(data) {
5   });
6   socket.on('end', function() {
7     console.info('client disconnected');
8   });
9 }).listen(7000);
```

图 2.4 创建 TCP 连接

如果在浏览器中执行如下代码后：

```
var ws = new WebSocket("ws://localhost:7000");
```

在服务端控制台看到 tcp client connected 的输出，说明已经建立 TCP 连接。

2. 建立 WebSocket 连接

首先客户端会发送一个握手协议包。握手协议包的报文格式必须符合 HTTP 报文格式规范。其中：

- (1) 方法必须为 GET 方法。
- (2) HTTP 版本不能低于 1.1。
- (3) 格式中必须包含 Upgrade 头部，值必须为 websocket。
- (4) 格式中必须 WebSocket 版本头部信息且该值为 13。

服务端验证客户端的握手协议包之后，若符合规范也会发送一个握手协议包给客户端。格式如下：

- (1) 格式中必须包含 Connection 头部。
- (2) 格式中必须包含一个 Upgrade 头部，值必须为 websocket。
- (3) 格式中必须包含一个 Sec-WebSocket-Accept 头部。

客户端收到服务端的握手包之后，验证返回报文信息是否符合规范，计算 Sec-WebSocket-Accept 并与服务端握手包里的值进行比对。其中任何一步不通过则不能建立 WebSocket 连接。部分关键代码如图 2.5 所示。

```
1 //index.js
2 var net = require('net');
3 var crypto = require('crypto');
4 var wsGUID = "258EAF5-E914-47DA-95CA-C5AB0DC85B11";
5 net.createServer(function(socket) {
6   console.info('tcp client connected');
7   socket.on('data', function(data) {
8     var dataString = data.toString();
9     key = getWebSocketKey(dataString),
10    acceptKey;
11    if (key) {
12      console.info(key);
13      acceptKey = genAcceptKey(key);
14      socket.write('HTTP/1.1 101 Switching Protocols\r\n');
15      socket.write('Upgrade: websocket\r\n');
16      socket.write('Connection: Upgrade\r\n');
17      socket.write('Sec-WebSocket-Accept: ' + acceptKey + '\r\n');
18      socket.write('\r\n');
19    }
20  });
21  socket.on('end', function() {
22    console.info('client disconnected');
23  });
24 }).listen(7002);
25 function getWebSocketKey(dataStr) {
26   var match = dataStr.match(/Sec-WebSocket-Key:\s(.+)\r\n/);
27   if (match) {
28     return match[1];
29   }
30   return null;
31 }
32 function genAcceptKey(webSocketKey) {
33   return crypto.createHash('sha1').update(webSocketKey + wsGUID).digest('base64');
34 }
```

图 2.5 建立 WebSocket 连接

重新运行服务端之后，在浏览器执行如下代码：

```
var ws = new WebSocket("ws://localhost:7000");

