专 业 学 位 硕 士 学 位 论 文

基于 FFmpeg 的教育直播系统设计与实现

Design and implementation of education live broadcast system based on FFmpeg

作 者 姓 名：陶奎印

工 程 领 域：电子与通信工程

学　　号：31809093

指 导 教 师：王洪玉　教授

完 成 日 期：2021-6-10

大连理工大学

Dalian University of Technology

摘　要

教育直播是一种新兴的教学模式，依托于互联网直播技术的飞速发展，让教学课堂不再拘泥于地点的限制，教师可以使用教育直播系统将教学信息实时地传递到多个学生的终端设备上，实现随时、随地的授课。现今市场上教育直播平台种类繁多，但存在着安全性、私密性、体验感受、适用性等方面的问题。

针对上述情况，本论文在深入研究直播技术的基础上，根据教学课堂的实际应用需求，设计了一种基于 FFMpeg 的教育直播系统。该系统划分为推流端、服务器端和收流端，具有支持多业务处理能力、操作方便、可无线传输、工作环境需求低等优点，能够满足学校教室、实验室等环境下的教学需求。本文主要的研究工作包括：首先采用 H.264 视频编码技术和 AAC 音频编码技术，完成对原始数据的压缩，保证数据质量的同时减小了传输数据量大小，降低对网络环境的依赖程度；其次，提出一种基于 FFMpeg 的首帧播放优化方案，有效地将首帧播放时间缩短至 1~2 秒，提升了用户使用体验；最后，利用 Nginx 服务器完成流媒体数据的转发，并提出一种 Samba 服务与 Nginx 文件下载功能相结合的文件传递方式；同时，采用 Nginx 鉴权功能，极大地防止了恶意信息的流入。

为了验证本文提出的基于 FFmpeg 的教育直播系统的性能，根据教育直播系统的应用场景，本文设计了多组实验检验系统是否可以满足教学课堂的使用需求，包括延时性、稳定性、多推流端切换等。实验结果表明，本系统在正常和极端情况下均能保持良好的运行状态，延迟及安全性能均可满足需求，是革新教学方式的可行性案例。

关键词：流媒体；FFmpeg；Nginx 服务器；RTMP

Design and implementation of education live broadcast system based on FFmpeg

Abstract

Education live broadcasting is a new teaching mode. Relying on the rapid development of internet live broadcasting technology, the teaching classroom is no longer limited by the location. Teachers can use the education live broadcasting system to transmit teaching information to multiple students' terminal devices in real time, so as to realize teaching at any time and anywhere. Nowadays, there are many kinds of education live broadcasting platforms in the market, but there are some problems such as security, privacy, experience and applicability.

In view of the above situation, this paper designs a FFMpeg based education live broadcast system based on the in-depth study of live broadcast technology and the practical application requirements of classroom teaching. The system is divided into data push stream end, server end and data receive stream end. It has the advantages of supporting multi-business processing ability, convenient operation, wireless transmission, low working environment demand, and can meet the teaching demand in classrooms, laboratories and other environments. The main research work of this paper includes: firstly, using H.264 video coding technology and AAC audio coding technology, complete the compression of the original data, ensure the data quality and reduce the dependence on the network bandwidth; Secondly, an optimization scheme based on FFMpeg is proposed to effectively shorten the reading time of the first frame to 1~2 seconds, which improves the user experience. Finally, Ngnix is used to complete the forwarding of streaming media data, and a file transfer method combining Samba service and Ngnix file download function is proposed.

In order to verify the performance of the education live broadcast system based on FFmpeg proposed in this paper, the basic functions of the system were tested in the real teaching environment, and experiments were designed for various key functions of the system, including delay, stability, multi-push stream end switching, etc. The experimental results show that the system can maintain a good running state under normal and extreme conditions, and the delay and safety performance can meet the requirements.

Key Words：Streaming media; FFmpeg; Nginx server; RTMP

目　录

1 绪论

1.1 研究背景和意义

近年来我国经济的发展得到了长足的进步，对科学技术与基础教育的投入也逐年增加，多媒体教学设备已经成为各大中小学的标准配置。当今的教学方式也以使用投影仪播放教师屏幕进行教学活动为主。这种教学方式给教师授课，学生接收信息带来了极大的便利的同时也有一些使用上的缺陷，例如教师在授课过程中，后排、角落等位置的同学观看投影仪的视角有限，这样就会导致一个后果：部分位置的同学将不能够清晰的看到投影仪上的文字，使得学生接收教师授课信息出现阻碍；其次，投影仪一般采用 VGA 或 HDMI 接口，常年使用情况下易出现设备老化或损毁的问题；另外，投影仪教学方式仅适用于单独一个教室，并不能做到多个教室同时接收教师讲课数据；最后，这种通过投影来授课的方式，并没有充分地使用发展良好的互联网科学技术。因此使用流媒体技术进行教学活动的趋势已经日益增大。

流媒体是指将多媒体文件以数据流的形式传输于各个设备之间[1]。在传输数据之前，将视频、音频等多媒体文件进行编码压缩。经由流媒体服务器进行转发，最后传输到目标计算机。相较于传统的多媒体，流媒体所具有的优点包括：

（1）下载速度快，占用的存储空间小

（2）能够支撑多用户同时使用

（3）支持实时访问

（4）数据连续播放

因为流媒体数据所具有的特点，各个领域内流媒体的普及率逐渐增多，包括网站视频播放、电商平台、直播互动、线上会议、物联网等众多网络行业。并且随着 5G 时代的到来，流媒体技术一定会得到长足的发展[2]。流媒体技术已经成为互联网技术的中坚力量。

根据《2019-2025 年中国在线直播行业市场竞争格局及未来发展趋势报告》的计算，2019 年我国的直播行业规模达到 710 亿元，随着手机等移动直播的兴起和人们消费水平的提升，2021 年我国直播行业规模将会达到 940 亿元。

虎牙直播、bilibili、斗鱼直播等部分网络在线直播的软件，受到广大网友的热烈追捧。在教育行业软件当中的领头羊，诸如学而思、好未来、猿辅导等也都在各自教学活动中使用上了教育直播[3]。在当下的社会潮流之中，对于在线教育平台输出内容和运营方式这类问题，教育部作出了更深层次的分析与指导，指出网络教育更应当注重信息传