ws.onopen = function() {
  console.info('connected');
};
```

在浏览器控制台看到打印出 connected 说明 WebSocket 连接已经创建成功。建立 WebSocket 通信连接需要经过请求、响应、建立连接这三个步骤^[32]，图 2.6 展示了

WebSocket 全双工通信模型。

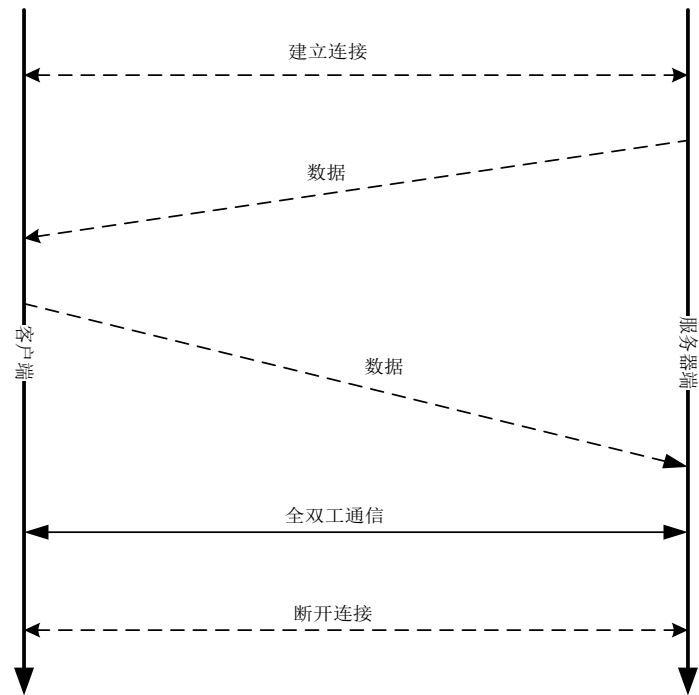


图 2.6 WebSocket 全双工通信模型

2.2 WebRTC 回声消除技术

2.2.1 回声消除原理

上一节在 WebRTC 整体框架中提到了语音引擎模块，它的主要处理过程是音频数据采集->编码->发送->接收->解码->播放。在此过程中，回声是影响通话质量的重要因素，本文将对声学回声消除作重点研究。

研究者们提出解决声学回声的方法主要分为三类：1. 对外部环境进行合理处理，使用强隔音或者吸音材料来搭建室内建筑，这样虽然能一定程度上减少回声干扰，但是花费成本高昂，降低了通信设备的灵活性，所以对大多数的用户还是不适应。2. 采用回声隔离器，它的实现原理是交替使用音频和视频信号隔离器，回声抑制效果明显且成本低，但是配备有回声隔离器的通信设备导致两端通话者不能实现同时说话的现象，同时路径的切换会产生消波现象^[33]。3. 采用自适应回声消除器。由于它具有良好的回声抑制效果、实现原理简单和开发成本低等特点，而被广泛应用于各种通信设备的回声消除应用中，如典型的音视频会议系统场景。因此

本文将对自适应回声消除重点研究。

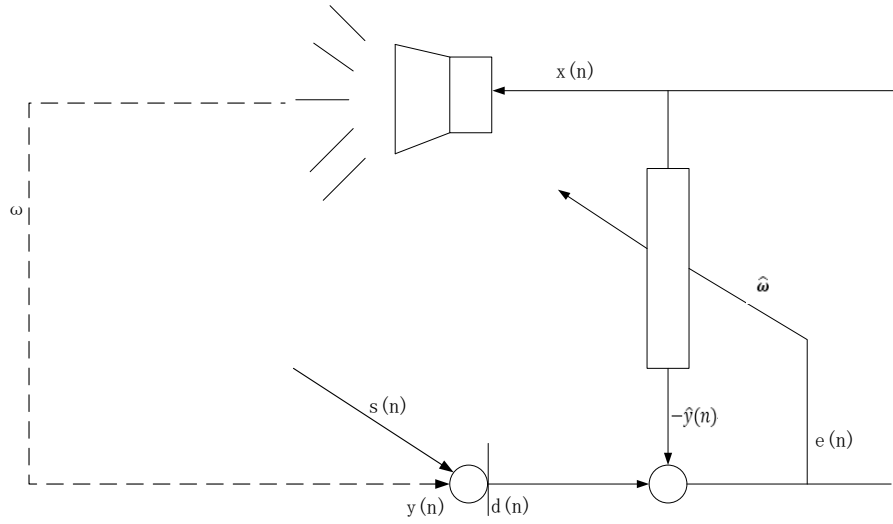


图 2.7 回声消除原理图

如图 2.7 是回声消除的原理图。 $x(n)$ 表示远端参考信号， $s(n)$ 是近端说话者的语音信号， $y(n)$ 表示回声。回声消除基本原理是以远端参考信号和产生的多路回声的相关性为基础，建立远端信号模型，模拟回声路径 $\hat{\omega}$ ，通过自适应滤波器算法不断修改滤波器参数，使其冲击响应和真实的回声路径 ω 相逼近，得到回声估计值 $\hat{y}(n)$ 。然后将麦克风接收到的信号 $d(n)$ 减去回声估计值 $\hat{y}(n)$ 得到误差信号 $e(n)$ ，即可实现回声消除功能。

如下是图中相关变量的表达式：

$$y(n) = x(n) * \omega \quad (2.1)$$

$$d(n) = s(n) + y(n) \quad (2.2)$$

$$\hat{y}(n) = x(n) * \hat{\omega} \quad (2.3)$$

$$e(n) = d(n) - \hat{y}(n) \quad (2.4)$$

2.2.2 自适应滤波器

自适应滤波器一般分为线性滤波器和非线性滤波器两种。如果滤波器输出端的量与它输入端的量是线性函数，则认定该滤波器是线性的；反之，则是非线性的。本文介绍的自适应滤波器是指输入信号进行过滤的同时，使用自适应算法不断修正滤波器系数，让滤波器输出量和期望量之间的误差越来越小，使它们越来越逼近

回声信道，最终收敛于最优。因此自适应滤波器实质上就是一种能调节自身传输特性以达到最优的维纳滤波器^[34]。

线性自适应滤波架构可以分为有限脉冲响应(Finite Impulse Response, FIR)自适应滤波器和无限脉冲响应(Infinite Impulse Response, IIR)自适应滤波器^[9]。在自适应滤波器中，横向滤波器、脉动阵列滤波器和格型滤波器是 FIR 滤波器的三种结构^[35]。横向滤波器是 FIR 滤波架构中用得最多的。如图 2.8 是有限脉冲横向滤波器的结构图。下面对该滤波器的结构进行详细说明。

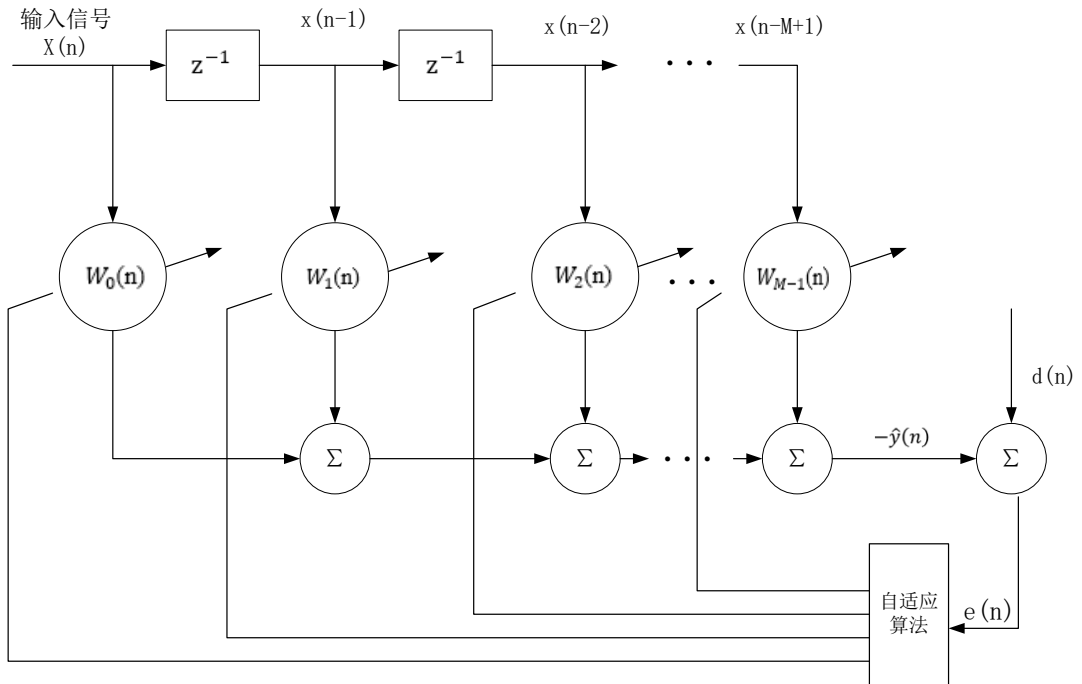


图 2.8 自适应滤波器结构图

记 $x(n)$ 为第 n 时刻的输入语音信号，将 M 时刻的语音信号输入向量表示为：

$$X(n) = [x(n), x(n-1), \dots, x(n-M+1)]^T \quad (2.5)$$

在系统中，回声路径 $H(n)$ 表示为：

$$H(n) = [h_0(n), h_1(n), h_2(n), \dots, h_{M-1}(n)]^T \quad (2.6)$$

式中 M 表示向量的长度。各个输入信号经过对应的回声路径，再通过加法器对各个乘法器输出求和，得到滤波器的输出信号为：

$$y(n) = H^T(n)X(n) \quad (2.7)$$

回声信号简化为输入信号 $X(n)$ 经过回声路径为 $H(n)$ 的 FIR 滤波器后，再受噪声 $v(n)$ 干扰，由麦克风拾取信号 $d(n)$ 为：

$$d(n) = y(n) + v(n) = H^T(n)X(n) + v(n) \quad (2.8)$$

为了估计回声，使用模拟回声信道 $W(n)$ ，记为：

$$W(n) = [w_0(n), w_1(n), w_2(n), \dots, w_{M-1}(n)]^T \quad (2.9)$$

式中 $w_k(n)$ 表示抽头系数。用模拟回声信道 $W(n)$ 估算回声信号 $\hat{y}(n)$ 表示为：

$$\hat{y}(n) = W^T(n)X(n) \quad (2.10)$$

由滤波器的模拟输出信号与接收到的实际信号 $d(n)$ 的差值得到误差信号 $e(n)$ 表示为式(2.4)。在有噪声的系统中，通过不断更新自适应滤波器系数，从而使误差信号 $e(n)$ 和噪声信号 $v(n)$ 的差值尽可能达到最小；在环境相对安静，噪声可以忽略不计的系统中，使误差信号 $e(n)$ 的值尽可能小，从而有效进行回声消除。

2.2.3 自适应滤波算法的性能指标

前面介绍了滤波器结构，自适应回声消除系统的核心处理技术就是自适应滤波算法，即自适应滤波算法性能的好坏决定了回声消除的效果。通常，衡量自适应回声消除算法性能有如下几个指标：

1. 收敛速度

回声消除系统进行滤波器参数调节，逐步逼近最优维也纳解或者达到稳定状态需要的迭代次数。它直观反映了回声消除系统工作效率，收敛速度越快，证明回声消除算法性能越佳，回声消除系统的工作效率就越高。

2. 稳态失调

对于自适应回声消除算法，收敛速度与稳态失调性能是难以平衡的^[36]，但在实际的应用中，需要找到一个平衡点来衡量这两大性能指标。

3. 跟踪性能

当回声信道发生变化时，采用回声消除算法再次使回声消除系统达到收敛状态所需要的时间。它与收敛速度表述的意义是不一样的，计算收敛速度的方法是截止至某个瞬时时刻，但计算跟踪能力是从某个时刻到另一时刻的时间段。针对同一个回声消除算法，算法的收敛速度越快并不意味着跟踪速度越快，它们之间没有必然的联系。

4. 计算复杂度

计算复杂度是设计回声消除算法时需要考虑的问题，虽然它不能直观反映自适应算法的工作性能，但是复杂度低回声消除算法更容易被研究者接受。因此在提升自适应回声消除算法性能的同时，应考虑少牺牲计算复杂度的途径来提升算法性能，算法复杂度越高，对采用的硬件设备的要求更大。

2.3 本章小结

本章是 WebRTC 相关技术知识介绍。主要分为两部分，首先主要对 WebRTC 整体框架进行分析，对 WebRTC 课堂会议系统能够采用的组件、开源 API 及信令协议进行详细介绍，为后续实现基于 WebRTC 回声消除的课堂会议系统提供了理论基础。第二部分是对 WebRTC 回声消除技术的理论知识进行研究，特别是对自适应滤波器的内部结构作重点介绍，为第 3 章回声消除算法奠定理论基础。

第 3 章 WebRTC 自适应回声消除算法分析及改进

自适应滤波是数字信号领域一个重要课题，应用领域包括有源噪声控制^[37]、反馈消去^[38]、源定位^[39]和分离^[40]等。此外，自适应滤波算法已广泛应用于系统识别，应用包括声回波抵消和网络回波抵消。在这类应用中，自适应算法通过输入和输出信号来估计和跟踪未知系统^[34]。到目前为止，两种最流行的自适应滤波算法是最小均方和归一化 LMS 算法^[41]。

3.1 自适应回声消除模块

声学回声消除模块是解决声学回声问题常用的方法。通过前面两章研究 WebRTC 的基本原理，发现 WebRTC 的回声消除算法 AEC 主要包括三个重要模块，分别是回声延迟估计模块、线性自适应滤波器模块以及非线性处理(NonLinear Processing, NLP)模块^[42]。如图 3.1 是 WebRTC 回声消除模块框架图。

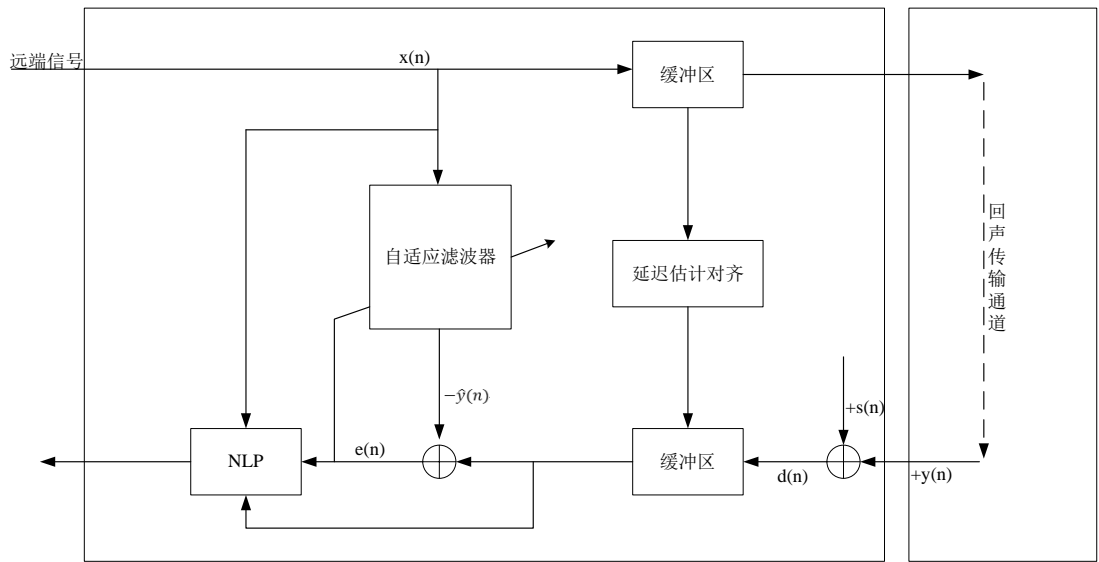


图 3.1 WebRTC 回声消除模块框架图

1. 回声延迟估计

在 AEC 处理前，需确保多路信号延时同步，例如近端信号中包含的回声信号和模拟输入信号，否则无法进一步进行回声消除工作。由于系统差异性，各个终端用户的延时是不一样的，如手机用户和 PC 用户，有的系统延时相对固定，有的则

是变化的，因此回声延时估计模块是 AEC 的组成部分之一。但本文不对此模块进行重点讨论。

2. 线性自适应滤波器

缓冲区将处理的语音信号进行延迟估计对齐后，接下来回声消除器中的线性自适应滤波器就可以进一步处理包含对应回声的近端语音信号。WebRTC 自适应算法通过在频域自适应滤波器上对此信号进行分块处理，得到估计的回声信号。本章将对自适应算法进行重点介绍。

3. 非线性处理

随着自适应线性声回波的发散性，在双端说话过程中会产生非线性声回波。由于近端语音与残余的非线性回声混合在一起，对非线性回声的抑制降低了近端语音的质量，有时还会导致部分语音失真。传统的线性 AEC 算法难以消除非线性回声，因此一些学者开始研究基于非线性模型的 AEC 算法。文献[43]和文献[44]中提到在频域使用非线性函数的方法计算不同频率下的抑制系数，实现对非线性回波抵消的精确调整。文献[45]提出了一种变步长分块频域自适应滤波(VSS-PBFDAP)方案来消除免提语音通信设备的声回波。具体来讲是为了减小双向通话过程期间的非线性回波，引入一组针对不同频率的不同步长，根据通话双方的发散程度来调整收敛速度。然后采用频域非线性回波处理(frequency-domain nonlinear echo processing, FNLP)抑制残余的非线性回波，保证近端语音的质量。文献[46]中提到使用二阶 Volterra 级数展开式建模的非线性均衡器设计。在实时性方面，基于频谱校正的非线性声回波抑制算法以其计算复杂度低、效果好等优点被广泛应用。总的来说，非线性处理过程较为复杂，整个过程涉及丰富的信号处理理论知识，本文并未对此模块作重点分析及研究。

3.2 自适应回声消除常见算法介绍

线性自适应滤波器的应用场景之一就是采用自适应滤波算法改善回声消除效果。而自适应滤波算法对滤波器的性能有至关重要的影响，本节中将阐述几种常用的自适应回声消除算法。前面第二章介绍了 FIR 横向滤波器，下面将以该滤波器结构为基础，对回声消除算法作深入研究并提出改进。自适应算法的发展呈现出不断改进的状态，本章提出改进的 NLMS 算法也是建立在基本回声消除的自适应算

法基础之上的。本章所有算法实现效果均采用 Matlab(R2016a)平台进行验证,实验数据通过手机软件和 PC 网页音频获取工具获取。本章的每个实验均经过几十次实验得到的综合实验效果,且每次独立仿真次数即迭代次数均在 200 以上,实验数据量足够,实验效果明显。

3.2.1 LMS 算法与 RLS 算法

1. LMS 算法推导过程

LMS 算法是最早应用于回声消除中的算法,为自适应滤波的回声消除算法奠定了扎实的理论基础^[47, 48]。下面介绍 LMS 算法的原理。

由前面第二章自适应滤波器的原理介绍可知,误差信号 $e(n)$ 可以表示为系统期望信号 $d(n)$ 与滤波器输出信号的差值:

$$e(n) = d(n) - W^T(n)X(n) \quad (3.1)$$

对上式两端计算平均值,并求其数学期望,此时代价函数可由 $e(n)$ 的均方误差表示为:

$$\varepsilon(n) = E[d^2(n)] - 2E[d(n)W^T(n)X(n)] + E[W^T(n)X(n)X^T(n)W(n)] \quad (3.2)$$

将 $P = E[d(n)X(n)]$ 定义为 $(N+1) \times 1$ 维互相关矢量, $R = E[X(n)X^T(n)]$ 为 $(N+1) \times (N+1)$ 维自相关矩阵^[49],式(3.2)可进一步表示为:

$$\varepsilon(n) = E[d^2(n)] - 2W^T(n)P + W^T(n)RW(n) \quad (3.3)$$

由式(3.3)可得,此时的代价函数与抽头向量呈二次函数关系,类似于一个超抛物面,具有最小值,且使得该代价函数最小的权系数值就是最佳解。基于最速下降法,沿着最速下降方向连续调整权向量 $W(n)$,代价函数 $\varepsilon(n)$ 的梯度向量 $\nabla \varepsilon(n)$ 记为:

$$\nabla \varepsilon(n) = \left[\frac{\partial \varepsilon(n)}{\partial W_0}, \frac{\partial \varepsilon(n)}{\partial W_1}, \dots, \frac{\partial \varepsilon(n)}{\partial W_{N-1}} \right]^T = 2[RW(n) - P] \quad (3.4)$$

为了求得代价函数 $\varepsilon(n)$ 的最小值,则梯度向量 $\nabla \varepsilon(n)$ 所有的元素必须同时都等于 0,即满足 $\nabla \varepsilon(n) = 0$,得到自适应 FIR 滤波器的最优抽头向量(通常也叫做 Wiener 权系数向量)为:

$$W_{opt} = R^{-1}P \quad (3.5)$$

将式(3.5)最优抽头向量的值代入式(3.3)得到最小均方误差:

$$E[e^2(n)]_{\min} = E[d^2(n)] - p^T W_{opt} \quad (3.6)$$

利用式(3.5)和式(3.6)求最佳抽头向量的精确解需要知道 R 和 P 的先验统计知识，而且还需要对矩阵求逆等运算。1960 年，Widrow 和 Hoff 共同提出了一种在先验统计知识未知的情况下求解 W_{opt} 的近似值的方法，取误差信号平方的瞬时值代替误差信号的统计平均^[50]，即是 Widrow and Hoff LMS 算法。此时代价函数 $\varepsilon(n) \approx e^2(n)$ ，则代价函数 $e^2(n)$ 的梯度向量 $\hat{\nabla} \varepsilon(n)$ 可表示为：

$$\hat{\nabla} \varepsilon(n) = \frac{\partial e^2(n)}{\partial W(n)} = \left[2e(n) \frac{\partial e(n)}{\partial W_0}, 2e(n) \frac{\partial e(n)}{\partial W_1}, \dots, 2e(n) \frac{\partial e(n)}{\partial W_{N-1}} \right]^T = 2e(n)X(n) \quad (3.7)$$

基于最速下降算法，LMS 算法将自适应滤波器的抽头向量迭代方程表示为：

$$W(n+1) = W(n) + \mu \nabla \varepsilon(n) \quad (3.8)$$

式中， μ ——步长因子

由公式(3.7)和式(3.8)可推导出抽头向量权值系数：

$$W(n+1) = W(n) + 2\mu e(n)X(n) \quad (3.9)$$

当迭代次数不断增大时，抽头向量的数学期望值可收敛至 Wiener 解，需要满足的条件是对角阵 $(1 - 2\mu \sum R)$ 的所有对角线元素均为 1，即：

$$|1 - 2\mu \lambda_{\max}| < 1 \quad (3.10)$$

式中， λ_{\max} ——自相关矩阵 R 的最大特征值

2. LMS 算法实现

LMS 算法的实现步骤如表 3.1 所示。

表 3.1 LMS 算法步骤

步骤名称	操作内容
步骤一：初始化	令 $n=0$ ；抽头向量 $W(0)=0$ ；估计误差： $e(0)=d(0)$ ；输入向量： $X(n)=[x(0), 0, \dots, 0]^T$ 。
步骤二：迭代运算	令 $n=1, 2, \dots, N$ ，更新权值向量如公式 3.9；更新估计误差 $e(n)$ 。
步骤三：重复运算	令 $n=n+1$ ，重复步骤(2)。

LMS 算法实现效果展示如图 3.2。

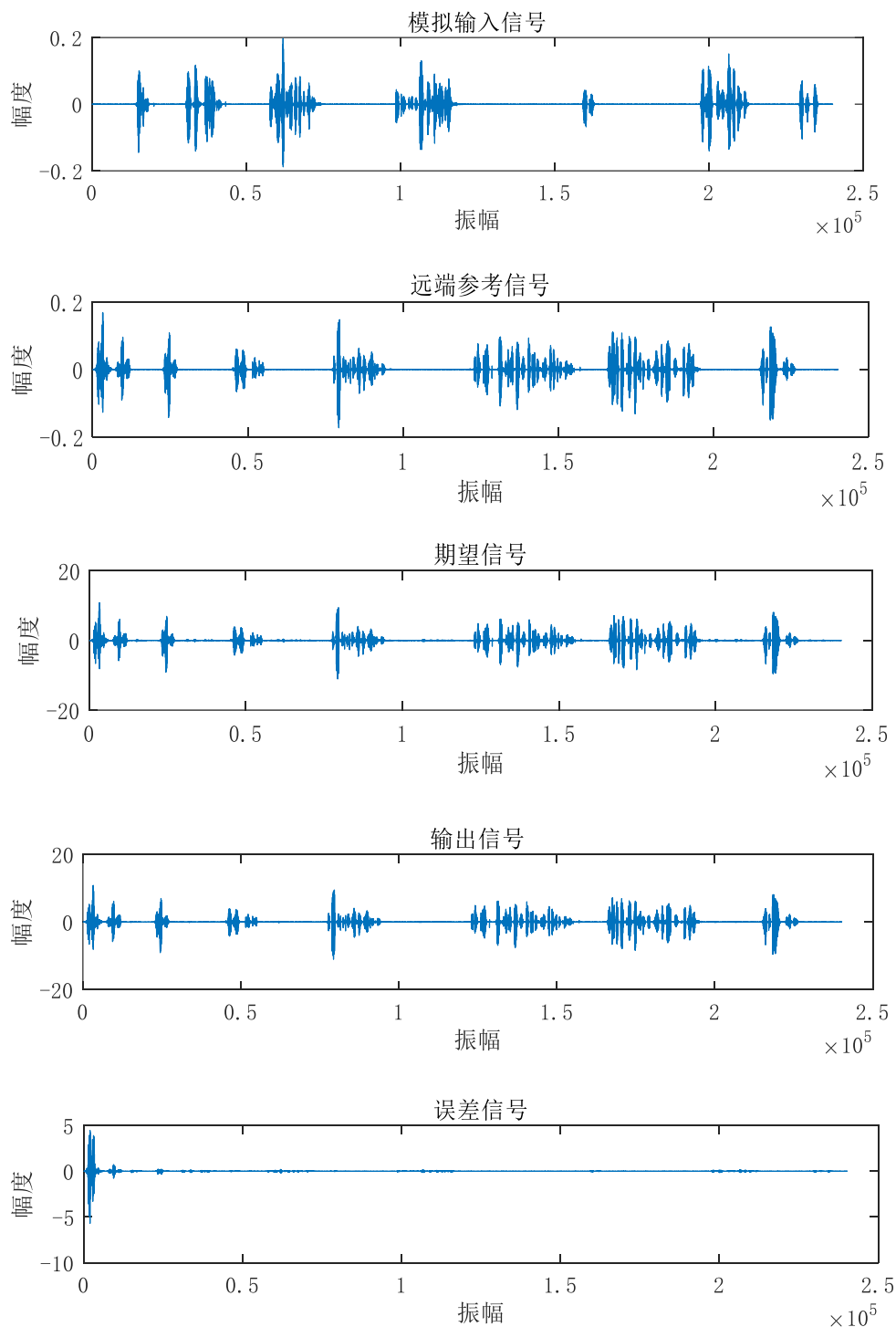


图 3.2 LMS 算法实现效果展示

图 3.2 第一子图是一个模拟输入信号，代表的是模拟人近端说话的语音信号；图 3.2 第二子图是远端参考信号，经过回声路径会生成一个回声信号；图 3.2 第三子图是滤波器系统的期望信号，是图 3.2 第一子图与图 3.2 第二子图产生的回声信号的混合信号。经使用 LMS 算法的自适应滤波器处理后，可以看到输出信号如图 3.2 第四子图，产生的误差信号如图 3.2 第五子图。输出信号与期望信号相比较，

有明显的回声消除效果；但也可以看到 LMS 算法前期的回声效果不是很理想，误差信号较为明显，即收敛速度有待提高。

3. RLS 算法推导过程

RLS 算法，又称作递归最小二乘算法，它利用已知 $n-1$ 时滤波器抽头系数，根据滤波器抽头系数更新准则，计算出 n 时刻的滤波器抽头权系数。该算法的核心思想是求出初始时刻到当前时刻的所有误差的平方和，并使其最小。此时的代价函数用指数加权的误差平方和表示为：

$$\varphi(n) = \sum \lambda^{n-i} |\varepsilon(i)|^2 \quad (3.11)$$

式中为保证过去某一段时间的观测数据被“遗忘”而导致的滤波器工作状态不稳定，引入平衡权重的遗忘因子 λ ，取值为(0,1)。这里定义误差估计为

$$\varepsilon(i) = d(i) - y(i) = d(i) - W^H(i)u(i) \quad (3.12)$$

这里通过使代价函数达到最小来保证滤波器的最佳性能。即联立式(3.11)和式(3.12)，可通过对权向量求导 $\frac{\partial \varphi(n)}{\partial W} = 0$ ，解得：

$$W(n) = r(n)R^{-1}(n) \quad (3.13)$$

式中

$$R(n) = \sum_{i=0}^n \lambda^{n-i} u(i)u^H(i) \quad (3.14)$$

$$r(n) = \sum_{i=0}^n \lambda^{n-i} u(i)d(i) \quad (3.15)$$

此时，最小二乘的解转化成 Wiener 滤波器的形式，下面根据时间相关性对自适应更新过程进一步说明，由公式(3.14)得到：

$$R(n) = \lambda \sum_{i=0}^{n-1} \lambda^{n-i-1} u(i)u^H(i) + u^H(n) * u(n) = \lambda * R(n-1) + u^H(n) * u(n) \quad (3.16)$$

令 $P(n) = R^{-1}(n)$ 为输入信号时间平均的逆相关矩阵，由矩阵求逆定理可得：

$$P(n) = \lambda^{-1} [P(n-1) - k(n)u^H(n)P(n-1)] \quad (3.17)$$

式中 $k(n)$ 为增益向量,表示为：

$$k(n) = \frac{P(n-1)u(n)}{\lambda + u^H(n)P(n-1)u(n)} \quad (3.18)$$

RLS 算法的抽头向量权值系数公式更新公式为：

$$W(n) = W(n-1) + k(n)e(n)X(n)$$

(3.19)

4. RLS 算法实现

RLS 算法的实现步骤如表 3.2 所示。

表 3.2 RLS 算法步骤

步骤名称	操作内容
步骤一：初始化	令 $W(0)=0$, $R(0) = \delta$ (δ 是小正数)。
步骤二：迭代运算	令 $n=1,2,\dots,N$ ，滤波： $y(n)=W^H(n-1)u(n)$ ；估计误差 $e(n)$ ；更新增益向量 $k(n)$ 如公式 3.18；更新权向量如公式 3.19；更新 $P(n)$ 如公式 3.17。
步骤三：重复运算	令 $n=n+1$ ，重复步骤(2)。

RLS 算法实现效果展示如图 3.3。

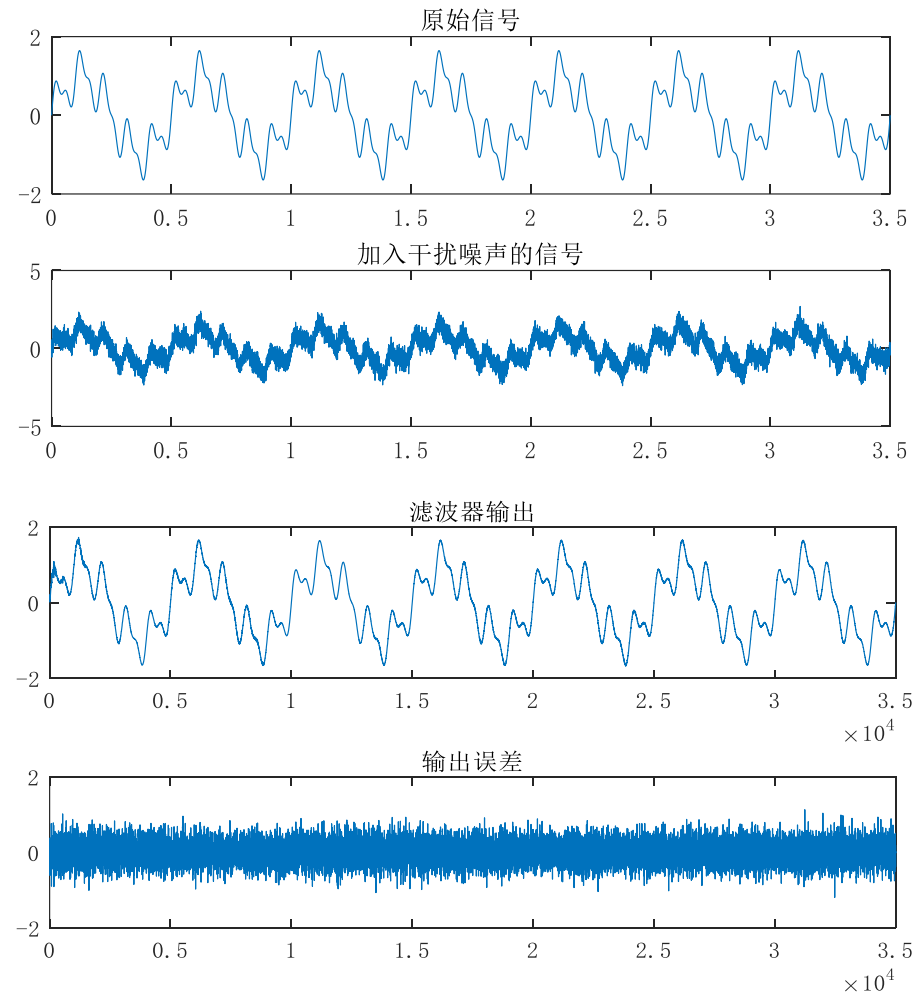


图 3.3 RLS 算法实现效果展示

图 3.3 第一子图是原始信号，即模拟输入信号记为 $signal$ 。当加入回声之后，本章算法实现加入的是理想状态下的高斯白噪声记为 $noise$ ，其中 $signoise = signal + 2noise$ 用来表示回声，图 3.3 第二子图表示期望信号，即原始信号混合回声后的信号。经采用 RLS 算法的自适应滤波器处理后，滤波器的输出信号如图 3.3 第三子图，误差信号如图 3.3 第四子图。可以明显看出经采用 RLS 算法的滤波器处理之后，误差信号较小且稳定。

5. LMS 算法与 RLS 算法对比

本节前面部分对回声消除的两大类算法，LMS 算法和 RLS 算法的理论研究及实验研究都分别进行了详细的阐述，接下来对两大算法之间的差异进行分析。

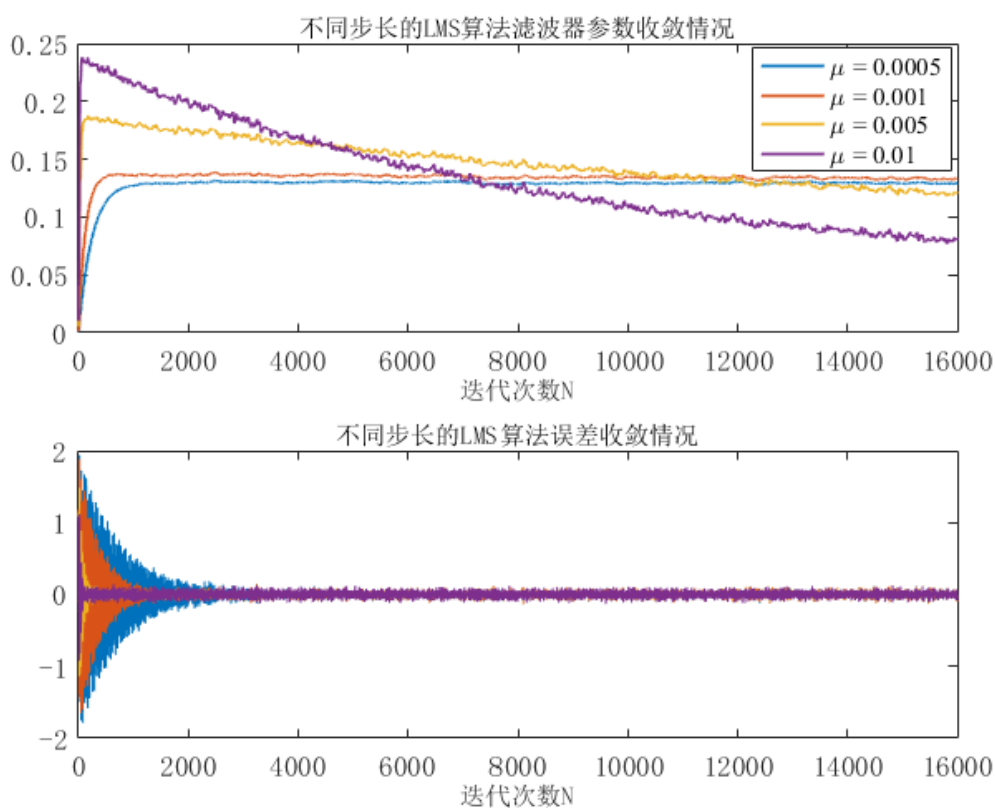


图 3.4 不同步长对 LMS 算法的影响

图 3.4 是不同步长 μ 对 LMS 算法的影响。本次实验中，取滤波器阶数 $M = 30$ ，如图 3.4 第二子图，随着 LMS 算法步长因子 μ 的增大， $\mu = 0.0005, 0.001, 0.005, 0.01$ ，极大加快算法的收敛速度。但如图 3.4 第一子图，LMS 算法滤波器参数收敛速度也受到影响，随着步长因子 μ 增大，LMS 算法滤波器参数收敛情况越慢，非平稳信号适应性差。

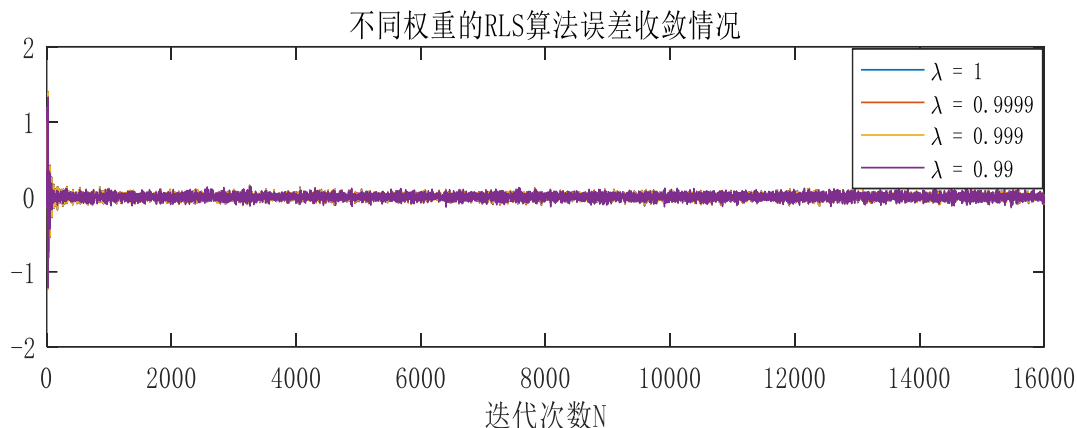


图 3.5 不同遗忘因子对 RLS 算法的影响

图 3.5 是不同遗忘因子 λ 对 RLS 算法的影响。本次实验中，取滤波器阶数 $M=30$ ， λ 的取值分别为 0.99，0.999，0.9999，1，在一定范围内对滤波器收敛情况影响不明显，总体收敛速度快。这里由于采样点数 $N=16000$ 过大，采用不同遗忘因子前期产生的算法收敛速度效果展示不明显，这里单独进行实验。如图 3.6，“遗忘”因子 λ 与增益向量 $k(n)$ 呈负相关关系， λ 取值越小，增益向量 $k(n)$ 越大，RLS 算法滤波器参数收敛性降低。

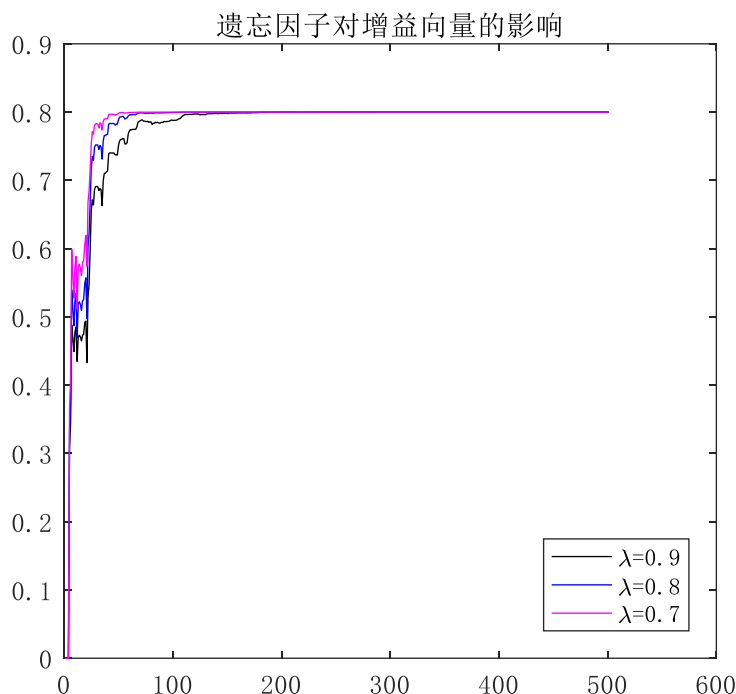


图 3.6 不同遗忘因子对 RLS 算法增益向量的影响

通过图 3.4 与图 3.5 比较，可以得出 RLS 算法的收敛速度明显优于 LMS 算法，但总体上看，不同步长及不同遗忘因子都会分别对 LMS 算法及 RLS 算法的滤波

器参数情况收敛情况及整体算法的收敛情况产生一定影响。如表 3.3 是 LMS 算法与 RLS 算法的一些性能比较。

表 3.3 LMS 算法与 RLS 算法对比

性能指标	LMS 算法	RLS 算法
收敛速度	慢	快
误差计算	后验误差	先验误差
非平稳信号适应性（稳态误差）	弱	强
原理	简单	复杂
易实现性	简单	复杂
计算复杂度	低	高

RLS 算法在自适应滤波器抽头向量的更新方程中，相较 LMS 算法，引入了与信号时间平均的逆相关矩阵，又称为“迭代矩阵 $P(n)$ ”和“遗忘”因子 λ ，增加了算法的计算量，提升了算法的复杂度。LMS 算法因其原理简单、复杂度低及易于实现的特点在回声消除领域广泛使用。

3.2.2 NLMS 算法

1. NLMS 算法推导过程

上节中提到 LMS 算法具有简单易实现的优点，但收敛速度与稳态误差难以调和的矛盾成为传统 LMS 算法应用瓶颈之一。针对 LMS 算法的固有缺陷，不少研究者基于此不足对其进行改进，如采用变步长或者步长矩阵调整步长大小提升算法稳定性^[51-53]。具有代表性的方法是通过自相关矩阵调节步长因子的大小^[22, 23]，具体过程如下：

用输入向量自相关矩阵 $R(n) = X(n)X^T(n)$ 的倒数去更新常数步长因子，NLMS 算法的抽头向量权值系数公式更新为：

$$W(n+1) = W(n) + \frac{1}{X(n)X^T(n)} e(n)X(n) \quad (3.20)$$

在实际应用中，为避免输入信号 $X(n)$ 的内积过小，即 $R(n)$ 太小或趋近于 0 而引起的算法稳定性差，式(3.20)分母中加一个较小的常数，即引入正则化参数 δ ；式(3.20)需要乘以一个常数 $\eta(0 < \eta < 2)$ 来平衡收敛速度和稳态误差。修正后的 NLMS 抽头向量权值系数公式为：

$$W(n+1) = W(n) + \frac{\eta}{\delta + X(n)X^T(n)} e(n)X(n) \quad (3.21)$$

2. NLMS 算法实现

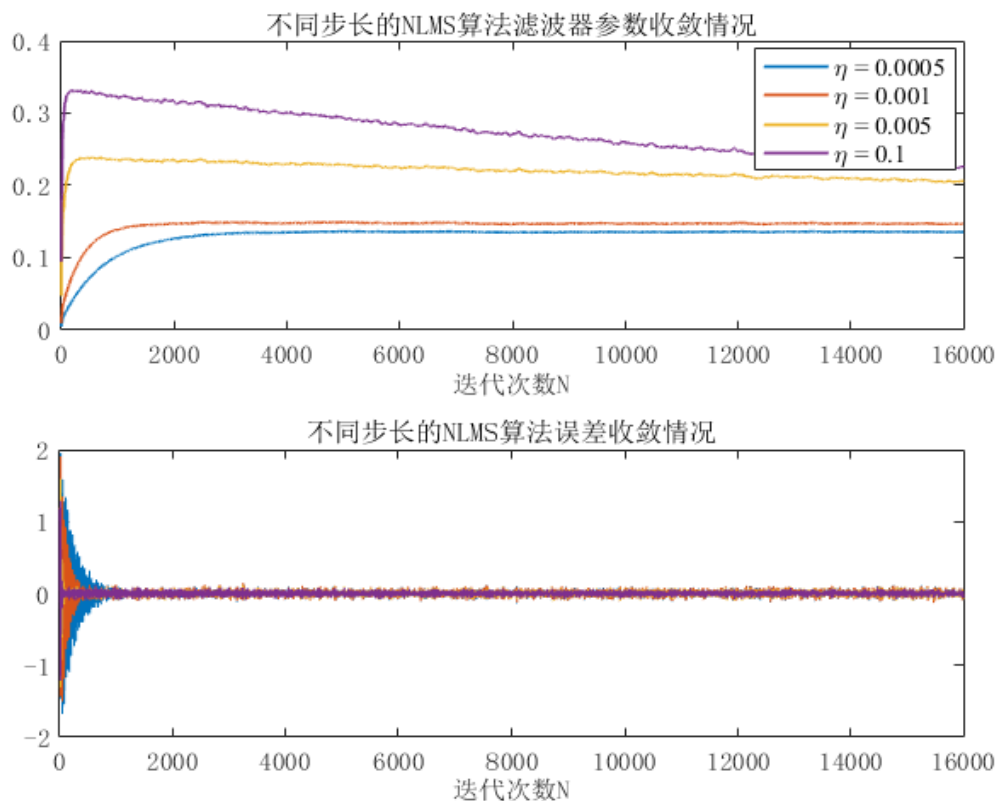


图 3.7 不同步长对 NLMS 算法的影响

NLMS 算法实现如图 3.7 所示。本次实验中取自适应滤波器阶数 $M=30$, $\delta=0.001$, 步长因子 η 取值为 0.0005, 0.001, 0.005, 0.01。由图 3.7 结果可知, 不同步长对 NLMS 算法有不同的影响, 如图 3.7 第二子图, 随着步长的增大, 收敛速度加快; 但稳定误差与收敛速度是一对矛盾, 如图 3.7 第一子图, 滤波器参数收敛情况变差。

3.3 改进的 NLMS 算法

3.3.1 理论研究

通过图 3.7 第二子图与图 3.4 第二子图比较, 即 NLMS 算法与 LMS 算法比较, 可以看出随着步长因子的不断增大, NLMS 算法的收敛速度加快, 且整体上要比 LMS 算法的收敛速度快。但步长因子取值越大, 滤波器参数收敛速度的情况下降。如图 3.4 第一子图与图 3.7 第一子图, 分别取 LMS 的步长因子 $\mu=0.01$, NLMS 算

法的步长因子 $\eta = 0.01$ 比较, LMS 算法前期的滤波器参数最大为 0.24, NLMS 算法前期的滤波器参数最大为 0.34, 经过滤波器的自适应更新, 后期滤波器参数达到稳定时的收敛速度 NLMS 算法比 LMS 算法快但收敛时的精度值也在增加。总体上 NLMS 算法在 LMS 算法的基础上提升了算法的收敛速度, 但增大了稳态误差, 本章对此不足提出改进。

本章提到的传统 NLMS 算法中抽头向量权值系数公式与一个固定的步长因子, 虽然可以针对回声环境进行更新, 即为了保证系统具有更快的收敛速度和跟踪速度, 可以选择更大的步长因子; 当算法接近收敛时, 滤波器权重接近最佳权重, 为了减少算法的稳态误差, 可以选择一个较小的步长因子。但是并没有相关文献对其进行合理的定义, 这样不能对步长因子进行精确的更新, 无法平衡收敛速度和稳定误差之间的效果。

在传统回声消除算法中, 假设系统背景噪声不存在, 将系统背景噪声忽略不计, 降低了算法的性能。本章考虑实际的回声消除环境中存在背景噪声信号 $v(n)$ 的影响, 并将 n 时刻和 $n-1$ 时刻的误差信号 $\varepsilon(n)$ 和表示 $e(n)$ 分别表示为

$$\varepsilon(n) = d(n) - W^T(n)X(n) + v(n) \quad (3.22)$$

$$e(n) = d(n) - W^T(n-1)X(n) + v(n) \quad (3.23)$$

文献[54]中提到误差信号的数学期望 $E\{\bullet\}$ 来表示系统背景噪声 $v(n)$ 的功率:

$$\gamma_v^2(n) = E\{\varepsilon^2(n)\} \quad (3.24)$$

此时输入信号 $X(n)$ 的自相关矩阵 $R(n) = X(n)X^T(n)$ 可表示为:

$$R(n) = L\gamma_x^2 = LE\{X^2(n)\} \quad (3.25)$$

其中 L 是抽头向量的长度, 且满足 $L \gg 1$, γ_x^2 表示输入信号的功率。将式(3.9)代入(3.22), 用式(3.23)消去 $W(n-1)$, 结合式(3.25)得到等式:

$$E\{\varepsilon^2(n)\} = \gamma_v^2(n) = \left[1 - \mu(n)L\gamma_x^2\right]^2 \gamma_e^2(n) \quad (3.26)$$

在式(3.26)中, $\gamma_e^2(n) = E\{e^2(n)\}$ 是 $n-1$ 时刻误差信号的功率, 进一步推导, 得到二次方程:

$$\mu^2(n) - \frac{2}{L\gamma_x^2} \mu(n) + \frac{1}{(L\gamma_x^2)^2} \left[1 - \frac{\gamma_v^2(n)}{\gamma_e^2(n)}\right] = 0 \quad (3.27)$$

此时是关于步长因子的二次函数，可推导出步长因子为：

$$\mu_{iml}(n) = \frac{1}{R(n)} \left[1 - \frac{\gamma_v(n)}{\gamma_e(n)} \right] \quad (3.28)$$

同样的为保证算法稳定性差，引入正则化参数 θ ，则此时抽头向量权值系数公式更新为：

$$W(n+1) = W(n) + \frac{1}{\theta + R(n)} \left[1 - \frac{\gamma_v(n)}{\gamma_e(n)} \right] e(n) X(n) \quad (3.29)$$

其中 $\gamma_v^2(n)$ 是背景噪声功率或者估计背景噪声功率，将误差信号功率进行归一化处理^[55]：

$$\gamma_e^2(n) = (1 - \lambda) \gamma_e^2(n-1) + \lambda |e(n)|^2 \quad (3.30)$$

式中 $\lambda = 1 - \frac{1}{KL}$ ($K \geq 2$)， L 为滤波器长度。

3.3.2 实验结果及分析

改进的 NLMS 算法的实现步骤如表 3.4 所示。

表 3.4 改进的 NLMS 算法步骤

步骤名称	操作内容
步骤一：初始化	令 $W(0)=0$, $\gamma_e^2(0)=0$ 。
步骤二：迭代运算	令 $n=1,2,\dots,N$ ，滤波： $y(n)=W^H(n-1)u(n)$ ；更新估计误差 $e(n)$ ；更新权值向量如公式 3.29；更新 $\gamma_e^2(n)$ 如公式 3.30。
步骤三：重复运算	令 $n=n+1$ ，重复步骤(2)。

本章在提出的算法，固定步长 LMS 算法和传统 NLMS 算法之间进行了性能仿真实验对比。实验条件如下：期望信号为输入信号、回声信号和标准高斯白噪声叠加的信号，并提取 1000 个点。为了平衡收敛速度和稳态误差，选择滤波器阶数 $M=8$ ，LMS 算法的步长为 $\mu=0.05$ ，在传统 NLMS 算法中取 $\delta=0.001$ ， $\eta=0.05$ ，在本节的算法中 $\theta=0.001$ ，在 Matlab 中进行实验，结果如图 3.8 所示。

在图 3.8 中，蓝色曲线是固定步长 LMS 算法，亮青色曲线是传统 NLMS 算法，红色曲线是本文改进的 NLMS 算法。通过仿真实验可以得出，本文改进的 NLMS

算法的收敛速度明显快于固定步长 LMS 算法和传统 NLMS 算法,且算法达到稳定状态的误差值低于传统 NLMS 算法,即改进后的 NLMS 算法在传统 NLMS 算法的基础上对稳态误差性能影响较小。进一步证明了本文改进的 NLMS 算法收敛速度更快,且能够更好的平衡收敛速度与稳态误差之间的矛盾。