递和资源传递的优势，以此来更好地将优质教学资源推广到社会各个角落，实现能够在使用教育直播平台时，可以充分的将优质教学资源同现代化教育平台相结合。针对教育部提出的要求，部分学校着手使用市场上开放而且免费的互联网直播平台，但是也产生了相应的弊端，一些恶意的信息就会经由互联网涌入到课堂当中，这样对学校来说进行正常教学活动就会受到其干扰 [4]。

本文研究的系统首要的目标是支持并满足正常授课需求，其次解决上述各大互联网直播平台存在的问题以及传统投影仪教学方式存在的问题，包括跨越物理空间的限制、多设备同时接收数据、防范恶意信息流入等问题。利用本文设计并实现的教育直播系统学生们可以实时获取教师授课信息，教师进行课件演示、软件操作教学、多终端同步等操作的时候就方便很多了，可以更加明显的提高课堂效率，也会提高学生学习乐趣。

1.2 研究现状

虽然我国互联网直播领域相较于国外起步较晚，但由于国内经济发展迅速，借着经济发展的快车，网络信息交互的及时性，我国的直播技术的发展与国外基本保持一致。在当今市场上较为成熟的直播框架均为服务器加客户端的模式，其主要方式为直播客户端将捕获到的屏幕数据和麦克风数据进行编码压缩，之后传输到服务器进行转发，最后客户端接收到经过压缩的屏幕数据和麦克风数据。直播系统都是采用推流端发送，服务器转发，收流端接收的运行模式，但是实现模式的技术是各个系统性能不同的关键所在。

一般情况下，直播画面的优劣是最影响用户体验的因素。当客户端软件播放原始音视频时，会让用户得到最好的使用体验，但是原始音视频数据往往信息量极大，对网络传输带宽要求高，因此为了平衡音视频观看质量和传输信息量之间的关系，在音视频传输前，要对采集到的数据进行压缩编码处理。由此可见，关于直播框架，编码技术处在了一个不可或缺的位置，处在相同码率和帧率条件下，使用性能优异的编码方式会获得更好的观看体验和传输效率。在 1988 年，ITU-T（International Telecommunication Union，国际电讯联盟）的 VCEG（视频编码专家组）和 ISO/IEC（International Standardization Organization，国际标准化组织）的 MPEG（运动图像专家组）以为视频编码定制统一规范为出发点，立足于将视频编码标准适用在各个领域，满足不同需求，共同提出了 H.26x 和 MPEG-x 视频编码标准[5]。H.26x 视频编码标准的更加侧重于网络视频传输的应用，以网络传输编码为主要目标。而 MPEG-x 视频编码标准，着手于视频存储领域。面对视频技术发展的迅猛势头，2002 年我国也着手研发适用于我国视频领域的视频编码标准，最终于 2005 年推出了完全由我国研发的视频编码标准 AVS（Advance Vusial System），同 H.26x 系列的 H.264 相比，视频压缩率已经几乎相同[6]。

在直播框架中，客户端与服务器之间的网络传输协议的选择也是整个体系重要的一环。一般情况下，TCP 协议的消息确认 ACK、拥塞控制与流量控制等机制会对直播传输的实时性造成严重的影响，因此流媒体数据在进行网络传输时，通常不会选择 TCP。在流媒体传输领域中主要包括以下传输协议：RTSP 实时传输协议、RTP 实时传输协议、RTMP 传输协议与 RTCP 控制协议等等。RTP 协议因其自身的局限性，一般情况下仅仅作为一个协议框架，搭配 RTCP 来应用。RTSP 协议有着优秀的性能，但是 RTSP 在使用时最少要有两个通道进行数据传输以及命令。相较于其他传输协议，RTMP 协议作为应用层协议通常使用 TCP 协议作为自己在传输层的协议，与此同时还添加了 SSL 等确保传输的安全性，使得流媒体数据传输在安全方面更深层次的加强[7]。有关传输效率，RTMP 将大块的数据分割成为小块的消息，并使用消息包头进行拼接，提高了传输效率。

流媒体数据在网络传输中的实时性、可靠性也依赖于系统使用的流媒体服务器[8]，目前直播行业主流的流媒体服务器包括：

（1）AMS

AMS 是一款商业产品，用户需购买后才能使用，价格在 4-5 万之间，是由 Adobe 公司研发推广的流媒体服务器，它是第一款用于流媒体系统的服务器。AMS 在行业内处于领先的地位，能够迅速部署用于直播和点播的服务器，并且支持 RTMP/RTMPE 协议、HLS（m3u8）等。

（2）SRS

SRS 设计目标是支持大规模的互联网直播服务器集群。SRS 为了更好的搭配 RTMP 协议提供了对 RTMP 数据的多种处理方式，例如转码、截图、转发至其他服务器、封装类型转换、录制为 FLV 格式[9]。SRS 作为互联网直播服务器集群，具有处理大规模业务的良好性能包括 CDN 业务、RMPT 多层集群、虚拟服务器等。除此之外，SRS 还向用户开放了自身函数调用权限，例如 HTTP 相关函数接口、RTMP 相关函数接口、安全策略接口等。

（3）Red5

Red5 支持 RTMP 协议，在应用方面可以完全兼容 AMS 服务器，可轻松替换 AMS 服务器且客户端无需做任何更改，提供了种类繁多的功能包括视频回播、共享对象等。

上述列举的 AMS、SRS 和 Red5 流媒体服务器，深度依赖于硬件性能，结构复杂、架构繁重[10]。本文的流媒体服务器采用了部署 Nginx 服务器并在其上添加第三方 RTMP 模块的方案，Nginx+RTMP 具有如下优点：

（1）Nginx 是一款基于 C 语言的轻量级服务器，采用模块化设计。

（2）Nginx 配置文件操作便捷，使用较为方便，

因此，以架构简洁、意见依赖程度低为首要考量指标，Nginx 服务器是最为适合本系统的流媒体服务器。

1.3 研究内容

本论文的主要关注内容是将互联网与教学相结合，设计并实现校园环境下的教学直播系统。教师可以将自己授课电脑的屏幕、麦克风的声音数据实时地推流给多个教室、多名学生的电子设备上，实现远程、任何时间地点的授课。教师在使用教学直播系统时，不再需要专用的多媒体机房以及投影仪，利用该教学直播系统在任何教室都可以向学生展示播放内容，将极大地提高教师授课效率。

本论文的主要工作为：

（1）研究直播技术以及相关理论，确认教学直播系统的技术办法。分析教学直播系统的需求，我们设计了教育直播系统的主要框架和满足需求的技术实现的逻辑思维方式，而且对有关的技术进行了深入的研究和学习。