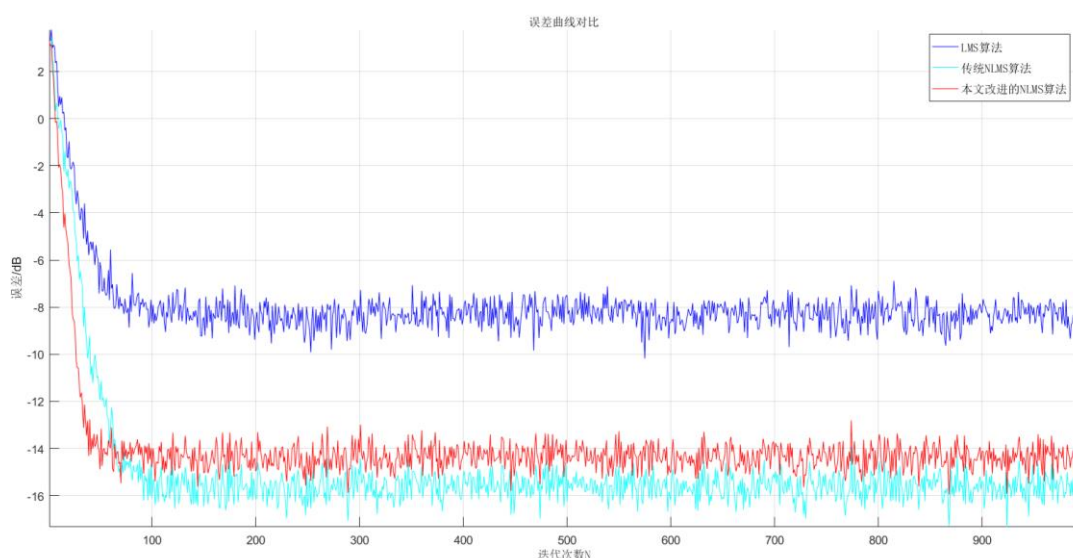


图 3.8 误差曲线对比

另外,在一定条件下,验证算法的跟踪性能,即在第1500个采样点时刻未知系统发生时变,对比本文改进的 NLMS 算法与传统 NLMS 算法和 LMS 算法的跟踪能力。为得出每一条曲线,分别做200次以上独立的仿真,然后求出其统计平均,得出学习曲线。本次实验取滤波器系数阶数 $M=2$,采样点数为3000。

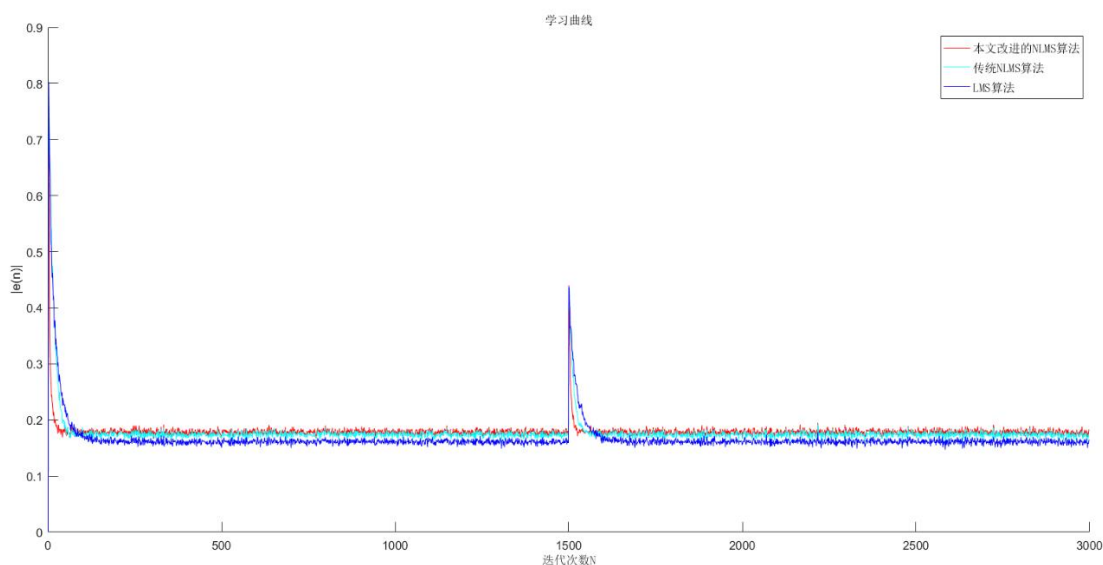


图 3.9 跟踪性能学习曲线对比

如图 3.9 所示，蓝色曲线是 LMS 算法的跟踪能力，取固定步长 $\mu=0.05$ ；亮青色曲线是传统 NLMS 算法的跟踪能力，取 $\delta=0.01$ ， $\eta=0.01$ ；红色曲线是本文改进的 NLMS 算法的跟踪能力，取 $\theta=0.001$ 。当 $n=1500$ 时，系统发生突变，在本文的实验条件下，本文改进的 NLMS 算法跟踪效果最佳。

本章在 LMS 算法基础上介绍了传统的 NLMS 算法，将相应的理论部分及核心算法步骤进行阐述，基于变步长思想和综合考虑背景噪声影响因素提出本文改进的 NLMS 算法。根据仿真实验结果验证本文提出的改进算法在获得相对较小的稳态误差时，其收敛速度效果也优于传统 NLMS 算法和 LMS 算法，并且当未知系统发生时变时，本文提出的改进算法也拥有较好的跟踪能力。

3.4 本章小结

本章首先对 WebRTC 回声消除模块原理进行介绍。针对回声延迟模块和非线性处理模块进行简要介绍，重点聚焦在自适应滤波算法研究。与 RLS 算法对比后发现，LMS 算法因具有固定步长，对信号变化调节能力不足的缺点。因此，本章在此基础上进一步研究 NLMS 算法并进行改进，并通过 Matlab 工具对其进行了仿真。通过实验和对比表明，考虑背景噪声影响且对步长因子给出更好的更新规则的改进算法可以有效地提升自适应滤波器性能，加快收敛速度，比 LMS 算法和 NLMS 算法具有更好的收敛效果和跟踪效果。

第 4 章 基于 WebRTC 回声消除的课堂会议系统设计与实现

前面三章介绍了 WebRTC 的核心技术，对 WebRTC 回声消除重点研究。本章在前文理论基础研究上，将改进后的回声消除算法运用在 WebRTC 底层的语音引擎模块中，同时结合课堂会议系统项目需求，设计基于 WebRTC 回声消除的音频系统。整个系统包含两个部分，分别是客户端设计和服务器端设计。客户端的设计是基于 Android 平台实现的，主要包括用户注册登录模块、音视频课堂会议模块、课堂交流讨论模块、课程管理模块及互动式课堂签到模块。服务端的设计包含 NAT 模块、信令模块和即时消息模块。系统结构原理图如图 4.1。

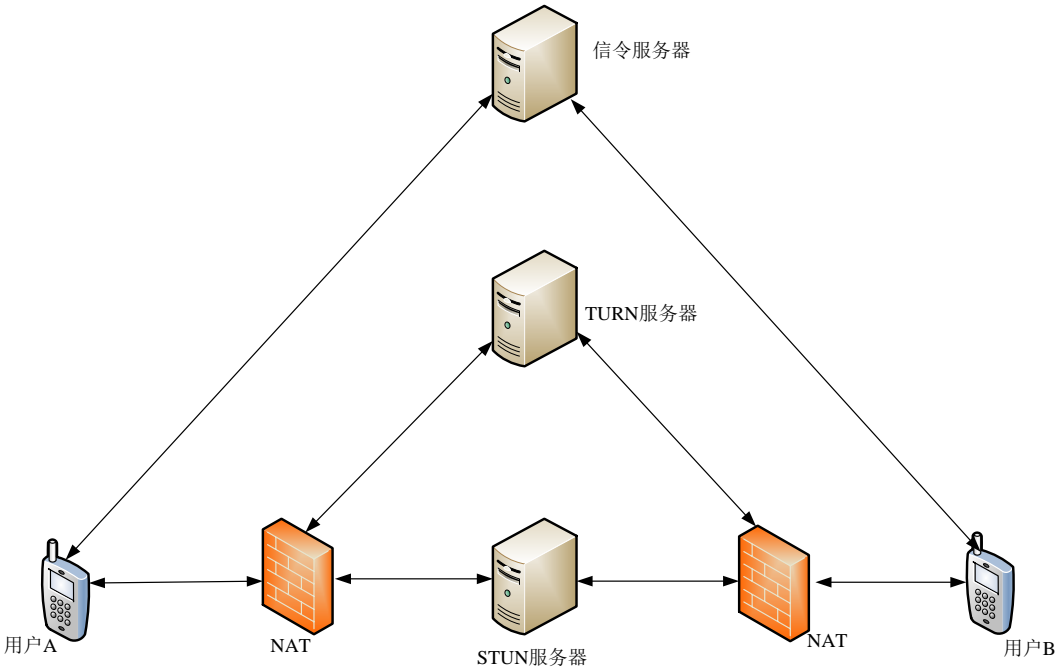


图 4.1 系统结构原理图

4.1 需求分析

需求分析是软件生命周期中相当重要的一个阶段，它可以确定待开发软件的功能、性能、数据、界面等要求，奠定了系统开发的基础。本节主要从功能性和非功能性两方面来介绍本系统的需求分析。

4.1.1 功能性需求分析

随着互联网的快速发展，用户对音视频交互系统的要求越来越高，在针对性方

面更加聚焦。音视频会议系统作为现如今推动社会发展的一大重要沟通工具，得到了广大用户的认可及支持。音视频会议系统是将音视频多媒体信息进行采集，经网络实时传输至接收端并进行页面渲染的交互系统。传统大型的音视频会议系统除需要有专有的硬件设备支持外，还需要建立独立的网络通道并规定对等终端之间的通信协议，再经 MCU(Multi Control Unit)的中转处理^[56]，形成专有系统^[57]。但这样会耗费巨大的网络资源，尤其在一些需求量小，用户数量不多的情况下。因此对音视频会议系统进行深入研究具有重大意义，如越来越多的对老年人专用实时通信、教育相关领域都进行了深入探究^[58-60]。

本章设计的基于 Android 端的课堂会议系统以 WebRTC 框架为主要技术背景，属于轻量级音视频会议系统，主要应用于云课堂交流、教学等场景。

本系统方案中的功能测试主要包括五个模块：用户注册登录模块、音视频课堂会议模块、课堂交流讨论模块、课程管理模块及互动式课堂签到模块。接下来对这五个模块的功能性需求进行分析。

1. 用户注册登录模块功能性需求

用户注册、登录是一个系统最基本的功能。本系统的所有功能及信息都是在用户已经注册且登录的前提下进行的。新用户使用本系统时，进行注册，需要填写姓名，手机号码，学生号或者工号（面向教师），所属学校名称，验证码以及密码（需要再次确认）等信息，需要注意一个手机号只能注册一个账号，注册完毕之后可使用已注册的手机号进行登录，进入系统。如果是已经参与注册的用户，之后登录不再进行注册，直接进行登录即可。

2. 音视频课堂会议模块功能性需求

一般情况下，已经注册的用户中并且在已经登录系统的情况下，无论是教师还是学生都可以主动创建会议，其他用户可根据会议名称以及会议 ID 进入。用户本身也可以直接进入其他已经创建的会议室中，进行课堂互动。

3. 课堂交流讨论模块功能性需求

该模块主要分为聊天室多人讨论及用户一对一讨论两个模块。课后教师与学生可以自由创建聊天室进行聊天，从而优化教学模式，增进师生交流讨论。具体来讲，教师与学生都可以主动创建聊天室，其他已经登录的用户可主动加入该聊天室。聊天室中，用户可以发送表情、文字等信息来进行消息互动。通过手机号添加联系

人到自己的通讯录，添加成功后，可直接点击通讯录中的好友用户名来选择进行音、视频聊天或者是发送消息的功能进行下一步的操作，也可以对联系人进行删除管理。

4. 课程管理模块功能性需求

为加强课程建设与课程管理，促进课堂内容，本章设计了课程管理模块。此模块主要分为学生用户和教师用户，学生用户拥有课程添加、课程查询及作业查看等功能。教师用户拥有课程添加、课程查询、作业发布及作业查看功能。

5. 互动式课堂签到模块功能性需求

传统的教学现场，都是采用教师点名，学生答到的方式进行课堂签到，该功能的设计目的是从节约课堂时间，改进课堂效率的角度出发，为推动教学质量做一点贡献。该模块主要分为学生用户和教师用户。由教师用户在手机端发布课程签到码，学生可在规定时间通过 APP 完成签到。针对教师用户还需要考虑到签到情况查看，这样有利于上课出勤率统计。为推动教师与学生的互动式课堂交流，还设计了随机点名的功能，促进课堂氛围。

4.1.2 非功能性需求分析

本系统的非功能性需求主要针对系统的可实施性及综合分析设计原则进行说明。主要表现在如下几个方面。

1. 可行性分析

(1) 技术方面

WebRTC 课堂会议系统采用 C/S 模式（用户下载安装 APP 后就可以进行使用），以 WebRTC 框架为核心，数据库采用 MySQL 进行管理，媒体服务器 Janus，防火墙打洞服务器 coturn 搭建在 Ubuntu 16.04 LTS Xenial（64 位）上，云服务器采用阿里云服务器。

(2) 实用性方面

本系统采用 WebRTC 开源框架为核心技术支持，一方面减少了系统的开发成本，在经济上减少了开发负担；另一方面，随着互联网的发展及课堂教学互动式教学的转变，课堂会议系统不仅成为了不少开发商的研究热点，课堂教学质量也不断得到提升，备受学生及教师的喜爱。

(3) 可操作性

本系统的操作界面简洁易操作，对于广大的使用者来说，不需要专人指导。

2. 系统的设计原则

(1) 网络适应性

1) 在同一视频会议系统中，各终端设备的配置不要求完全一致，只需符合相关标准即可。

2) 通过 SFU(Selective Forwarding Unit)架构数据流转发，可方便地避免不同网络服务商之间的网络瓶颈，保证不同地域、不同网络间的数据顺畅稳定。

3) 独特的编码处理，自动调节码流大小。音、视频数据在网络传输中自带网络纠错和数据包丢失机制。

(2) 冗余度和可扩展性

1) 系统具备容错机制，当某个用户发生意外连接情况时，保证其他用户的正常使用。

2) 系统的可扩展性这里主要是指软件的可扩展性，本系统设计的各个功能模块之间的耦合度低，利于本系统不断升级更新，使得软件的健壮性较高。

3. 系统安全性

(1) 首先 WebRTC 机制本身提供一些拦截非加密的媒体数据的功能。

比如 WebRTC 进行数据传输时采用的是安全实时传输 SRTP 协议，为实时传输的音视频数据提供加密、消息认证和完整性保护等功能。用户进入系统的音视频通信功能时，首先由应用程序询问是否允许访问摄像头和音频设备，得到用户的许可才能使用。

(2) 媒体服务器 Janus 的媒体数据传输层中包含有 DTLS (Datagram Transport Layer Security, 数据包传输层安全性) 协议和 SRTP 协议。

(3) 支持多种型号的手机，满足不同用户的需求。

4.1.3 WebRTC 回声消除接口分析

根据第二章 WebRTC 相关基础理论介绍，可通过调用 WebRTC 音频引擎中的开源 API 来分析回声消除的大致过程。WebRTC 回声消除实现过程如图 4.2 所示。

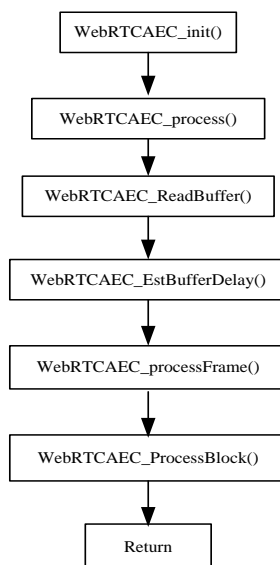


图 4.2 AEC 处理过程

`ReadBuffer()`用来读取缓存数据。`EstBufDelay()`函数用来实现回音延迟估计,本文第三章对回声延时估计估计模块有简要分析这里就不做过多解释。`ProcessFrame()`函数对信号按照实际应用的参数设置进行采样。最后 `ProcessBlock()`函数会根据 `ProcessFrame()`提供的数据进行回音消除的相关操作,里面包括语音状态检测,计算 NLMS 算法的迭代步长,更新自适应滤波器的抽头系数等操作。

4.2 总体设计

4.2.1 系统通信模型

WebRTC 采用的是 P2P 通信模型^[61],无需中心化节点,属于对等端通信。但是在两人以上通讯场景中,如果直接使用 P2P 连接,带宽和性能会一定程度上受到限制。对于基于 WebRTC 的多对多通信而言,有三种通信模型。

1. Mesh 架构

这里先说明本节中 Mesh、MCU 和 SFU 架构图中的四个终端都是用 A、B、C、D 表示,同时这里只是举四个终端的例子以图 4.3 为例,在实际的通信过程中,终端的个数可结合实际情况变动。如图 4.3, Mesh 架构由多个终端和一个 STUN/TURN 服务器组成,各终端之间两两互连,直接进行数据交互;每个终端与 STUN/TURN 服务器直接连接,实现 NAT 穿越,形成网状的拓扑结构。

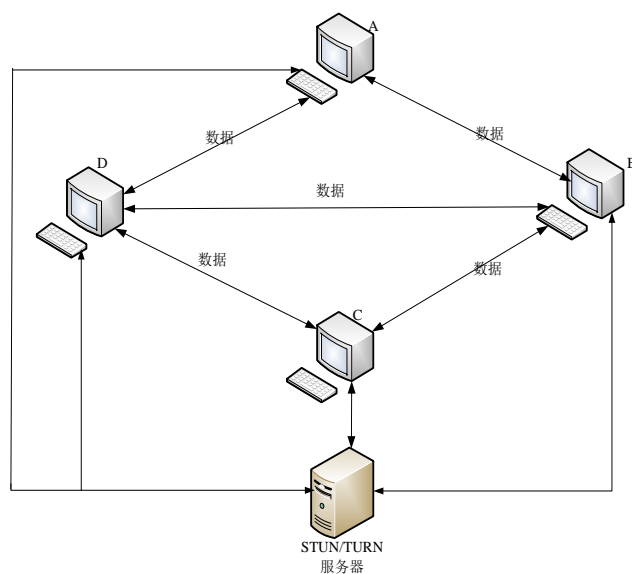


图 4.3 Mesh 架构图

Mesh 架构有一些显而易见的优点：一是这样不需要服务器对数据进行中转，具体来讲当某个终端想要共享它的音视频媒体流时，它会将共享的媒体流分别发送给其它 3 个终端，这样就实现了多人通信；二是充分利用了每个终端的带宽资源；三是节省了服务器成本。但是 Mesh 架构也有一些不足之处：一是多终端共享媒体流的时候，需要单独给每个终端都转发一份数据，需要极高的上行带宽，同时对终端的 CPU、Memory 等资源也是一个考验，因此这种通信架构一般不适合终端用户超过 4 个的情况。二是在多终端通信的过程中，若某个终端或者多个终端出现问题，不能实现 NAT 穿透的时候，这些终端就无法与其他终端直连，导致该终端无法与其他终端进行通信。因此该架构一般适用于小型系统开发，对于终端用户数量不少的情况下，该架构还需要做出更多的可靠性改进。

2. MCU 架构

如图 4.4，MCU 架构是由多个终端和一个 MCU 服务器组成的星型结构。MCU 架构的实现原理是 MCU 服务器将接收到的每个终端共享的音视频媒体流进行音视频混合，再将混合后的音视频媒体流发给各个终端，最终实现多人通信。MCU 架构目前技术非常成熟，主要用在硬件视频会议领域，消除了不同编解码设备的差异化。同时，MCU 架构将多路媒体流进行混合再转发，使多个终端接收到的画面是一致的，提升用户体验。但是，正因为 Mesh 架构的混和编码能力，会带来一定的延迟，同时对 CPU 资源的消耗很大，一般十几路视频就是上限了。

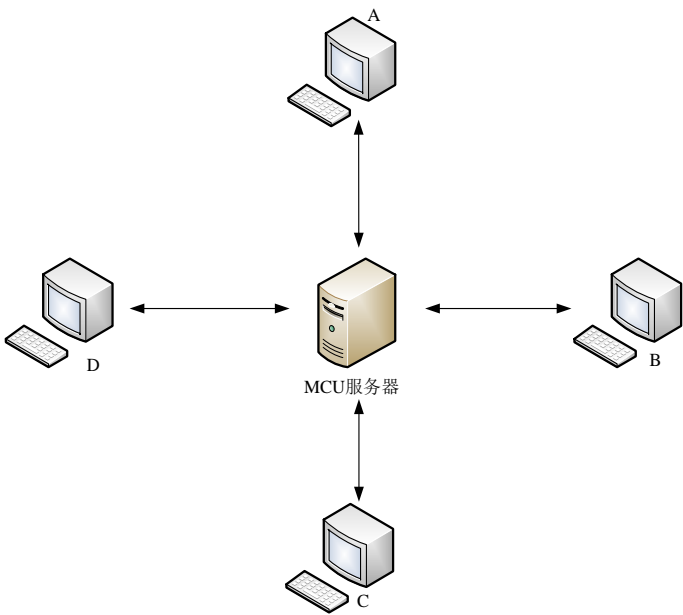


图 4.4 MCU 架构图

3. SFU 架构

如图 4.5，SFU 架构是由多个终端和一个 SFU 服务器组成。SFU 可以同时接受多个终端的媒体流，根据需要将接收到的音视频媒体流转发给其他终端。SFU 与 MCU 架构的不同之处就在于 SFU 不像 MCU 那样将接收到的音视频媒体流进行混合，而是直接将数据包转发，不需要编解码，CPU 资源消耗率较小，提高了实时性。

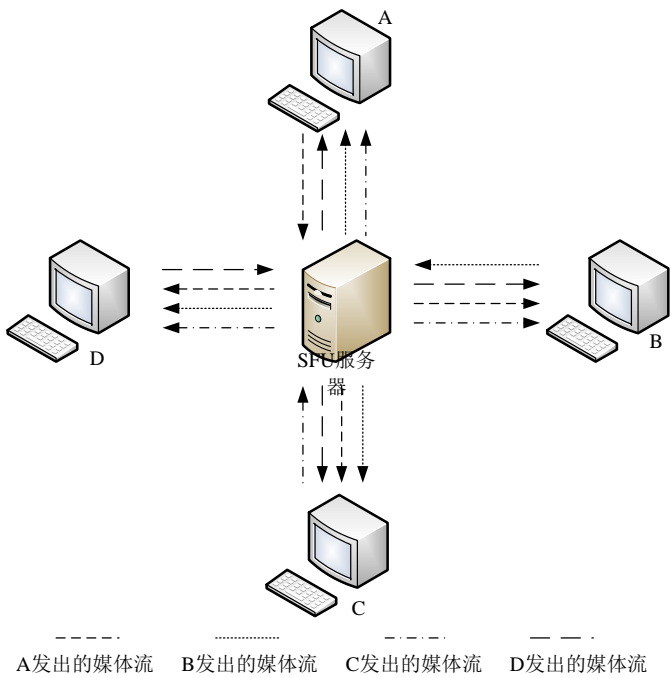


图 4.5 SFU 架构图

通过对比，综合三种架构的优劣特点，如果终端用户较多，Mesh 架构与 MCU 架构可能会导致多路视频窗口显示，渲染不同步、难处理的问题。SFU 架构无论是从资源消耗，还是网络状况、数据处理灵活性考虑，都有明显的优势，并且 SFU 架构也被大多数的研究者接纳。本文也采用 SFU 架构来设计实现基于 WebRTC 回声消除的课堂会议系统。

4.2.2 系统总体设计

本系统基于 ASSF(Access Service Standard Foundation)架构^[24, 62]，系统总体架构如图 4.6 所示。访问层为终端用户，如需要配置的硬件设备或者软件环境；功能层包括系统功能模块的设计；标准层为采用的标准架构及数据库系统，本系统采用 C/S 架构和 MySQL 数据库；基础层为本系统使用的核心技术。

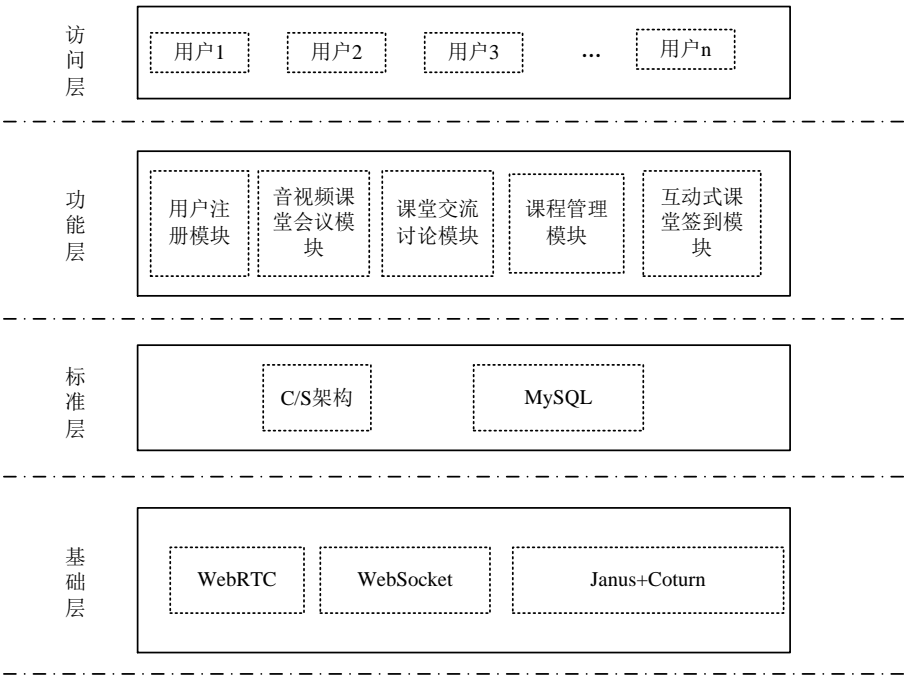


图 4.6 系统架构图

本系统的具体实现可分为两部分：服务端和客户端。由于本系统需要实现一对一以及多对多之间的音视频聊天，因此本系统的服务器端设计三部分来实现基于 WebRTC 的客户端之间的 P2P 通信：媒体服务器、信令服务器和 NAT 穿透服务器。媒体服务器的主要目的是在客户端之间转发媒体流。信令服务器负责信令的发送和响应。NAT 穿透服务器的目的在于穿越防火墙，找到用户在外网的 IP 地址及端口号。客户端为 Android 端，负责采集本地音视频媒体流和功能模块展示。系统

的整体功能模块如图 4.7 所示。

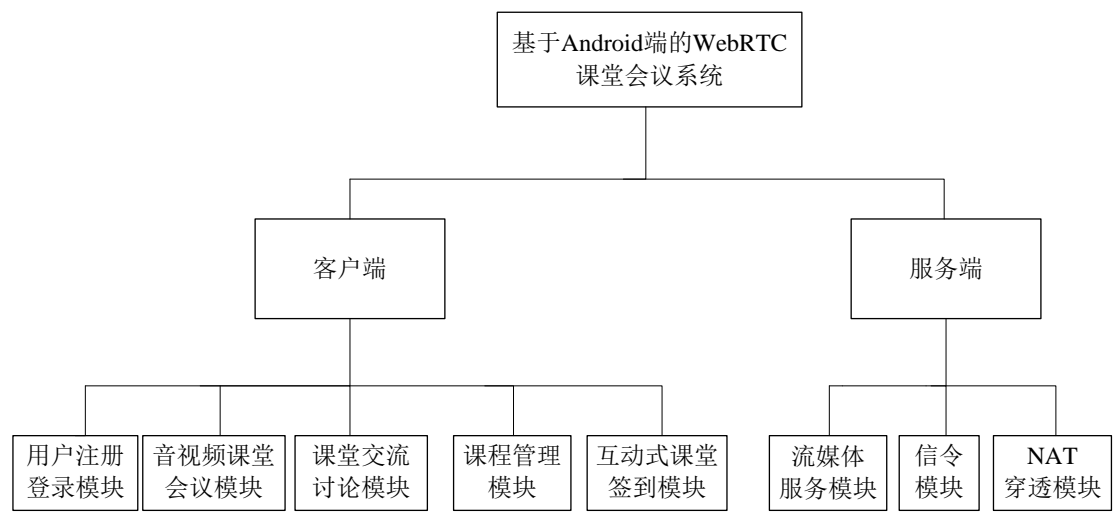


图 4.7 系统的整体功能模块

4.2.3 服务端设计与实现

本条主要对基于 Android 端的 WebRTC 课堂会议系统的服务端三大模块：媒体服务器、信令服务器和 NAT 穿透服务器作详细介绍。

1. 流媒体服务模块

在本章前面提到的 WebRTC 多对多通信模型中，对本系统所要采用的 SFU 通信架构已作详细介绍。这里不再描述。基于 SFU 通信架构，本文使用 Janus 作为 SFU 流媒体服务器进行系统环境搭建，其整体架构图如图 4.8 所示。

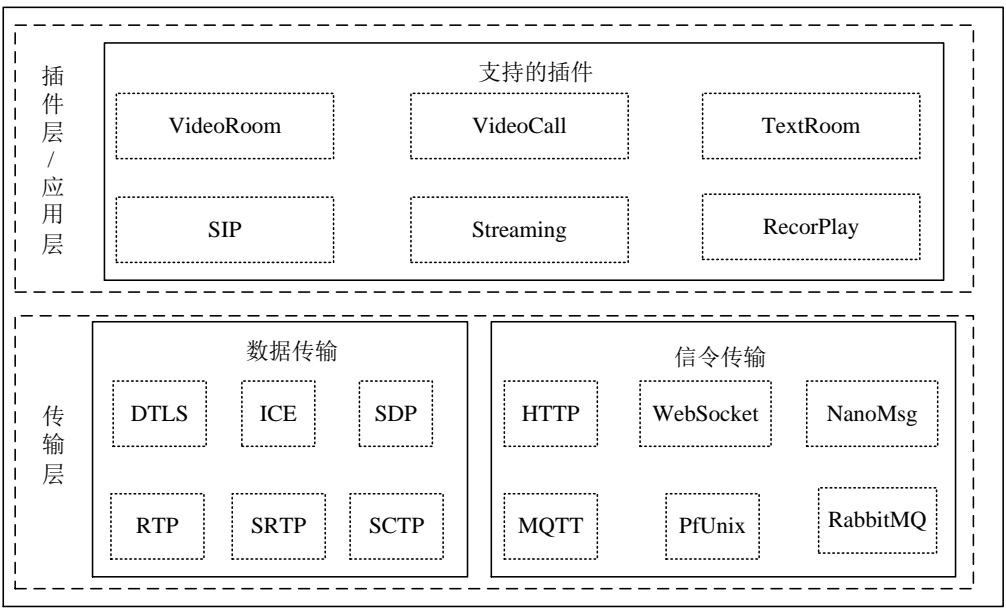


图 4.8 Janus 整体架构图

从其整体架构来看，Janus 分为两层，应用层和传输层。应用层又被称为插件层，每个应用都是一个插件，可以根据用户的需求动态选择某个应用进行加载。比如说 TextRoom 可以实现文本聊天室应用，VideoRoom 可以实现多人音视频通信，VideoCall 可以实现一个简单的视频呼叫应用等等。传输层包括媒体数据传输和信令传输，媒体数据传输层主要实现了 WebRTC 中需要有流媒体协议及其相关协议，支持的协议可以参照图 4.8 所示。信令传输层用于支持处理不同终端所需要的信令协议。用户可以根据自己的需求选择使用不同的信令协议。本系统采用的信令协议是 WebSocket 协议。

第一步：安装 janus 依赖库和工具，并设置环境变量