（2）对 FFmpeg 音视频处理框架进行研究，掌握 FFMpeg 处理音视频数据的相关方法和原理。在大量针对性实验的基础上，针对本论文提出的教育直播系统，通过对 FFmpeg 框架中函数参数的优化，缩短了教学直播过程中推流端与收流端的延迟。

（3）搭建了 Nginx 服务器，并在该服务器上添加了 RTMP 功能模块，以实现直播数据的转发。针对本文面向教学课堂的应用场景，实现了 Nginx 文件下载服务与 Samba 服务相结合的文件传递方式，极大地提高了师生文件传递的便捷性。并深入研究 Nginx 鉴权模块，编写 php 信息校验脚本，有效的提升了系统安全性能。

（4）在整体教学直播系统设计并实现完成后，为了测试系统各项功能的是否能够满足应用需求，设计了多组针对性实验，并对实现的教学直播系统的使用效果进行了相关评估。

1.4 论文组织结构

本论文的组织结构如下：

第一章，绪论。首先对本论文课题的研究背景进行了简要介绍，之后对传统投影仪教学方式存在的问题进行分析，然后详细阐述了直播系统使用技术的相关背景以及优缺点。在本文研究内容一节中，论述了本文的主要工作，根据课堂授课使用需求以及市场上直播平台存在的问题提出了解决方案并实现。

第二章，相关技术和原理。对教育直播系统在实现过程中使用到的技术进行研究，包括 FFMpeg、H.264 视频编码技术、AAC 音频编码技术以及 RTMP 协议。

第三章，系统总体设计。针对本系统的校园应用环境，本文提出五点功能性需求。之后根据需求设计了系统的组成，将整个系统分为推流端、服务器、收流端三个大部分。在接下来的章节，根据各个部分不同的功能需求，对不同部分的进行了设计与实现。

第四章，教育直播系统的服务器实现。详细介绍了 Nginx 服务器的研发背景、架构、配置文件等。之后成功部署了 Nginx 服务器，并在 Nginx 服务器成功添加第三方 RTMP 模块，使服务器可以转发流媒体数据。实现了在 Nginx 服务器添加文件下载 server 以及在服务器上添加 Samba 服务，方便了教师和学生之间传递文件。实现了系统推流鉴权功能，保障了教学过程中的信息流入安全性。

第五章，教育直播系统的软件实现。首先对系统所需的开发环境进行搭建，简要介绍 Android 使用 FFmpeg 前的交叉编译。针对 Windows 端模块的音视频数据采集关键技术进行详细阐述；对音视频的编码进行了详细阐述与实现；对音视频的解码进行了详细阐述与实现；详细介绍了音视频同步技术，并根据系统需求，选择了恰当的方式并实现。

第六章，系统测试分析。本章以教育直播系统应用环境为出发点，设计了多组实验，根据得到的数据对系统性能进行分析。首先，测试了系统的基本功能。然后通过设计大量实验，测试了系统在时延、鲁棒、兼容性等方面的性能，并根据实验的结果对教育直播系统进行了全面的评价。

2 相关技术和原理

2.1 FFmpeg 音视频处理技术介绍

FFmpeg 是可以在多种操作系统环境下运行，提供了音视频数据处理完整的技术框架，在音视频处理领域中占有了不可忽视的地位。FFmpeg 处理音视频的能力非常强，并且支持大部分的音视频编码标准[11-13]。因此，在音视频开发工程中经常会出现它的身影。我们所熟知的如 Mplayer、暴风影音、QQ 影音、KMPlayer 等知名播放器均是以 FFmpeg 作为内核[14]。FFmpeg 除了拥有在处理音视频方面优秀的能力，它还有非常好的扩展性，可以在不同平台对 FFmpeg 进行移植，本论文在 Windows 和 Android 环境下对 FFmpeg 进行了移植。FFmpeg 主要包含如下重要模块：

AVUtil：核心工具库，其它模块经常依赖该库做一些基本的音视频处理操作，比如 av_image_fill_arrays（填充原始图像数据到 AVFrame）、av_image_get_buffer_size（根据图像宽高、格式获取填充该图像需要的字节数）、av_get_pix_fmt_name（获取像素格式的名称）等等。

AVFormat：文件格式和协议库，提供了与协议、解封装、封装相关的函数。该模块在处理音视频过程中，负责编写和读取数据信息，例如 avformat_write_header（写文件头）、av_write_trailer（写文件尾）、av_read_frame（从文件中读取一帧编码后的图像/音频数据）、av_write_frame（往文件中写一帧编码后的图像/音频数据）、av_seek_frame（给定一个时间戳，移动读指针到对应位置）等等。

AVCodec：编码和解码库。在 FFMpeg 中，默认不存在可用于编解码 H.264 格式的库以及编解码 AAC 的库，但是 FFMpeg 因其开源的特性，向使用者提供了扩展接口。例如在使用 FFMpeg 过程中，可以调用 avcodec_find_decoder 函数来查找第三方编解码库文件。

AVFilter：滤镜库。该模块提供了包括音频特效和视频特效的处理，比如把“drawbox=10:20:200:60:red@0.5”这条命令，传递给函数 avfilter_graph_parse()解析，并传递原始图像数据到该 filter 中，就能在图像坐标为(10，20)的点上生成一个宽高为(200，60)、透明度为 0.5 的红色矩形。

FFMpeg 不仅仅提供了上述文件格式和协议库、编码和解码库、核心工具库和滤镜库，还提供了与设备交互相关的库、音频重采样库以及视频像素格式转换库等[15]。

结构体也是 FFMpeg 处理数据需要用的重要变量，包括：

（1）解协议，包括 http、rtsp、RTMP、mms 等协议。

这类结构体主要用于存放协议种类以及状态，其中重要的结构体包 AVIOContext、URLProtocol 和 URLContext 等。在 FFMpeg 中，文件也被认为是一种协议类型。

（2）解封装，包括 flv、avi、rmvb、mp4 等封装格式。

解封装结构体主要包括 AVFormatContext 和 AVInputFormat。其中音视频封装格式中包含的信息存放在 AVFormatContext 中，输入数据采用封装格式 AVInputFormat 进行存储。

（3）解码，包括 h264、mpeg2、aac、mp3 等音视频编码格式。

与解码相关的结构体主要有 AVStream、AVCodecContext 以及 AVCodec。其中 AVStream 存储了音视频数据，AVCodeContext 存储了解码相关数据，AVCodec 为音视频对应的解码器[16]。

（4）存储数据。

一般每个 AVPacket 会存储一帧视频数据而音频每个 AVPacket 可能包含好几帧。AVPacket 存储编码的数据例如 H.264、AAC 等。AVFrame 存储解码后的数据例如 YUV、PCM 等。

2.2 视频编码标准 H.264 研究

2.2.1 H.264 视频编码介绍

当今视频信息的处理已迈向了数字化的领域，在使用数字化处理视频信息时，能够大幅度减少多网络环境的依赖、减轻硬件设备的负担并且在存储数字化视频信息时更加安全，视频处理本质上就是对计算机数据的处理[17]。

表 2.1 主要视频编码标准一览表

Tab. 2.1 List of main video coding standards

名称	推出机构	推出时间
H.265	MPEG/ITU-T	2013
H.264	MPEG/ITU-T	2003
MPEG-4	MPEG	2001
VP9	Google	2013

在收集了图像信息之后生成的原始视频数据具有大量数据，对于某些收集后直接播放的应用程序，无需考虑压缩技术。但是，实际上，对于更多的应用，在视频传输和存

储方面，传输网络和存储设备不能容忍大量的原始视频数据并且无法对原始视频数据进行编码和压缩，之后必须进行传输和存储。因此，统一视频编码标准非常重要。表 2.1 列出了当前视频编码技术领域的主流视频编码标准，每种视频编码标准的观点和应用领域都不尽相同。

新生的视频编码标准 H.265，包含 35 种预测模式，相较于 H.264 编码标准，视频压缩率提升了 39%到 44%，但 H.265 的高复杂度使得编码所需计算量大幅提高，对硬件性能也提出更高的要求。MPEG-4 视频编码标准在相同的视频质量前提下，视频压缩性能远差于 H.264。VP9 在网络浏览器中有更良好的支持性，被广泛应用于互联网视频网站中。鉴于不同视频编码标准的各自特点，H.26x 系列中的 H.264 视频编码标准常常用于实时流媒体视频直播系统。

H.264 视频编码标准是由 JVT 组织提出，面向未来无线领域的视频编码标准[18]。JVT 在 2001 年成立于泰国的 Pattaya，该组织由 ITU-T 的 VCEG 和 ISO/IEC 的 MPEG 大量国际标准组织的专家组成。JVT 成立之初，就以提出适合现代化的视频编码标准为目标，将压缩率、图像质量、网络依赖程度作为视频编码标准的衡量准则。H.264 已经成功被 JVT 组织获准成为 MPEG-4 标准的第十部分[19]。

2.2.2 H.264 编码原理

H.264 创新性的使用了多种压缩技术包括：整数变换、多参考帧、帧内预测等，其编码方式使用 DPCM 加变换编码相结合的混合编码方式，并且采用了环路滤波器，完善了视频编码系统的同时也大大提升了视频压缩率[20]。

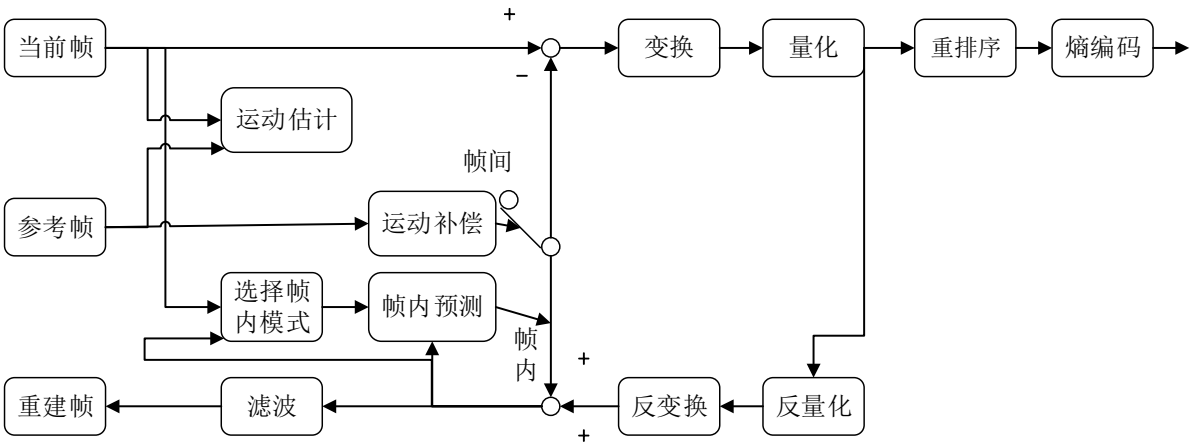

图 2.1　H.264 编码器功能组成

Fig. 2.1　H.264 encoder function composition

H.264 仅规定了在编码过程中的逻辑框架，对实现过程未做详细的要求，因此实现起来非常灵活[21]。H.264 与以前的编解码器功能模块（例如 H.261，H.263，MPEG-1 和 MPEG-4）的配置相似。不同的部分是其内部各功能模块的细节部分，H.264 编解码器的功能组成如图 2.1 所示。

H.264 编解码器的工作原理。H.264 编解码器采用转换和预测混合编码方案。在编码时，首先以宏块为单位对输入帧或字段 Fn 进行编解码处理。宏块具有两种模式，帧内和帧间。使用在当前帧内编码的宏块来预测帧内模式。帧间模式参考一个或多个先前帧进行运动预测。之后，对预测值和原始值之间的差进行转换，量化，排序，熵编码，对量化系统 X 进行逆量化，逆转换，然后将其添加到预测系统中，并且无需滤波器即可获得 uF*。对 uF*帧进行块间滤波以获得当前重建的帧 uF*。H.264 的解码过程是编码过程的逆向流程，将编码过程从尾至头执行，逆量化、逆变换。最后经过滤波器的处理后得到原始画面。