```
apt install git aptitude
```

```
export PKG_CONFIG_PATH=/usr/lib/pkgconfig
```

第二步：安装依赖库 libnice 库，下载该库之后操作如下。

```
cd libnice
```

```
./autogen.sh
```

```
./configure --prefix=/usr
```

```
make
```

```
make install
```

第三步：安装依赖库 libsrtp 库，下载该库之后操作如下。

```
tar xfv v2.0.0.tar.gz
```

```
cd libsrtp-2.0.0
```

```
./configure --prefix=/usr --enable-openssl
```

```
make shared_library
```

```
sudo make install
```

第四步：安装 websocket

```
git clone https://github.com/warmcat/libwebsockets.git
```

```
cd libwebsockets
```

```
mkdir build
```

```
cd build
```

```
make && sudo make install
```

第五步：安装主程序 janus-gateway，下载该包之后操作如下。

```
cd janus-gateway
```

```
sh autogen.sh
./configure --prefix=/usr/local/janus
make
make install
make configs
第六步：启动 janus
/usr/local/janus/bin/janus
```

2. 信令模块

传统的通信协议都是基于 HTTP 协议的，而它是遵从“请求-响应”的典型协议，并且每次发送请求时候都必须携带完整的头部信息、开销大和信息交换效率低。HTML5 标准中的 WebSocket 协议解决了这些问题，客户端在与 WebSocket 服务器建立了连接通道后，服务器就能主动向客户端发送消息。

关于 WebSocket 的基础介绍，在本文的第二章已详细介绍，这里不再赘述。本文采用基于 WebSocket 协议的 Google 服务器 Collider。具体详细的环境搭建过程如下：

```
第一步：安装 golang
sudo apt-get install golang
第二步：部署
mkdir -p $HOME/goWorkspace/src
go get collidermain
go install collidermain
```

3. NAT 穿透服务器

由本章的系统架构图 4.1 可知，如果有用户位于 NAT 之后，就需要 STUN（Session Traversal Utilities for NAT，NAT 会话穿越应用）/TURN（Traversal Using Relays around NAT，中继穿透网络）来提供穿透服务。本文使用开源的 coturn 服务器来实现 NAT 穿透功能，下面为 coturn 搭建在 Ubuntu 16.04 LTS Xenial(64 位)上的具体代码：

```
第一步：安装依赖
sudo su root #先切换到 root
apt-get install build-essential #（可选），如果后面的./configure 失败时，可
```

先安装 gcc。

```
apt-get install openssl libssl-dev make
tar -zxvf libevent-2.1.10-stable.tar.gz
cd libevent-2.1.10-stable
./configure
make & make install
apt-get install sqlite libsqlite3-dev
cd ~/
```

这里的 coturn 的用户信息默认是存储在 sqlite 中，但本文需要将其保存在 MySQL 中，需要改成 MySQL 相关的依赖项。具体操作步骤如下：

- (1) apt-get install mysql-server -y #可选（安装 MySQL）
- (2) apt-get install mysql-client -y
- (3) apt-get -y install libmysqlcppconn-dev libmysqlclient-dev libmysql++-dev

第二步：安装 coturn 源码并进行编译

```
wget https://github.com/coturn/coturn/archive/4.5.1.1.tar.gz
tar -zxvf 4.5.1.1.tar.gz
cd coturn-4.5.1.1
./configure
make & make install
```

第三步：创建用户

```
turnadmin -a -u cquptslz -p cquptslz.cnblogs.com -r cnblogs
#上面的命令是创建用户名和密码，可以根据实际情况修改。
```

第四步：修改相关配置信息

```
listening-port=3478 #监听端口
```

```
listening-device=eth0 #监听的网卡
```

```
external-ip=52.81.17.142 #公网 ip
```

```
user=cquptslz: cquptslz.cnblogs.com #用户名:密码
```

#硬件设备上有多块网卡时，注意将 listening-ip 与 listening-device 进行匹配设置，参考图 4.9。

```

docker0  Link encap:Ethernet  HWaddr 02:42:d6:6b:9f:92
          inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0     Link encap:Ethernet  HWaddr 02:ff:e6:4e:09:30
          inet addr:10.23.72.250  Bcast:10.23.255.255  Mask:255.255.0.0
          inet6 addr: fe80::ff:e6ff:fe4e:930/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9001  Metric:1
          RX packets:45167 errors:0 dropped:0 overruns:0 frame:0
          TX packets:35751 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:58238336 (58.2 MB)  TX bytes:3644022 (3.6 MB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

```

图 4.9 coturn 服务器配置图

第五步：开启并测试服务器

`turnserver -o -a -f -v -r cnblogs`

#可用命令校验服务器是否启动成功，如果有类似下面的输出，说明 3478 监听正常。

```

root@ip-10-23-72-250:~/coturn-4.5.1.1# lsof -i:3478
COMMAND  PID USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
turnserve 16935 root   16u  IPv4  22422      0t0  UDP ip-10-23-72-250.cn-north-1.compute.internal:3478
turnserve 16935 root   17u  IPv4  22423      0t0  UDP ip-10-23-72-250.cn-north-1.compute.internal:3478
turnserve 16935 root   31u  IPv4  22431      0t0  TCP ip-10-23-72-250.cn-north-1.compute.internal:3478 (LISTEN)
turnserve 16935 root   32u  IPv4  22432      0t0  TCP ip-10-23-72-250.cn-north-1.compute.internal:3478 (LISTEN)

```

图 4.10 coturn 服务器配置成功显示图

4.2.4 客户端设计与实现

1. 基于 Android 平台环境搭建

WebRTC 技术内部兼容 Android 系统，且 Google 对 WebRTC 安卓端的源码进行了开源，本系统设计选择在 Ubuntu 系统下安装和部署安卓平台的编译工具，从而获取安卓端的 WebRTC 源码。接下来介绍基于 Ubuntu 系统下的 WebRTC 安卓平台编译工具的安装、部署过程、源码的获取和编译过程。

(1) 源码下载

第一步:安装 depot_tools 工具包

下载 depot_tools 工具包,之后要把 depot_tools 目录添加到系统环境变量中:

```
export PATH=$PATH:/path/depot_tools
```

通过如下命令验证是否安装成功:

```
fetch --help
```

显示如下内容说明 depot_tools 配置好了:



```
glumes ~ fetch --help 8732 13:34:02
usage: fetch.py [options] <config> [--property=value [--property2=value2 ...]]

This script can be used to download the Chromium sources. See
http://www.chromium.org/developers/how-tos/get-the-code
for full usage instructions.

Valid options:
  -h, --help, help      Print this message.
  --nohooks              Don't run hooks after checkout.
  --force                (dangerous) Don't look for existing .gclient file.
  -n, --dry-run          Don't run commands, only print them.
  --no-history           Perform shallow clones, don't fetch the full git history.
```

图 4.11 depot_tools 工具配置成功显示

第二步:使用 depot_tools 去下载 WebRTC 源码

找到 WebRTC 源码地址进行下载,下载后执行如下命令:

```
mkdir webrtc
```

```
cd webrtc
```

```
fetch --nohooks webrtc_android
```

```
gclient sync
```

(2) 添加依赖

下载 WebRTC 后需要添加相关的依赖,进入到 WebRTC 源码的 src 目录中,执行如下命令:

```
cd src
```

```
./build/install-build-deps.sh
```

```
./build/install-build-deps-android.sh
```

(3) WebRTC 源码编译

通过如下命令进行编译:

```
./build/android/envsetup.h
```

```
gn gen out/release/armeabi-v7a --args='target_os="android" target_cpu="arm"
is_debug=false'
```

```
ninja -C out/release/armeabi-v7a
```

编译后会输出两个文件：

```
out/release/armeabi-v7a/lib.java/sdk/android/libwebrtc.jar
```

```
out/release/armeabi-v7a/libjingle_peerconnection_so.so
```

上面两个文件，一个是 jar 包和一个 so 动态库，这就是最终需要的编译结果文件。利用 WebRTC 进行开发就需要导入编译结果文件。

(4) Android 端环境搭建

除了编译 WebRTC 源码得到 so 动态库和 jar 包之外，WebRTC 还提供了 Maven 仓库供开发者下载，在 Android Studio 开发工具中导入编译结果文件，添加 Maven 仓库后即可完成开发环境搭建。

2. Android 端音视频流采集

在 Android Studio 开发工具中利用 WebRTC 技术进行安卓端系统设计的时候，涉及到音视频流的采集及传输处理，所以首先需要在配置文件中设置打开相应的权限，如摄像头、网络等权限。这个过程需要涉及调用一些 WebRTC 安卓端的开源类和方法，如表 4.1 所示。

表 4.1 Android 端实现音视频通信的主要类和方法

接口类/函数	说明
PeerConnectionFactory	创建连接以及创建在连接中传输采集的音视频流数据
VideoCapturerAndroid	负责选择摄像头
Audio/VideoSource	负责选择音视频源
Audio/VideoTrack	用于封装音视频源
MediaStream	用于封装音视频流
PeerConnection	表示两个对等端的链接
setLocalDescription()	将对象记录为本地描述
setRemoteDescription()	将对象记录为远程描述
CreateOffer()	用于创建请求信令，包含 SDP
CreateAnswer()	用于创建应答信令，包含 SDP
OnIceCandidate()	发送 ICE 候选项给远程对端
AddIceCandidate()	为 ICE 代理提供远程候选项
AddStream()	将媒体流加入 PeerConnection 对象中
onAddstream()	添加远程流

在采集音视频数据前先对核心类 PeerConnectionFactory 初始化，其代码如下：
PeerConnectionFactory.initializeAndroidGlobals(
context, //上下文，可自定义监听
initializeAudio, //是否初始化音频，布尔类型

```
initializeVideo, //是否初始化视频, 布尔类型
videoCodecHwAcceleration); //是否允许硬件加速, 布尔类型
```