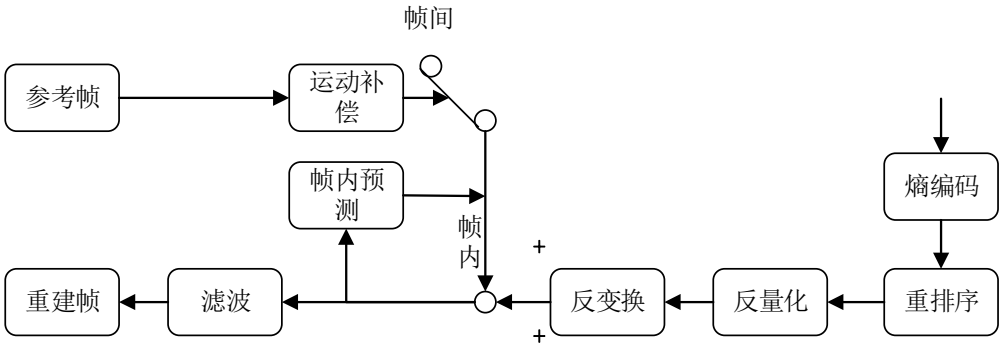

图 2.2　解码原理框架

Fig. 2.2　Decoding framework

H.264 视频编码标准中共定义了三种帧类型包括：I 帧，它是一个完全编码帧，包含了完整的画面；P 帧，需要参考 I 帧并通过与 I 帧的差异进行编码；B 帧，需要参考自身之前和之后的帧[22]。H.264 图像编码标准规定了图像的组织方式是以序列作为基本单位，每个序列都是图像经过编码后的数据信息，且序列由一个 I 帧起始，终止于下一个 I 帧，两个 I 帧之间的距离被成为 GOP。在序列中，第一个图像是 IDR 图像为立即更新图像，这些 IDR 图像都是 I 帧。编码标准定义 IDR 的意义在于任何一个序列都不会受到之前序列的影响，对于解码而言都是一个新的序列。以 IDR 图像为标定点，每当解码到一个标定点位，就代表应刷新解码队列，将存储的所有以解码的图像数据立刻处理，然后清空队列，准备开始解码新的序列。当一个图像变化很小时，就意味着一段序列会

很长，由一个 I 帧以及众多 P 帧、B 帧组成。但是当图像剧烈变化时，一个序列就会很短，仅包含一个 I 帧和几个 P 帧、B 帧。

I 帧为图像关键帧，包含了一帧图像的完整信息，解码时仅仅需要该帧数据就可以达成。

I 帧特点：

（1）将完整的一帧画面进行编码并传输

（2）单独一个 I 帧就可解码出一个完整的画面

（3）I 帧描述了图像背景和运动主体的详情

（4）I 帧在解码时，不需要参考 P 帧或者 B 帧

（5）P 帧和 B 帧参考 I 帧进行解码

（6）I 帧包含的图像信息较大

P 帧为前向预测编码帧。P 帧中保存的信息为自身和前一个帧之间的差异，前一个帧可以为 I 帧或者 P 帧。解码时需要参考前一帧的画面加上本帧所包含的差别才能生成画面。

在 P 帧进行预测时，会参考 I 帧并在所参考的 I 帧中计算出 P 帧的预测值和运动矢量。在接收方对 P 帧进行重构时，会根据运动矢量在 I 帧中找到 P 帧的预测值，然后使用差值加上预测值得到完整的 P 帧。

P 帧特点：

（1）P 帧跟随在 I 帧后传输

（2）P 帧在解码时，必须参考前面距离它最近的 I 帧或者 P 帧

（3）P 帧使用差值的方式进行传输，因此数据压缩率较高

（4）P 帧不仅可以作为身后 B 帧的参考帧，也可以是身后 P 帧的参考帧

（5）P 帧没有包含完整的图像信息，是参考帧，所以可能造成解码错误的扩散

B 帧为双向预测内插编码帧。B 帧传输的内容是该帧与前后帧的差异。在解码 B 帧时，不仅仅要得到前面的帧信息，还要的到后一帧的信息，然后通过前后帧信息与 B 帧信息的运算得到完整的画面。相较于 I 帧和 P 帧，B 帧的压缩率较高，但处理过程较为复杂，会占用大量的 CPU 资源。

在 B 帧进行预测时，会参考本帧前后的 I 帧或者 P 帧，通过计算得出 B 帧的预测值和两个运动矢量信息。接收方对 B 帧进行重构时，会根据运动矢量在 B 帧前后的参考帧中计算出预测值，之后与差值进行运算，最终得到完整的画面。

B 帧特点：

（1）B 帧中包含的信息是与前面关键帧或参考帧和后面参考帧的差异

（2）B 帧图像压缩率最高

（3）B 帧是双向预测编码帧，不会在解码时造成解码错误扩散

2.3 音频 AAC 编码研究

AAC 音频编码是感知型编码的一种，根据听众耳朵对获取到的音频信号的振幅、频率有限的分辨能力等特性，针对某些不敏感、感知不到的信号在编码量化的时候允许这些信号拥有较大的失真范围；但是对于人耳感知较为敏锐的音频成分进行较为细致的量化操作，尽最大可能的保证音频细节，这种编码方式在大大减少了压缩数据量的同时，保证了较好的音频质量。AAC 编码的具体框架如图 2.3。