```
PeerConnectionFactory peerConnectionFactory“new PeerConnectionFactory();
```

3. 客户端音视频通信

安卓客户端之间的通信核心技术是使用类 `PeerConnection`，该类的方法能够在对等用户之间建立 `WebRTC` 连接，具体指能够使两个客户端创建、保持和关闭连接。图 4.12 为用户 A 和用户 B 建立音视频通信的具体执行过程。

从图 4.12 可知，用户 A 向用户 B 发起呼叫请求的流程包含三个阶段。第一阶段：对等方与相关服务器交换有关启动信息，包括对等方通道和用户媒体的控制消息。第二阶段：基于 `SDP`（`Session Description Protocol`，会话描述协议）交换创建对等连接。第三阶段：建立对等连接，对等方直接交换消息。这里用户双方通过控制 `WebRTC API` 实现以上步骤。主要任务是建立对等连接，在建立对等连接之前，将执行本地和远程描述的交换，如音频和视频媒体信息。

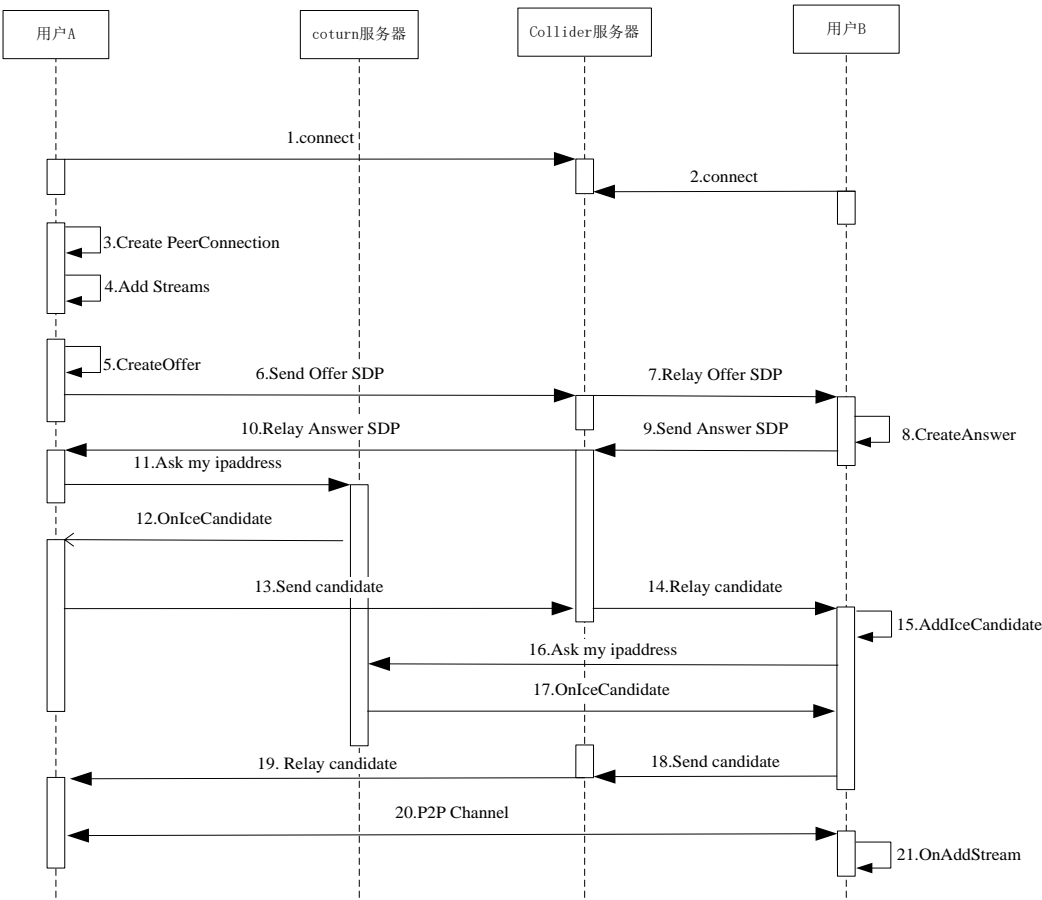


图 4.12 基于 WebRTC 的用户音视频通信呼叫过程

这里将描述如何使用 SDP 协议实现信令交换和提供应答。首先,用户 A 运行 `RTCPeerConnection createOffer()` 方法。之后,它使用 `setLocalDescription()` 方法设置本地描述,然后将该会话描述通过信令服务器发送到用户 B。然后,用户 B 使用 `setRemoteDescription()` 方法将用户 A 发送的描述设置为远程描述。用户 B 运行 `RTCPeerConnection createAnswer()` 方法。在 `createAnswer()` 方法中,它设置本地描述并将其发送给用户 A。当用户 A 获得用户 B 发出的会话描述时,它将使用 `setRemoteDescription()` 方法将其设置为远程描述。这样用户 A 和用户 B 就已经建立了音视频传输的 P2P 通道。

4.3 功能模块设计与实现

4.3.1 用户注册登录模块

用户注册登录模块的流程图如图 4.13 所示。

用户注册登录模块主要用来处理用户登录及注册,获得进入本系统设计 APP 的应用权限,明确用户之间的关系。新用户进行注册时,首先填写用户名、手机号、学生学号或者老师工号,然后再获取验证码。随即输入的用户名和手机号会被传入到数据库中,与已有的用户注册信息进行对比,如果数据库中没有该用户名,则此次注册为新用户注册过程,随即跳转至注册成功页面。反之则提示该用户已注册,需要重新输入手机号。

本模块是系统设计初期必不可少的功能模块,它规范了系统用户的身份,区分了用户的使用权限,整体上提升了系统的安全性。同时登录的用户能够有效的存储个人信息,方便用户下次使用。结合本系统的需求分析,此模块是实现其它模块的基础模块,其主要功能包括个人信息校验、验证码接收和数据库信息比对等。

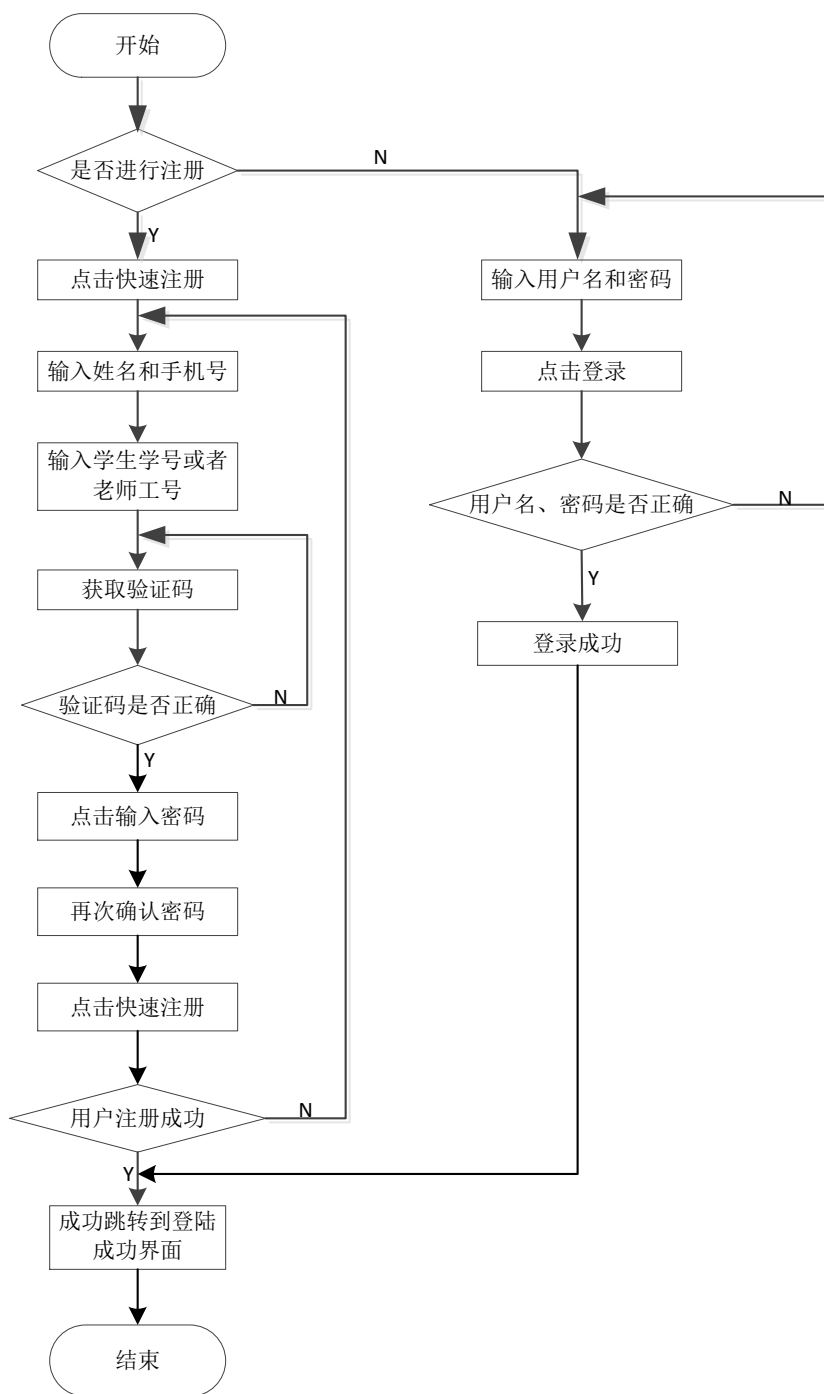


图 4.13 用户注册登录模块流程图

4.3.2 音视频课堂会议模块

在登录的状态下，进入到多人视频聊天的界面，可进行创建新的会议或者直接进入已经存在的会议室进行音视频通话。在进入之后，可进行麦克风及摄像头的设置管理，从而实时地根据课堂需求打开/关闭摄像头及麦克风，方便音视频课堂会议交流。会议结束或中途退出课堂会议可自行退出。

音视频课堂会议模块是整个设计系统的重点，它帮助我们获取本地与远程音视频流，并进行实时展示。设计过程中，考虑到用户对于多人音视频的需求，实现了多人音视频会议功能，与传统的一对一音视频会话形式更方便用户的交流，尤其适用于视频会议，例如本章的课堂教学应用方面。考虑到整体系统的开放性，针对已经注册并登录的用户，可以在本系统中根据自己的需求创建会议室并通知好友进入该会议室，避免了繁琐的会议室权限申请过程，也可以作为一对一的音视频课堂会议室供用户使用。音视频课堂会议模块的流程图如图 4.14 所示。

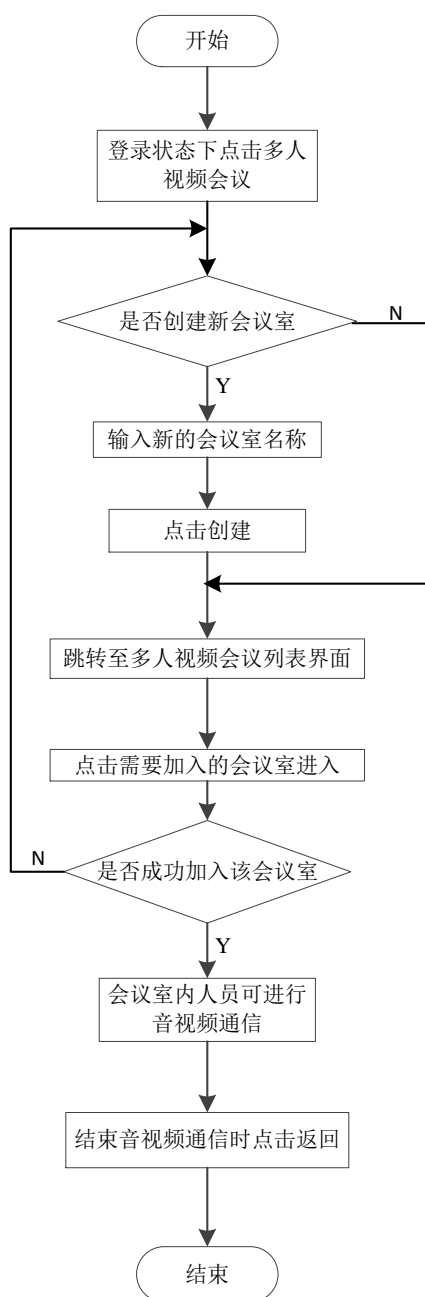


图 4.14 音视频课堂会议模块流程图

同时，音视频课堂会议模块作为本系统的核心功能模块之一且本模块中的语音引擎中采用第三章的回声消除改进算法，这里对回声消除的处理过程进行介绍，且对本文研究的滤波器系数更新条件进行说明，如图 4.15 所示。当麦克风接收到输入信号时，先进行远端语音状态检测，如果没有检测到远端语音信号，说明只存在近端语音信号，此时滤波器不进行滤波和权系数更新，直接将近端语音信号传输到远端。当检测存在远端语音信号时，再进行双讲语音检测，假如不存在双讲状态，说明只存在远端信号，此时滤波器既要进行滤波又要进行系数更新。假如存在双讲状态，此时滤波器暂停权系数更新，用当前时刻滤波器权系数继续滤波消除回声。

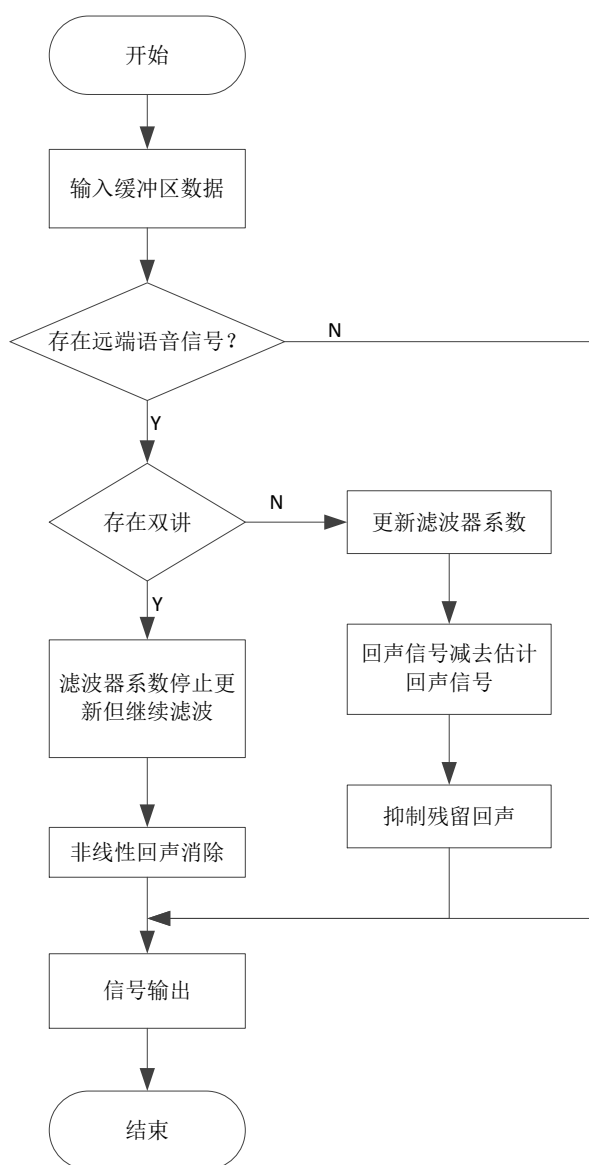


图 4.15 回声消除执行过程

4.3.3 课堂交流讨论模块

在登录的状态下，进入到多人聊天的界面，可进行创建新的聊天室或直接进入已经存在的聊天室进行课堂交流讨论。讨论过程中可根据需要自行进行聊天室进行消息读取。课堂交流讨论模块的流程图如图 4.16 所示。

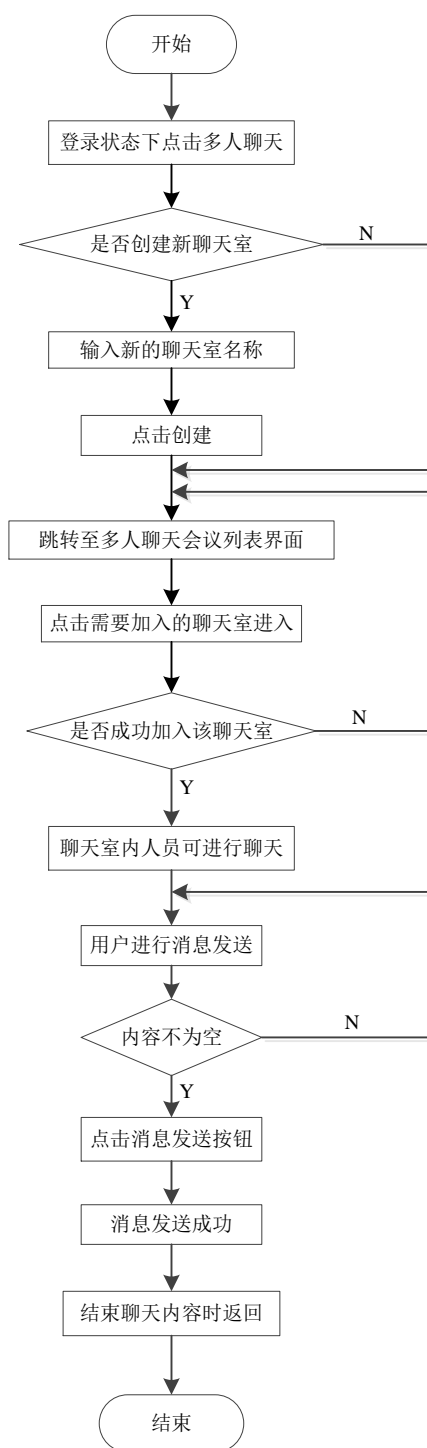


图 4.16 课堂交流讨论模块流程图

当音视频通信受到网络环境，比如带宽等因素的影响时，此时文本消息的传输带宽更小，传输性能更佳。再结合实际的系统设计需求，本系统的文本传输模块提供音视频课堂会议之外的课堂文本交流讨论功能，保证教师与学生的日常交流以及课堂问题答疑。课堂交流讨论功能模块主要是通过 PeerConnection 建立连接，RTCDatChannel 创建通道进行非媒体流实时数据传输^[63]。

4.3.4 课程管理模块

课程管理模块的流程图如图 4.17 所示。

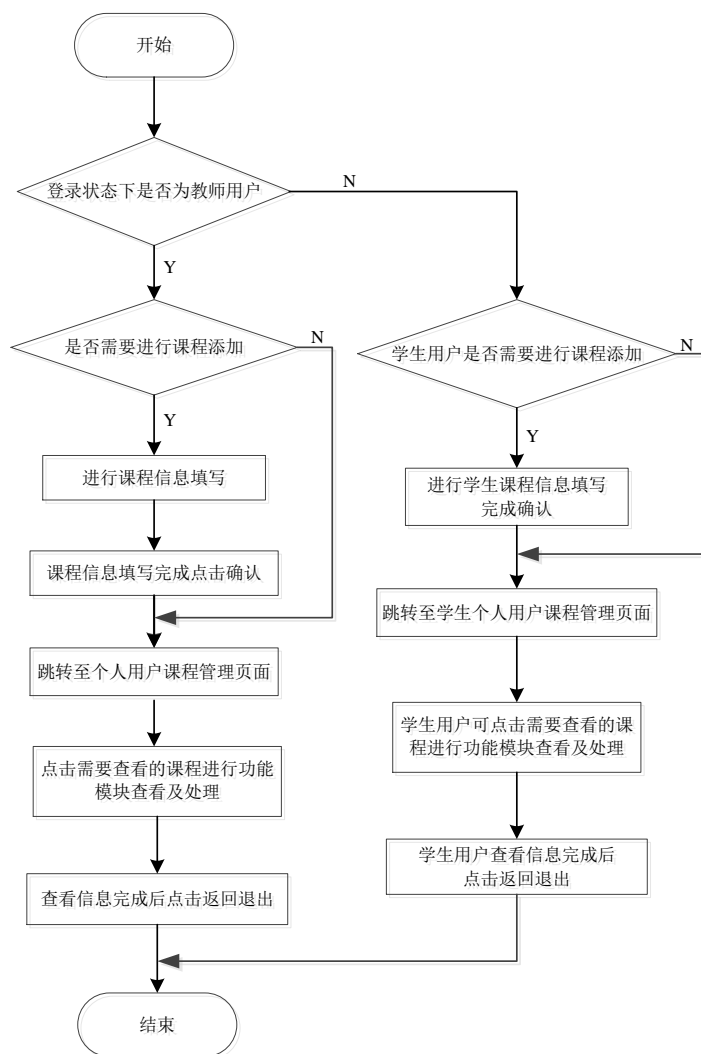


图 4.17 课程管理模块流程图

本模块的使用方法与前面的三个模块有不同之处，这里需要根据第一个用户注册登录模块中输入自己（学生或者教师）对应的学号或者工号区分本模块的使用权限。先选择自己的用户类别，如果是学生用户，可以进行课程的添加及删除

管理，还可以进行作业任务及课程的查看；如果是教师用户，相应地可以进行授课信息的添加，同时具备课程查看，发布作业，查看作业等权限，但是在教师用户与学生用户的设计上会有差别。

4.3.5 互动式课堂签到模块

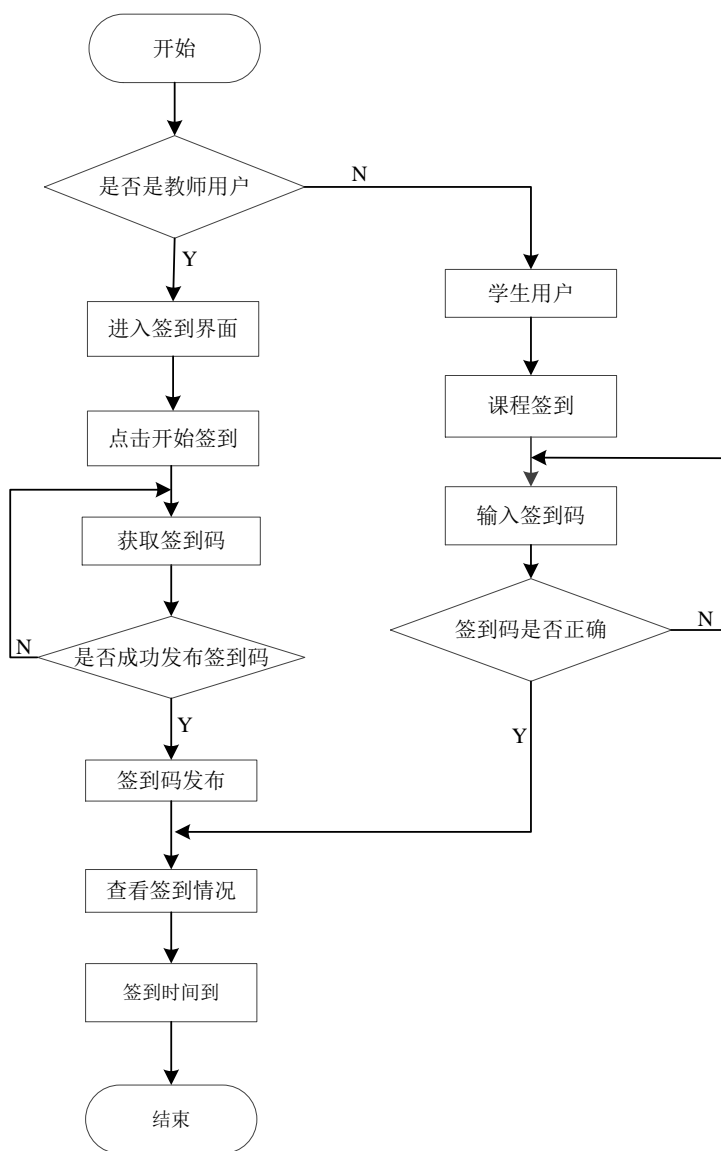


图 4.18 互动式课堂签到模块流程图

互动式课堂签到模块如图 4.18 所示。学生用户可以在教师用户公布签到码后进行课堂签到，同时教师用户还可以实现课堂签到情况查看以及课堂随机点名。本模块主要是为增加本设计系统的专业性，系统自动统计各项考勤数据，给出相应的统计报表，并记录到后台系统，供学校管理使用。

4.4 软件测试

4.4.1 测试环境

本系统使用的代码调试工具是 Android Studio，一些详细的服务器端及客户端的机型展示分别如表 4.2 和表 4.3 所示。本节展示的是部分测试手机的配置信息，通过对不同品牌的手机进行本设计系统 APP 的安装、使用 and 卸载操作，表明本系统可以供大部分安卓用户使用。

表 4.2 服务器配置信息

设备参数名	参数信息
操作系统	Ubuntu 16.04 LTS Xenial
分配内存	4GB
处理器	1 个

表 4.3 手机端测试设备配置信息

手机设备名称	处理器	系统版本	分辨率（像素）
荣耀 Play4T Pro	海思（八核）	Android 9	1600*4800
华为 nova8 5G	麒麟 985	Android 9	3200*6400
Vivo X60	三星（八核）	Android 9	3200*4800
Redmi K20 Premium Edition	高通骁龙 855Plus	Android 10	2340*1080
Opportunity R15	八核	Android 9	2280*1080

4.4.2 系统效果展示

1. 用户注册登录模块展示



(a) 用户登录模块功能展示



(b) 用户注册模块功能展示

图 4.19 用户注册登录模块效果展示

2. 音视频课堂会议模块展示

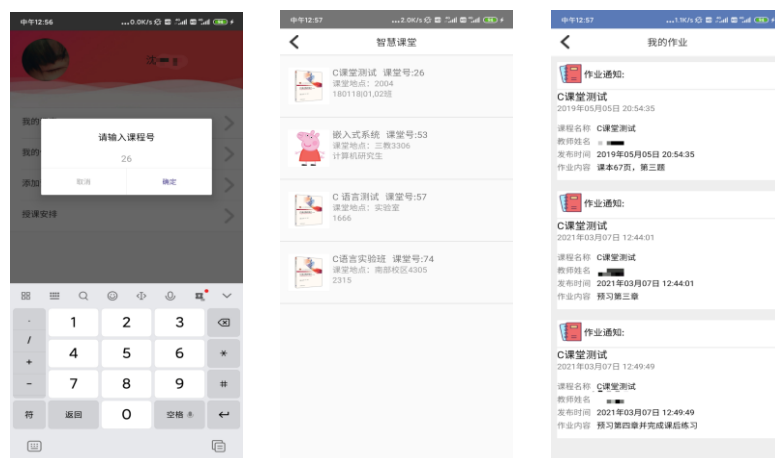


(a) 音视频课堂会议室界面展示 (b) 音视频课堂会议通信展示
图 4.20 音视频课堂会议模块效果展示

3. 课程管理模块展示

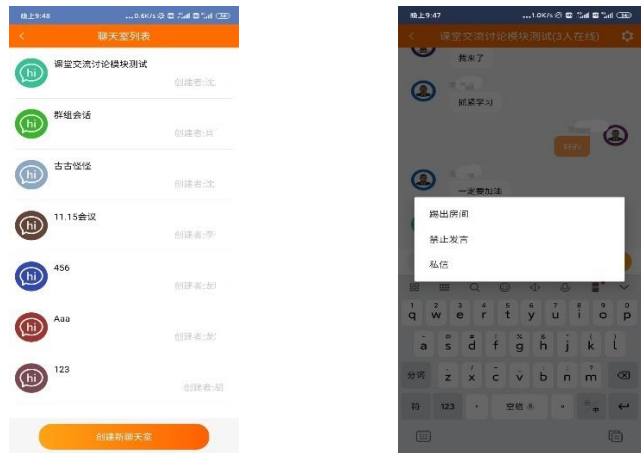


(a) 教师课程查询功能 (b) 教师课程发布功能 (c) 教师作业发布功能 (d) 教师作业查看功能



(e) 学生课程添加功能 (f) 学生课程查看功能 (g) 学生作业查看功能
图 4.21 课程管理模块效果展示

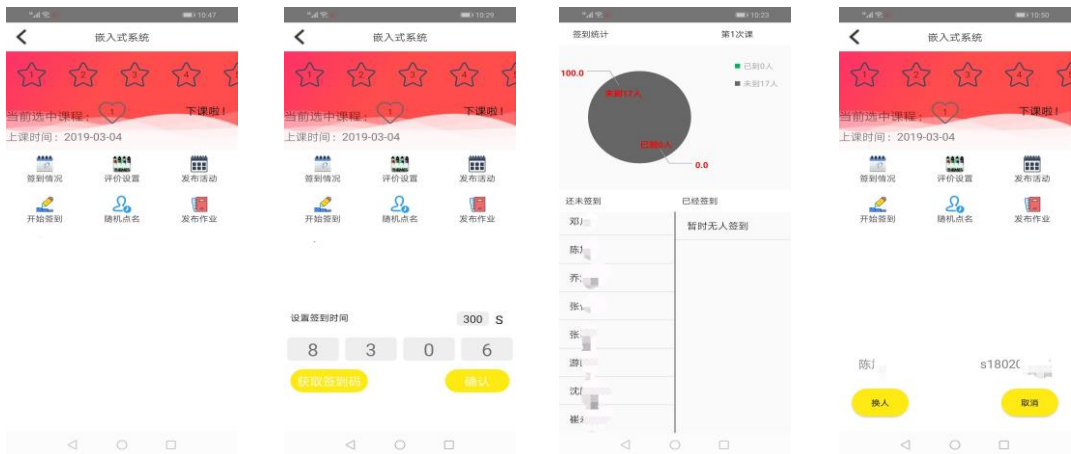
4. 课堂交流讨论模块展示



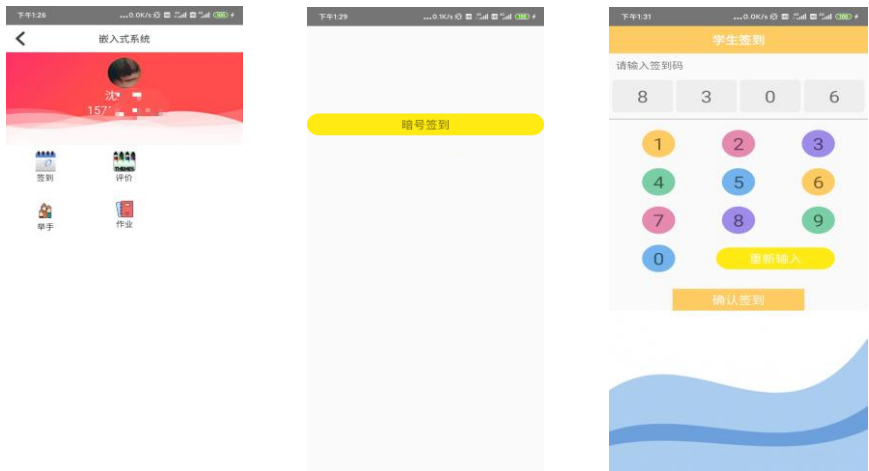
(a) 课堂交流讨论界面展示 (b) 课堂交流讨论功能展示

图 4.22 课堂交流讨论模块效果展示

5. 互动式课程签到展示



(a) 教师签到页面 (b) 教师设置签到码页面 (c) 教师查看签到情况 (d) 教师随机点名功能



(e) 学生签到页面展示 (f) 学生用户点击签到 (g) 学生使用签到功能
图 4.23 互动式课程签到模块效果展示

4.4.3 系统功能测试

系统的功能测试是检查实际软件的功能及可操作性是否符合用户的需求，它的最终目的是确保程序以期望的方式运行。功能测试可以分为手工测试和自动化测试，本系统设计采用手工测试方式，对系统的五大功能模块进行测试。

1. 用户注册登录模块测试

用户注册登录模块主要分为注册模块和登录模块，用户注册模块功能测试用例表如表 4.4 所示，用户登录模块功能测试用例表如表 4.5 所示。

表 4.4 用户注册模块功能测试用例表

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
1	无输入注册	已进入注册页面	1.所有输入均为空 2.点击快速注册	提示输入框不能为空	与预期结果一致，测试通过	高
2	正确的用户名+错误的手机号	已进入注册页面	1.输入用户名 2.输入错误格式的手机号（非 11 位）	点击获取验证码，提示手机号不正确	与预期结果一致，测试通过	高
3	正确的用户名+未注册的手机号	已进入注册页面	1.输入正确的用户名 2.输入未注册的手机号	点击获取验证码，进行一分钟可重新发送验证码倒计时	与预期结果一致，测试通过	高
4	正确的用户名、手机号+两次注册密码不一致	已进入注册页面	1.输入正确的用户名、手机号 2.输入两次密码不一致	点击快速注册，提示两次密码输入不一致	与预期结果一致，测试通过	高
5	正确的用户名、手机号+两次注册密码一致+学生/教师用户信息未填写	已进入注册页面	1.输入正确的用户名、手机号 2.输入正确的验证码 3.输入两次密码一致 4.用户者身份填写为空	点击快速注册，提示输入框不能为空	与预期结果一致，测试通过	中
6	正确的用户名、手机号+两次注册密码一致+学生/教师用户信息已填写	已进入注册页面	1.输入正确的用户名、手机号 2.输入正确的验证码 3.输入两次密码一致 4.用户者身份填写完整	点击快速注册，提示注册成功	与预期结果一致，测试通过	高

表 4.5 用户登录模块功能测试用例表

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
1	无输入登录	已进入登录页面	1.所有输入均为空 2.点击登录	提示输入框不能为空	与预期结果一致，测试通过	高
2	无密码登录	已进入登录页面	1.输入正确的用户名，不输入密码 2.点击登录	提示密码不能为空	与预期结果一致，测试通过	高
3	错误密码登录	已进入登录页面	1.输入正确的用户名、错误的密码 2.点击登录	提示输入正确的用户名或密码	与预期结果一致，测试通过	高
4	成功登录	已进入登录页面	1.输入正确的用户名和密码 2.点击登录	跳转至登录成功界面	与预期结果一致，测试通过	高

2. 音视频课堂会议模块测试

音视频课堂会议模块主要包括新建音视频课堂会议室、加入音视频课堂会议室、音视频课堂会议室摄像头音视频设备管理、退出音视频课堂会议室等功能。音视频课堂会议模块功能测试用例表如表 4.6 所示。

表 4.6 音视频课堂会议模块功能测试用例表

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
1	新建音视频课堂会议室且不输入会议室名称	已登录且进入线上会议界面	1.点击创建多人视频会议 2.不输入会议室名称信息	提示会议室 id 名称不能为空	与预期结果一致，测试通过	高
2	新建音视频课堂会议室且输入会议室名称	已登录且进入线上会议界面	1.点击创建多人视频会议 2.输入会议室名称信息	创建会议室成功，跳转视频会议列表界面	与预期结果一致，测试通过	高
3	直接进入已创建的音视频课堂会议室	已登录且进入线上会议界面	1.在视频会议列表中找到要加入的会议室名称 2.点击该会议室加入	进入到多人音视频课堂会议室中	与预期结果一致，测试通过	高
4	处于音视频课堂会议室中，且有相关用户进入	已登录且处于课堂会议室中	无操作	可以看到其他新用户加入	与预期结果一致，测试通过	高

表 4.6 (续)

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
5	处于音视频课堂会议室中摄像头麦克风关闭管理	已登录且处于音视频课堂会议室中	1.关闭麦克风 2.关闭摄像头	其他用户不能听到本用户的声音，然后其他用户不能看到本用户的画面	与预期结果一致，测试通过	高
6	处于音视频课堂会议室中摄像头麦克风打开管理	已登录且处于音视频课堂会议室中	1.打开麦克风 2.打开摄像头	其他用户能听到本用户的声音，然后其他用户能看到本用户的画面	与预期结果一致，测试通过	高
7	退出该音视频课堂会议室	已登录且处于音视频课堂会议室中	1.点击返回键并点击取消 2.点击返回键并点击确认	没有退出该音视频课堂会议室，点击确认后成功退出	与预期结果一致，测试通过	高

3. 课堂交流讨论模块测试

课堂交流讨论模块主要包括新建课堂交流讨论聊天室、加入课堂交流讨论聊天室、课堂交流讨论聊天室消息发送与实时接收、退出课堂交流讨论聊天室等功能。课堂交流讨论模块功能测试用例表如表 4.7 所示。

表 4.7 课堂交流讨论模块功能测试用例表

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
1	新建课堂交流讨论聊天室且不输入聊天室名称	已登录且进入线上聊天界面	1.点击创建新聊天室 2.不输入聊天室名称信息	提示聊天室 id 名称不能为空	与预期结果一致，测试通过	高
2	新建课堂交流讨论聊天室且输入聊天室名称	已登录且进入线上聊天界面	1.点击创建新聊天室 2.输入聊天室名称信息	创建聊天室成功，跳转聊天室列表界面	与预期结果一致，测试通过	高
3	直接进入已创建的课堂交流讨论聊天室	已登录且进入线上聊天界面	1.在聊天室列表中找到要加入的课堂交流讨论聊天室名称 2.点击该聊天室加入	进入到课堂交流讨论聊天室中	与预期结果一致，测试通过	高

表 4.7 (续)

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
4	处于课堂交流讨论聊天室中进行表情、文字信息发送	已登录且处于课堂交流讨论聊天室中	1.发送表情 2.发送文本消息	其他用户可以看到本用户发出的表情及文本消息	与预期结果一致，测试通过	高
5	其他用户退出该课堂交流讨论聊天室	处于课堂交流讨论聊天室中	1.有用户退出	可看到课堂交流讨论聊天室界面顶上的在线用户数量减少	与预期结果一致，测试通过	高
6	本用户退出该课堂交流讨论聊天室	处于课堂交流讨论聊天室中	1.点击返回键并点击取消 2.点击返回键并点击确认	先未退出该课堂交流讨论聊天室，点击确认后成功退出	与预期结果一致，测试通过	高

4. 课程管理模块测试

课程管理模块主要分为学生用户与教师用户两大类。两类终端用户的功能都有课程添加、课程查询和作业查询。但面对不同的用户，在设计功能的目的上会有差异。部分课程信息查询模块功能测试用例表如表 4.8 所示。

表 4.8 课程管理模块功能测试用例表

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
1	教师用户添加课程，课程开始时间不正确课程添加失败	教师用户状态下	1.点击添加课程 2.输入课程开始时间不正确 3.点击提交	提示开课时间不正确，请重新输入	与预期结果一致，测试通过	高
2	教师用户添加课程，课程信息添加完整且开课时间正确课程添加成功	教师用户状态下	1.点击添加课程 2.输入课程完整信息且输入正确的开课时间 3.点击提交	提示课程添加成功	与预期结果一致，测试通过	高
3	教师用户发布作业，发布作业成功	教师用户状态下	1.点击未结课班级发布作业 2.输入作业内容 2.发布作业	提示发布成功	与预期结果一致，测试通过	高

5. 互动式课堂签到模块测试

互动式课堂签到也分为教师用户和学生用户，这里的主要功能还是集中在教师用户，包括发布签到码、查看签到情况和随机点名等功能。学生用户通过教师用户发布的签到码签到。部分互动式课堂签到模块功能测试用例表如表 4.9 所示。