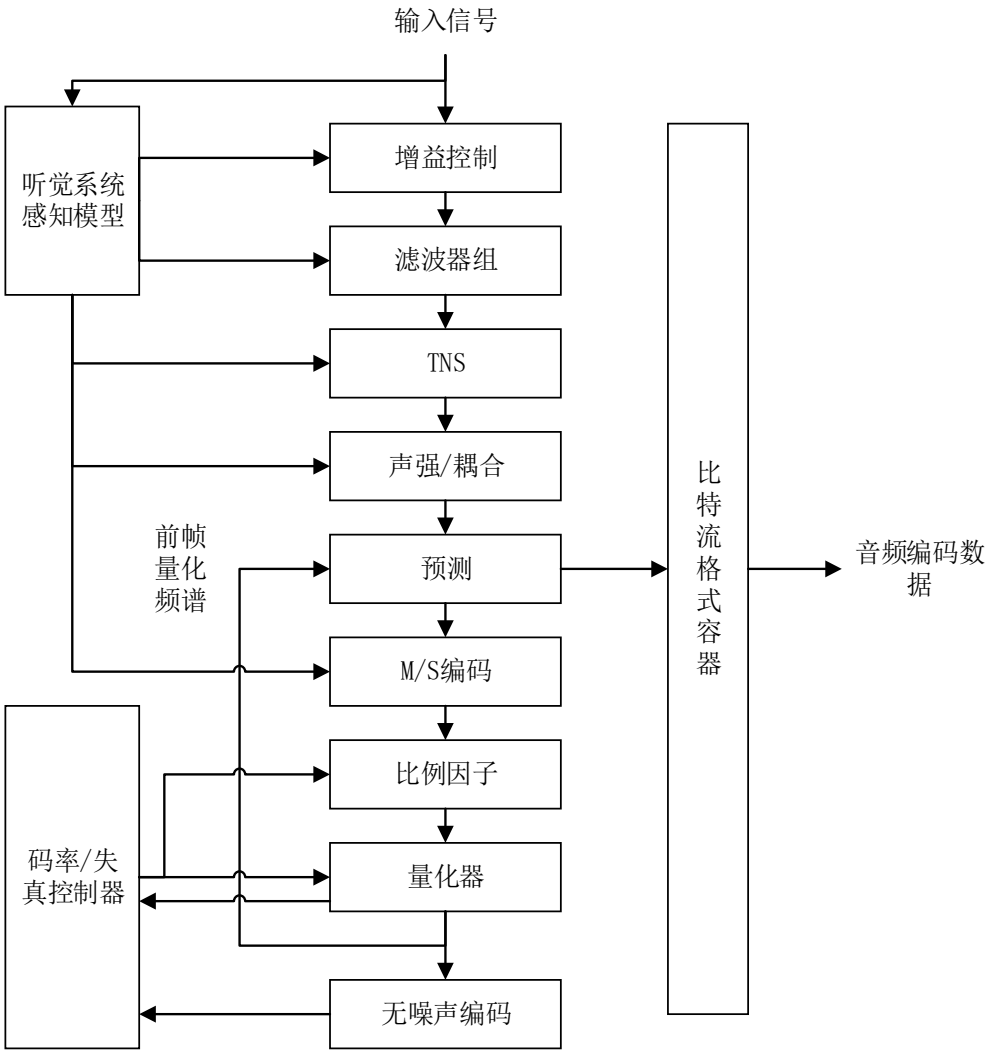

图 2.3 AAC 编码框架

Fig. 2.3 AAC coding framework

输入的音频信号首先进入到听觉感知模块，在该模块计算得出输入音频信号需要的窗类型，之后分别经过增益控制模块和滤波器组。瞬时噪声定型（Temporal Noise Shaping，TNS）在编码时需要使用的感知商以及 M/S 强度立体声需要用到的信息均由感知模块提供。感知模块也会提供输入音频信号的掩蔽阈值，通过阈值能够计算出信号的信号隐蔽比（Single to Mask Ratio，SMR）。通过 SMR 能够在量化模块中降低低频音频信号在量化过程中产生的量化噪声。

在瞬时噪声定形模块滤波处理过频谱系数之后，输入的音频信号将分别进过声强/耦合模块、预测模块以及 M/S 模块进行降噪。通过降噪可以去除音频编码的冗余信息，能够更好的压缩码流[23]。在降噪处理过程中，预测编码模块通过采用前两帧数据的频谱去预测当前的系数值的方式，去除了帧之间的频谱系数冗余，并对帧和帧之间的预测残差数值编码。在音频量化阶段，需要比例因子信息针对人耳的感知特性信息进行压缩编码，根据心理声学得到的 SMR 和掩蔽阀值，通过双侧迭代循环进行量化处理。最后编码后的频谱数据与边信息完成组帧操作，形成编码后的音频 AAC 比特流数据输出。

2.4 RTMP 协议研究

RTMP 协议是使用在应用层的协议，由 Adobe 公司研发推广，设计 RTMP 协议的目的就是用于处理在流媒体数据传输过程中的多路复用、分包问题[24-25]。VR 技术、AR 技术 UI 及移动视频直播领域的快速发展，也使得 RTMP 协议的应用范围逐步扩大，被越来越多的开发者所关注[26]。

因为 RTMP 是应用层协议，所以在传输过程中要选择安全可靠的传输层协议来确保数据可以准确送达目标计算机，一般使用 TCP 协议[27-28]。在 TCP 协议三次握手建立链接之后，RTMP 协议为了增加数据传输的可靠性，客户端与服务器之间也会进行握手来建立 RTMP Connection 链接。服务器和客户端之间会通过 RTMP Connection 链接发送控制报文，例如 SetChunkSize、SetACKWindowsSize、CreateStream 等。其中 CreateStream 会创建一个逻辑通道，该通道用于进行消息通信[29-30]。

RTMP 协议使用 RTMP Message 格式在链路中传输数据，但是在实际应用中为了更加出色的实现分包、多路复用以及信息的公平性，在发送端会把 Message 切分为 Chunk，根据原始 Message 大小的不同，每个 Chunk 可能是完整的一个 Message，也可能是一个 Message 的一部分。在接收方会根据 Chunk 中数据的大小、Chunk 包含的 Message id 信息将 Chunk 重新拼接为一个完整的 Message[31]。

RTMP 协议中定义的最基本的数据单元为消息（Message），当客户端和服务器通信传输数据时，Message 中可以包含视频、音频等信息。RTMP 协议中消息根据 Message type id 的不同，划分为不同的类别，这些 id 代表这不同的功能，具体如下。

编号为 1 到 7 的 Message type id：这些消息类型属于协议控制，一般情况下是由 RTMP 协议自己本身的管理调控过程中要使用的信息，用户不需要进行操作。

编号为 8 到 9 的 Message type id：这些消息类型是用来传输音频和视频的。

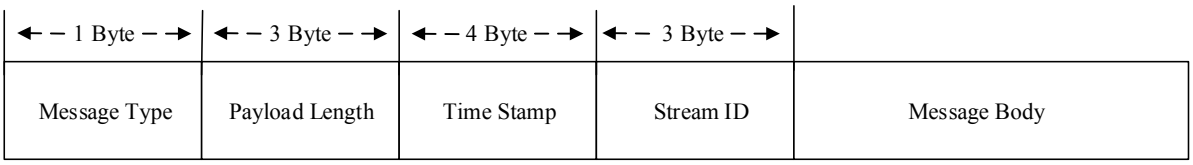

图 2.4　Message 报文结构

Fig. 2.4　Message structure

Message 报文中各部分的用途如下：

消息类型（Message type）：用于表明传递的消息类型。

负载长度（Payload Length）：表明传递信息的具体字节数，由大端模式进行表示。

消息的时间戳信息（Time Stamp）：使用大端方式进行表达，长度为 4 个字节。

每个消息的唯一标识（Stream ID）：通过此 ID 判断是否为同一消息或者消息流，使用小端方式表达，长度为 3 个字节。

消息体（Message Body）：消息传递的数据。

RTMP 协议定义了数据在传输过程中的基本单位为 Chunk。分成不同 Chunk 的目的在于将数据量大的消息切割成为一个一个的小量的消息，防止消息在传递过程中阻塞。在传输数据量较小的消息时，可以使用 Chunk Msg Header 字段进行压缩。