表 4.9 互动式课堂签到模块功能测试用例表

用例编号	用例标题	前置条件	操作步骤	预期结果	执行结果	优先级
1	教师用户成功发布签到码	教师在该课程的签到页面	1.点击开始签到 2.设置签到时间 3.点击获取签到码	点击确认，提示发布签到成功	与预期结果一致，测试通过	高
2	学生用户签到成功	学生用户进入签到页面	1.输入正确的课堂签到码 2.点击确认签到	提示签到成功	与预期结果一致，测试通过	高
3	学生用户签到不成功	学生用户进入签到页面	1.输入错误的课堂签到码 2.点击确认签到	提示请输入正确的签到码	与预期结果一致，测试通过	高
4	教师用户查看签到情况显示签到人数逐渐增多	教师用户已发布签到码且学生用户逐渐签到成功	1.点击智慧课堂下我的课表 2.选择进行签到的班级 3.点击签到情况	分别显示已签到的人数和未签到的人数	与预期结果一致，测试通过	高
5	教师用户随机点名	教师用户签到情况中已经显示有同学进行签到	1.点击智慧课堂下我的课表 2.选择进行签到的班级 3.点击随机点名 4.点击换人可更换	显示随机点到的学生的姓名及学号，点击换人可更换	与预期结果一致，测试通过	高

4.4.4 客户端性能测试

本节将电流、电压、温度及内存占用情况作为衡量客户端性能的关键指标。由于本章采用基于 Android 操作系统进行系统开发，因此这里将使用不同手机设备进行测试，部分机型展示如本章的表 4.2 所示。为验证本文的课堂通系统性能，本章还对比了主流的实时音视频通信系统：TIM、微信和腾讯会议。如图 4.24 所示，这

里选取一部分测试数据作为参考数据进行展示，下面对本设计系统的性能测试作简要说明。



图 4.24 课堂通 APP 电流、电压、温度及内存占用测试对比展示图

本次展示的数据是使用 Redmi K20 Pro Premium Edition 机型进行测试的，但客户端性能测试不限于本机型且不限于本次测试，其余测试数据未进行展示。本次测试主要分为电流、电压、温度及内存占用测试，横坐标表示的是时间，测试时间总计 60min，每条曲线使用的测试数据共计 1500 余组，可以得到相应的数据对比。课堂通电流基本在 -160mA 上下浮动，因有时频繁操作会导致电流波动较大，但属于正常范围；课堂通电压基本恒定在 3.729V 左右；课堂通温度基本在 25°C 左右浮动；课堂通内存占用基本恒定在 108.87MB 左右；与 TIM、微信及腾讯会议性能指标均值相比较电流、电压、温度易稳定且低能耗，内存占用情况低的性能。总的来说，系统耗能情况较小，电流、电压、内存占用都比较稳定，由此可见，系统具备一定的可靠性和稳定性，可以满足课堂会议系统的基本需求。

4.4.5 系统回声消除测试

为了进一步证明本文改进的 NLMS 算法的有效性,本节针对课堂通 APP 自适应回声消除效果进行了独立的测试。

1. 测试情景

系统设计完成后需要进行声学回声消除性能测试,主要测试多人音视频通信过程中双讲语音通话与近端和远端通话状态下的回声消除效果。

2. 测试环境及过程

本次实验在课堂通 App 真实运行环境下进行测试,测试场景选择在 90 平方米左右的实验室中,6 人通信过程中共记录 500 余组音频数据,3 男 3 女,平均每人 80 余条左右。每条语音数据通过本系统软件直接输入,在此过程中通过 CoolEdit 软件进行音频数据采集,在近端通话、远端通话、双讲语音通话模式下进行测试效果验证,如图 4.25。



(a) 六台设备同时接入显示界面

(b) 单台设备显示界面

图 4.25 六台设备接入回声测试界面

3. 实验结果及分析

在远端说话模拟远端语音信号,语音信号通过手机扬声器播放出来在房间内经多次路径传播形成回声信号。此时回声信号被手机麦克风捕获,利用软件进行音频数据采集。同理在双讲或者单端通话模式下,可通过采集音频数据,从而通过软

件上的语音信号波形来判断回声消除的效果。

(1) 单端模式下，仅存在远端信号

人为模拟远端信号，调整扬声器播放声量，远端语音信号输入波形如图 4.26 所示。

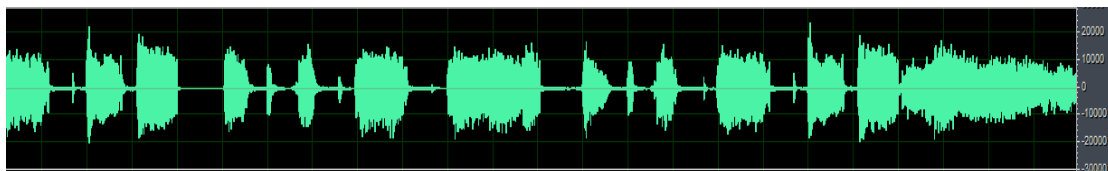


图 4.26 远端信号输入波形

此时，处理后的输出信号波形如图 4.27，可看出远端参考信号经回声通道传输之后得到的回声信号，再经 WebRTC 回声消除模块处理后得到的残余回声很小，这种回声几乎不会影响通话质量。

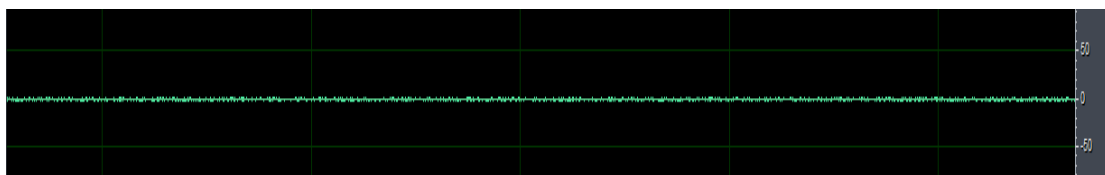


图 4.27 远端处理后的回声信号

(2) 单端模式下，仅存在近端信号

保持环境相对安静，让远端参考信号尽可能小至忽略不计，但实际场景中仍存在一定的背景噪声，近端语音信号输入波形如图 4.28 所示。

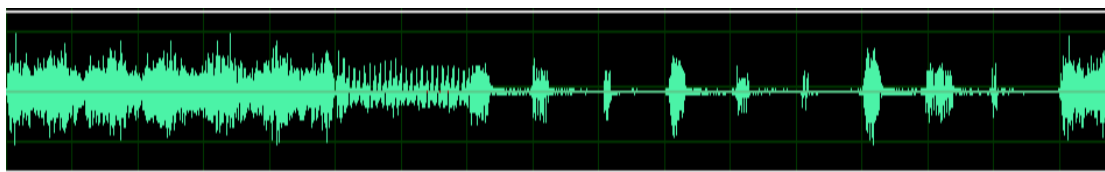


图 4.28 近端信号输入波形

此时，处理后的语音输出信号波形如图 4.29 所示，可以看到此时的近端信号的输入波形与处理后的输出信号波形基本一样，由于背景噪声影响，仅存在微小变化。

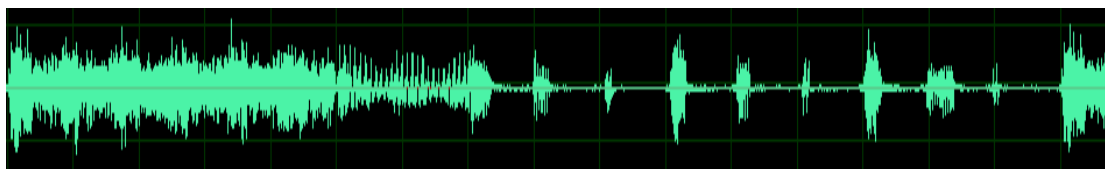


图 4.29 近端处理后的信号波形

(3) 双讲模式下，两端同时讲话

远端和近端同时进行语音信号输入，且实际实验环境中有一定的背景噪声，接下来对输入输出波形进行分析。

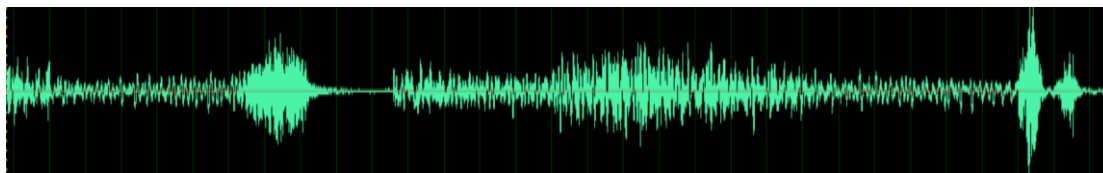


图 4.30 双讲模式下近端参考信号波形

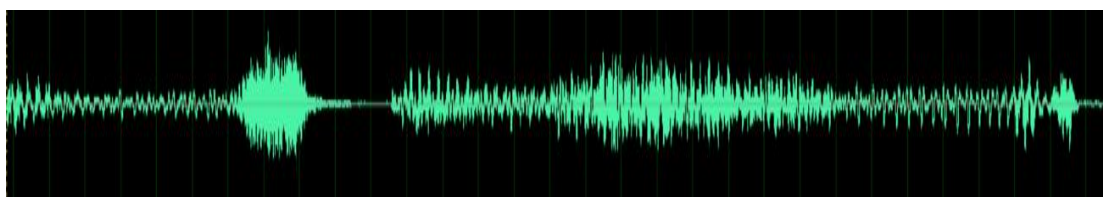


图 4.31 双讲模式下处理后的信号波形

图 4.30 是采集到的近端参考信号，包含远端参考信号产生的回声信号与近端语音信号；图 4.31 是经过 WebRTC 回声消除模块处理后输入信号。两者比较，发现处理后的信号相比处理前的参考信号，其语音包毛刺减少了很多并且语音幅值相对稳定，达到了回声消除的效果。

4.5 本章小结

本章是系统的设计实现以及效果展示。在第一章、第二章和第三章的理论分析基础上，结合项目的需求，设计实现基于 WebRTC 回声消除的音频系统。主要分为架构设计、客户端及服务器端设计，并对系统的各个功能模块进行分析及测试，通过实际运用验证系统的稳定性。架构选取主要是采取了 SFU 架构，充分从带宽消耗、音视频的服务质量及灵活性考虑。服务器端的设计主要包含了三大模块：Janus 开源流媒体服务器、基于 WebSocket 协议的 Collider 服务器实现信令模块及 coturn 服务器实现 NAT 穿透模块。本系统客户端功能设计主要包括用户注册登录、音视频课堂会议、课堂交流讨论、课程管理以及互动式课堂签到功能。最后通过采用多种手机机型对本系统 APP 逐一进行功能测试，性能测试对比以及回声消除测试，大量测试结果表明本设计系统的基础功能完善且性能稳定，其回声消除效果具有有效性。

第5章 总结与展望

5.1 主要工作与创新点

本文对 Google 公司推出的 WebRTC 实时通信技术的自适应回声消除算法进行改进，并基于 Android 平台设计实现 WebRTC 回声消除的音频系统。最后，通过规范的软件测试流程测试各个功能模块、系统性能和回声消除模块并改进相应的不足之处。针对以上重点，本文详述了 WebRTC 的核心技术，深入研究了 WebRTC 回声消除技术。在实现系统的过程中，重点分析了系统架构，服务器端和客户端设计，其中包括 WebRTC 源码编译，以此供 Android 端调用。

本论文的主要工作是：

1. 深入研究 WebRTC 的理论基础，如 WebRTC 技术的整体框架，掌握其组件部分、接口和响应协议等，为后文的系统设计奠定理论基础。
2. 对 WebRTC 中的回声消除模块进行深入研究、分析并改进。针对自适应算法收敛速度慢的问题，本文提出基于 NLMS 算法的改进算法。在声学回声消除情况下的仿真实验表明，考虑背景噪声影响且对步长因子给出更好的更新规则能有效提升算法的整体性能，包括收敛速度快、稳定失调少且跟踪能力强。
3. 完成基于 WebRTC 回声消除的音频系统设计与实现：包括前期的理论研究、需求分析，然后进行服务器端环境搭建和客户端模块实现。
4. 完成音频系统的功能展示、功能性测试、性能测试及回声消除效果测试。经过大量测试结果表明本系统可以实现课堂会议系统的基本功能需求。同时与主流实时音视频通信系统：TIM、微信和腾讯会议进行性能测试对比，结果表明本文的课堂通系统性能稳定。通过回声消除模块测试验证改进后的回声消除算法的有效性。总的来说，针对用户本系统具有良好的可操作性，软件维护费用低、运行流畅，具有实际的应用价值。

5.2 展望

通过本文 5.1 节所述的主要工作，本文取得了一定的研究成果。但是由于实验

条件以及实际背景环境噪声复杂，在设计后期发现本文存在一些不足之处需要改进：

一是对 WebRTC 的回声消除模块展开深入研究，对 WebRTC 自适应回声消除算法提出改进。这里的验证方式是通过仿真平台及实际的应用场景进行验证，但在真实的应用环境中测试时，语音数据集不够充分，后续将进一步扩大语音数据集来完善回声消除效果验证这部分实验工作。

二是本系统虽然完成了课堂会议系统的一些基本功能。但本文最重要的音视频课堂会议模块由于时间关系没有进一步得到优化，如当参与用户过多时，增至几十甚至是几百人，没有考虑到手机界面的展示问题。还有一些市场上已经存在的功能模块，如文件传输、会议主持等可以进一步进行系统功能添加。在日后的工作中，将完善系统的这些不足，从而提高课堂会议系统功能的丰富性和鲁棒性。

参考文献

- [1] Alkhulaiwi R., Sabur A., Aldughayem K., et al. Survey of secure anonymous peer to peer instant messaging protocols[C]//Privacy, Security & Trust. Auckland: IEEE, 2017: 294-300.
- [2] 徐政健. 通信运营商在移动互联网产业链中主导作用变化及其对策研究[D]. 宁波: 宁波大学, 2014.
- [3] Li Gaohe, Ding Yongqi, Xu Bo, et al. Development and research based on WebRTC mobile phone video communication[C]//2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). Chengdu: IEEE, 2019: 2487-2490.
- [4] Daldal B., Bilgin I., Basaran D., et al. Using web services for WebRTC signaling interoperability[C]//NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium. Istanbul: IEEE, 2016: 780-783.
- [5] Alimudin A., Muhammad A. Online video conference system using WebRTC technology for distance learning support[C]//2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC). Indonesia: IEEE, 2018: 384-387.
- [6] 张向辉. 基于 WebRTC 网络教学系统的设计与实现[D]. 武汉: 华中科技大学, 2015.
- [7] Romano S., Loreto S., Davids C. Real time communications in the web: current achievements and future perspectives[J]. IEEE Communications Standards Magazine, 2017: 20-21.
- [8] Togay C., Levi A. WebRTC based augmented secure communication[C]//Signal Processing & Communication Application Conference. Zonguldak: IEEE, 2016: 1621-1624.
- [9] 魏自强. WebRTC 会议系统的回声消除算法研究与设计实现[D]. 重庆: 重庆邮电大学, 2019.
- [10] Wu Fuying, Tong Fen. Non-uniform norm constraint LMS algorithm for sparse system identification[J]. IEEE Communications Letters, 2013, 17(2): 385-388.
- [11] Su Guolong, Jian Jin, Gu Yuantao, et al. Performance analysis of l_0 norm constraint

- least mean square algorithm[J]. IEEE Transactions on Signal Processing, 2012, 60(5): 2223-2235.
- [12] Duttweiler D. Proportionate normalized least-mean-squares adaptation in echo cancelers[J]. IEEE Transactions on Speech and Audio Proceeding, 2000, 8(5): 508-518.
- [13] Ghauri S., Sohail M. System identification using LMS, NLMS and RLS[C]//2013 IEEE Student Conference on Research and Developement. Putrajaya: IEEE, 2013: 65-69.
- [14] Zhang Sheng, Zhang Jiashu, Han Hongyu. Robust variable step-size decorrelation normalized least-mean-square algorithm and its application to acoustic echo cancellation[J]. IEEE/ACM Transactions on Audio Speech & Language Processing, 2016, 24(12): 2368-2376.
- [15] Lee K., Baek Y., Park Y. Nonlinear acoustic echo cancellation using a nonlinear postprocessor with a linearly constrained affine projection algorithm[J]. Circuits and Systems II: Express Briefs, IEEE Transactions on, 2015, 62(9): 1-1.
- [16] Wang Nan, Haiquan Zhao. A variable step size LMS adaptive filtering algorithm based on maximum correntropy criterion for identification of low frequency oscillation modes[J]. IFAC-PapersOnLine, 2019, 52(24): 163-167.
- [17] Lqbal N., Bashir M., Zerguine A. Convex combination of transform domain LMS and sparse LMS[C]//2018 52nd Asilomar Conference on Signals, Systems, and Computers. California: IEEE, 2018: 1726-1729.
- [18] Salman M., El-Sayed F., Youssef A. A sparse variable step-size LMS algorithm for impulsive noise[C]//2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART). Paris: IEEE, 2019: 1-4.
- [19] Rahman S., Rashid M., Alam M. A unified analysis of proposed wavelet transform domain LMS algorithm for ARMA process[C]//2019 5th International Conference on Advances in Electrical Engineering (ICAEE). Dhaka: IEEE, 2019: 195-200.
- [20] Shen Binbin, Lv Xiafu, Shuang Zhang. An improved LMS adaptive filtering algorithm and its analysis[C]//2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS). Chongqing: IEEE, 2019: 549-551.