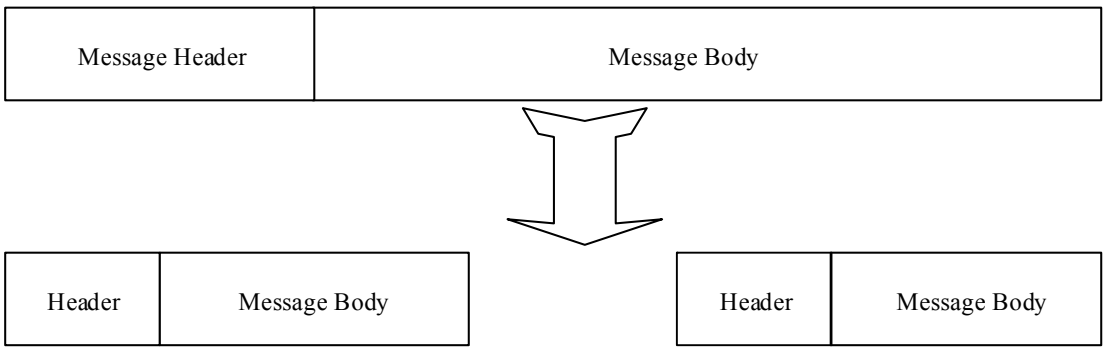

图 2.5　消息分块示意图

Fig. 2.5　Message Block

如图 2.5 在 Message 分割成为一个一个的小 Chunk 时，将 Message Body 分隔为多个 Chunk 块，这些数据块最大为 128 Byte，并在切割成为的 Chunk 头部加入相应的 Header 消息头，组合成为消息块。

要建立一个有效的 RTMP Connection 链接，首先要“握手”，如图 2.6 所示。

图 2.6 中客户端要向服务器按序发送 C0，C1，C2 三个 Chunk，服务器向客户端按序发送 S0，S1，S2 三个 Chunk，然后才能进行有效的信息传输。具体握手规则为：

（1）客户端首先向服务器发送 C0、C1，握手开始。

（2）客户端直至接收到服务器发送 S1，才能继续向服务器发送 C2。

（3）客户端需等待接收到服务器发送的 S2 后，才能发送数据。

（4）服务器在向客户端发送 S0、S1 前，必须接收到 C0。

（5）服务器仅在接收到 C1 后，才能发送 S2。

（6）服务器在接收到客户端发送的 C2 后，才能发送数据。

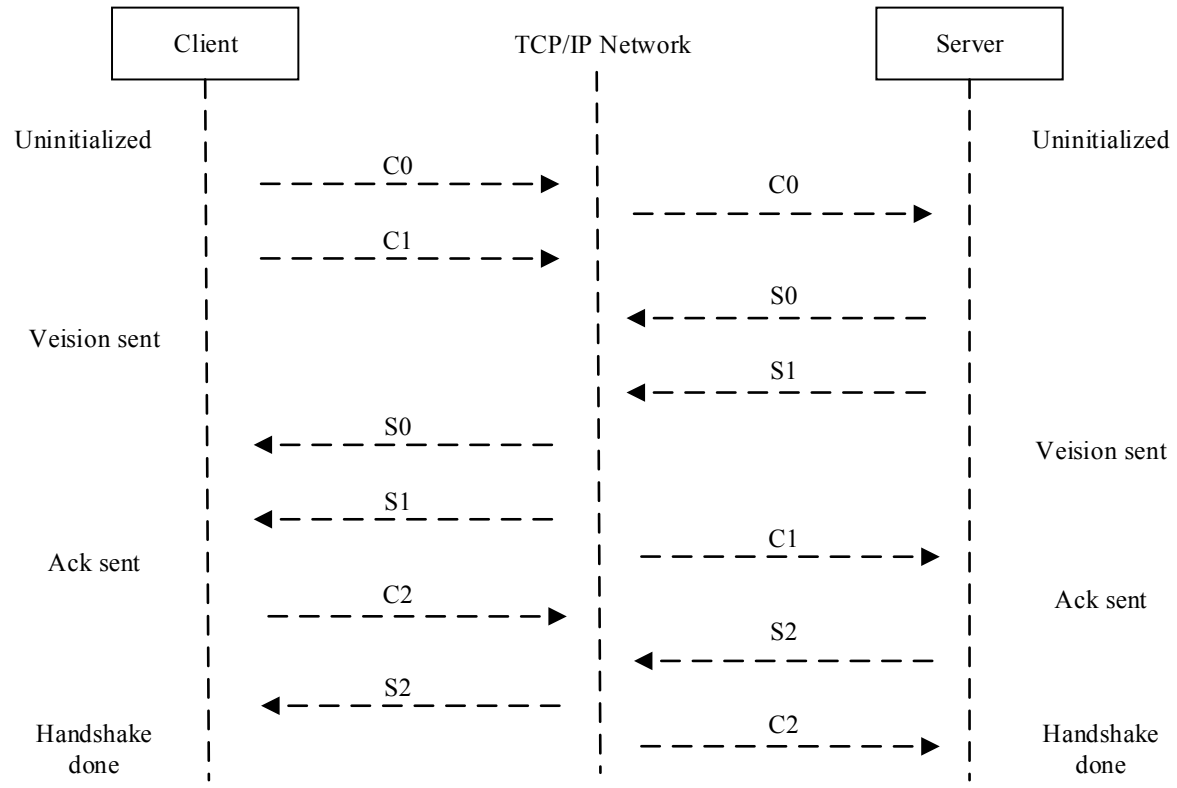

图 2.6　RTMP 链接

Fig. 2.6　RTMP Connection

在 RTMP 三次握手建立链接之后，客户端与服务器开始使用 Chunk 进行数据传输，这些生成的 Chunk 均使用唯一的 Chunk Id 进行关联，这些 Chunk 负载了来自于一个 Message 的一类消息。在传递 Chunk 过程中，Chunk 按次序发送，在当前 Chunk 发送完毕后才能准备发送下一个 Chunk。接收端根据每个 Chunk 的 Id 将它们组合成为完整的 Message。Chunk 的组成如图 2.7 所示。