- [21] 王显飞. 用于包络跟踪功放的线性化技术研究[D]. 四川: 电子科技大学, 2014.
- [22] Morgan R., Kratzer S. On a class of computationally efficient, rapidly converging, generalized NLMS algorithms[J]. IEEE signal processing letters, 1996, 3(8): 245-

- 247.
- [23] Egelmeers G., Sommen P., De B. Realization of an acoustic echo canceller on a single DSP[C]//1996 8th European Signal Processing Conference (EUSIPCO 1996). Trieste: IEEE, 1996: 1-4.
- [24] Cola C., Vaele H. On multi-user web conference using WebRTC[C]//2014 18th International Conference on System Theory, Control and Computing (ICSTCC). Sinaia: IEEE, 2014: 430-433.
- [25] Phankokkruad M., Jaturawat P. An evaluation of technical study and performance for real-time face detection using web real-time communication[C]//2015 International Conference on Computer, Communications, and Control Technology (I4CT). Kuching: IEEE, 2015: 162-166.
- [26] 龚琦. 基于 WebRTC 的实时通信能力平台研究[D]. 北京: 北京邮电大学, 2015.
- [27] Sredojev B., Samardzija D., Posarac D. WebRTC technology overview and signaling solution design and implementation[C]//2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO). Opatija: IEEE, 2015: 1006-1009.
- [28] 熊雨新. 基于 WebRTC 引擎的音频视频交互系统设计与实现[D]. 四川: 电子科技大学, 2014.
- [29] 曾照成. 基于 WebRTC 的视频会议系统的设计与实现[D]. 长沙: 中南林业科技大学, 2016.
- [30] 莫倩雯. 电子数据存证可信性评估及存证方法研究[D]. 重庆: 重庆邮电大学, 2018.
- [31] Zhang Lijing, Shen Xiaoxiao. Research and development of real-time monitoring system based on WebSocket technology[C]//Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC). Shengyang: IEEE, 2013: 1955-1958.
- [32] 陆晨, 冯向阳, 苏厚勤. HTML5 WebSocket 握手协议的研究与实现[J]. 计算机应用与软件, 2015, 32(01): 128-131+178.
- [33] 林贵春. 用于回声消除的变步长自适应算法研究[D]. 武汉: 华中科技大学, 2019.
- [34] Haykin S. 自适应滤波器原理(第四版)[M]. 郑宝玉, 译. 北京: 电子工业出版社, 2010: 1-25, 159-298.

- [35] 陈旭. 特定场景下的变步长自适应滤波算法研究[D]. 苏州: 苏州大学, 2020.
- [36] 秦海娟. 数字助听器中回声消除算法的研究[D]. 南京: 南京邮电大学, 2015.
- [37] Kuo S., Morgan D. Active noise control systems: algorithms and DSP implementations[M]. John Wiley & Sons, Inc. 1996: 138-142.
- [38] Ma G., Gran F., Jacobsen F., et al. Adaptive feedback cancellation with band-limited LPC vocoder in digital hearing aids[J]. IEEE Transactions on Audio Speech & Language Processing, 2011, 19(4): 677-687.
- [39] Dandach S., Fidan B., Dasgupta S., et al. Adaptive source localization by mobile agents[C]//Proceedings of the 45th IEEE Conference on Decision and Control. California: IEEE, 2006: 2045-2050.
- [40] Zarzoso V., Nandi A. Adaptive blind source separation for virtually any source probability density function[J]. IEEE Transactions on signal processing, 2000, 48(2): 477-488.
- [41] Teja M., Meghashyam K., Verma A. Comprehensive analysis of LMS and NLMS algorithms using adaptive equalizers[C]//2014 International Conference on Communication and Signal Processing. Melmaruvathur: IEEE, 2014: 1101-1104.
- [42] Malik S., Enzner G. State-space frequency-domain adaptive filtering for nonlinear acoustic echo cancellation[J]. IEEE Transactions on Audio Speech and Language Processing, 2012, 20(7): 2065-2079.
- [43] Park Y., Park H. DTD-free nonlinear acoustic echo cancellation based on independent component analysis[J]. Electronics Letters, 2010, 46(12): 866-868.
- [44] Bernardi G., Waterschoot T., Wouters J., et al. An all-frequency-domain adaptive filter with PEM-based decorrelation for acoustic feedback control[C]//2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). New Paltz: IEEE, 2015: 1-5.
- [45] Wang Qingyun, Chen Xin, Liang Ruiyu, et al. A frequency-domain nonlinear echo processing algorithm for high quality hands-free voice communication devices[J]. Multimedia Tools and Applications, 2021: 1-20.
- [46] Padhi T., Kar A., Chandra M. Family of adaptive algorithms based on second order volterra filters for nonlinear acoustic echo cancellation: a technical survey[C]//2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI). Delhi: IEEE, 2014: 909-914.
- [47] 曹亚丽. 自适应滤波器中 LMS 算法的应用[J]. 仪器仪表学报, 2005(S2): 452-

- 454.
- [48] Barik A., Mohanty M., Das K. Convergence analysis of adaptive MSFs used for acoustic echo cancellation[J]. International Journal of Information and Communication Technology, 2018, 13(2): 196-207.
- [49] 马国栋, 阎树田, 贺成柱, 等. 基于 LMS 算法与 RLS 算法自适应滤波及仿真分析[J], 电子设计工程, 2014, 22(06): 43-45+49.
- [50] 严涛. 声学回声消除的自适应滤波方法研究[D]. 南京: 南京信息工程大学, 2020.
- [51] Aboulnasr T., Mayyas K. A robust variable step-size LMS-type algorithm: analysis and simulations[J]. IEEE Transactions on Signal Processing, 2002, 45(3): 631-639.
- [52] 田斌鹏, 张翠芳, 闫磊. 一种新的可变步长 LMS 自适应滤波算法[J]. 计算机仿真, 2007, 24(06): 89-91+238.
- [53] Wu Xueli, Liang Gao, Tan Zizhong. An improved variable step size LMS algorithm[C]//Proceedings of 2013 2nd International Conference on Measurement, Information and Control (ICMIC). Harbin: IEEE, 2013: 533-536.
- [54] Benesty J., Rey H., Vega L., et al. A nonparametric vss nlms algorithm[J]. IEEE Signal Processing Letters, 2006, 13(10): 581-584.
- [55] Fazel A., El-Khamy M., Lee J. CAD-AEC: context-aware deep acoustic echo cancellation[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Barcelona: IEEE, 2020: 6919-6923.
- [56] 杜民. 视频会议系统中 DirectShow 技术的研究与应用[D]. 哈尔滨市: 哈尔滨理工大学, 2008.
- [57] 高梓尧. 基于 WebRTC 的移动视频会议系统的设计与实现[D]. 武汉: 华中科技大学, 2019.
- [58] Chai R., Nguyen H. Real-time WebRTC-based design for a telepresence wheelchair[C]//2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). Jeju: IEEE, 2017: 2676-2679.
- [59] Jiang Chuanyan, Yan Linchen, Tei Xunzai, et al. A video conferencing system based on WebRTC for seniors[C]//2014 International Conference on Trustworthy Systems and their Applications. Taichung: IEEE, 2014: 51-56.
- [60] Shukhman A., Polezhaev P., Legashev L., et al. Creation of regional center for shared access to educational software based on cloud technology[C]//2017 IEEE Global

- Engineering Education Conference (EDUCON). Greece: IEEE, 2017: 916-919.
- [61] Sarkar I., Rouibia S., Lopez-Pacheco D., et al. Proactive Information dissemination in WebRTC-based live video distribution[C]//2020 International Wireless Communications and Mobile Computing (IWCMC). Limassol: IEEE, 2020: 304-309.
- [62] 王昕. 基于 WebRTC 的视频会议系统的设计与实现[D]. 沈阳: 沈阳工业大学, 2019.
- [63] 李宇轩. 基于 WebRTC 的即时通信视频系统的设计与实现[D]. 北京: 北京交通大学, 2016.

致谢

时光是一张有去无返的单程票，关于崇文路 2 号有一段我带不走的美好。在三年研究生生活期间，我在努力提高自我修养的同时还认识了很多帮助过我的老师、同学及朋友。借此机会，向你们表达感谢，谢谢你们！

首先非常感谢我的导师龙昭华教授，当初我刚开始进行未来网络标准课题研究的时候，对其并不熟悉，是龙老师一直在耐心地教导我和鼓励我，使我的专业知识日益精进。龙老师谨慎执着的科研态度，对研讨会定期召开的科研精神都极大地影响了我并不断帮助我成长。通过研究生阶段的学习，如今我面对任何事情，都比以往有着更加坚定和执着的心态。在此，再一次向龙老师致以最真诚的谢意！

感谢信科 2004 实验室这个大家庭，让我的研究生生活更加灿烂。感谢我的同门、师兄、师姐、师妹、师弟。还有感谢学校提供的科研环境，让我们能够在良好的环境中学习！

特别感谢最爱我的父母和外婆，谢谢你们对我无条件的支持和保护，无论什么时候都是以我本人的想法为主导来关心和激励我。感谢所有爱我的家人！

感谢相识七年的同窗兼闺蜜对我的包容和陪伴。感谢所有帮助我的朋友，谢谢你们！

最后，由衷地感谢各位审阅论文和答辩委员会的老师，感谢各位老师的批评和指导！

栀子花季，来日再访，祝愿老师们工作顺利，同学们前程似锦！

攻读硕士学位期间从事的科研工作及取得的成果

参与科研项目：

- [1] 会训无线投影云管理系统的推广(KJZH17118), 重庆市教委高校优秀成果转化项目, 2018 年 12 月-2021 年 4 月。
- [2] 重庆交运新天地环循科技有限公司信息化管理平台建设项目(E021E2020110), 2020.12-至今。
- [3] ISO 国际标准研究项目(ISO/IEC CD 21558-2): Information Technology - Future Network -Architecture - Part 2: Proxy model based Quality of Service.
- [4] ISO 国际标准研究项目(ISO/IEC CD 21559-2): Information Technology - Future Network - Protocols and Mechanisms - Part 2: Proxy Model based Quality of Service.

发表及完成论文：

- [1] **Lizhi Shen**, Zhaohua Long. Principle and Algorithms of WebRTC Echo Cancellation [C]// 2020 International Conference on Software, Modeling and Intelligent Systems (SMIS 2020), 2021年5月已出刊.
- [2] 范天文, 龙昭华, **沈励芝**. 基于内存计算框架Spark的性能优化和参数配置方法. 中国, 201911241267.9. [X]P.2019-12-06.
- [3] 龙昭华, 邓青青, 梅文博, **沈励芝**. 多终端多用户无线投影云管理系统 V1.0. 中国. 计算机软件著作权. 登记号: 2019SR0743005.