图 2.7　RTMP Chunk 基本组成

Fig. 2.7　Basic composition of RTMP chunk

基本头信息（Basic Header）：Header 中包含了 Chunk 的 Id 和类型，长度为 1 到 3 字节。

块消息的消息头信息（Chunk Msg Header）：包括 0、3、7、11 四种不同的长度，信息的具体长度由基本头信息中的 chunk typq 和 fmt 决定。

扩展时间戳（Extended Time Stamp）：只有当块消息头中的时间戳设置为 0xffffff 时，本字段才被传送。。

块数据字段（Chunk Data）：存放 Chunk 传递的数据。

2.5　本章小结

在本章对教育直播系统实现过程需要涉及到的技术进行了详细介绍，包括 FFMpeg 音视频处理技术、H.264 编码、AAC 编码以及 RTMP 传输协议。对这些技术的研究，为之后系统设计和实现打下了理论基础。

3 基于 FFmpeg 的教育直播系统总体设计

3.1 教育直播系统需求分析

本论文设计的基于 FFmpeg 的教育直播系统的意义在于提高教学效率，实现教师授课时多媒体数据实时共享、教师与学生之间更加便利的分享教学文件等。针对本课题的设计目的，直播系统应具有对电脑桌面信息采集、电脑麦克风数据采集、流媒体数据转发、文件上传和下载、播放音视频数据等功能。

为实现本研究所设计的教育直播系统并达到良好的使用效果，针对系统的不同部分提出了五点功能性需求：

（1）音视频数据采集速率

PC 推流端音视频数据来源分别是电脑屏幕和电脑麦克风。在获取到原始音视频数据的过程中，数据捕获的快慢都会对本系统的工作效率和视频、音频质量造成影响，因此需要控制采集速率。

（2）视频高清编码

采集到的原始屏幕数据的数据量非常巨大，如果使用该数据进行传输，会对网络造成极大的负担，因此在采集到屏幕数据之后，应对视频数据进行编码压缩。当前视频编码领域有两种优秀的编码标准分别为：H.26x 和 MPEG-x。H.26x 编码标准表现出了在网络传输过程中的良好性能，并且 H.26x 编码标准比 MPEG-x 拥有更加高效的视频压缩率，鉴于 H.26x 编码标准的良好性能，本文使用 H.26x 编码采集到的屏幕数据。

（3）音频编码

在音频编码领域，主流的编码标准有两个：一种是 MP3 音频编码标准，另一种是 AAC 音频编码标准。经过查阅相关资料进行对比，AAC 编码相较于 MP3 编码在编码效率、高频段编码以及听觉感知质量等方便均有更良好的表现。因此，本文使用 AAC 音频编码标准编码采集到的麦克风音频数据。

（4）音视频同步技术

在直播系统中，接收数据的收流端软件在准备播放音视频数据时，如果不对音视频播放速度进行控制，就会出现音视频不一致的情况。因此，为了得到一致的播放进度，应对音视频播放进度进行调控。声音与视频的变换中，人对声音更为敏感，本文将音频播放进度作为参考，视频播放将根据音频播放进度进行调整。

（5）并发性能良好的流媒体服务器

在直播过程中，本系统服务器首要考虑的两个性能指标为网络传输低延迟以及高效的处理并发业务。因此本论文的教育直播系统选取了 Nginx 服务器，并在其上添加了 RTMP 模块。Nginx 服务器与 RTMP 传输协议的组合，不仅在处理高并发业务时的性能良好，而且更进一步地优化了系统在网络传输过程中的实时性。

3.2 系统架构

本论文设计的教育直播系统架构共分为三个部分：推流端、流媒体服务器、收流端。推流端的主要功能是采集电脑屏幕数据、采集电脑麦克风数据，将采集到的数据进行编码压缩、封装、推送至服务器。其使用 RTMP 协议在客户端与服务器之间传输数据，数据包括 H.264 编码的视频数据和 AAC 编码的音频数据。本文在服务器设备上部署了 Nginx 服务器，并在 Nginx 服务器上添加了第三方 RTMP 模块，使其支持 RTMP 流数据的接收和分发。同时，在服务器设备上添加了 Samba 服务器和 Nginx 文件下载服务，以方便教师和学生更加便利地分享教学文件。收流端包括 Windows 收流端、Android 收流端。收流端的主要功能是接收流媒体服务器分发的 RTMP 流，并对接收到的流进行解封装、解码、播放。系统框图如 3.1 图所示：

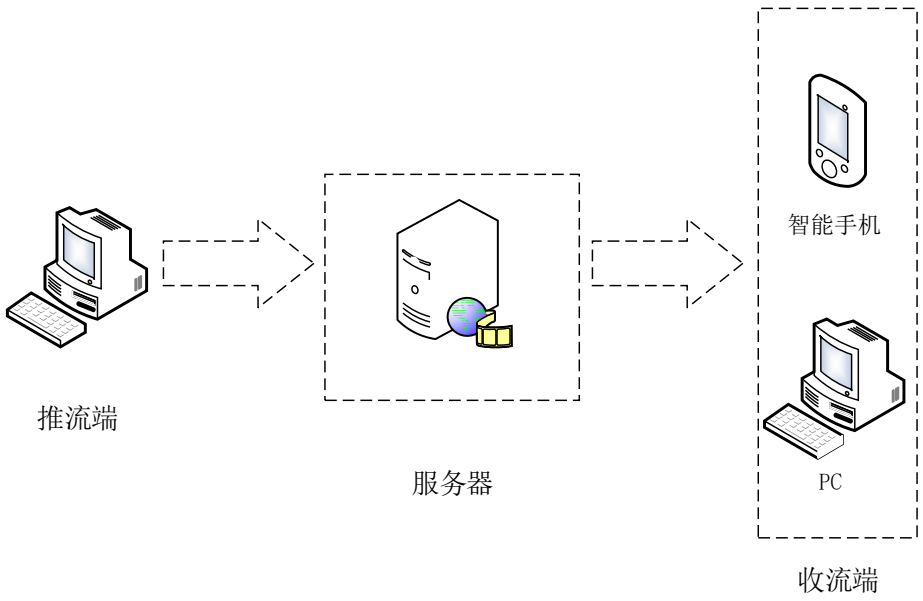

图 3.1 系统架构

Fig. 3.1 System architecture

在系统开始运行时，Windows 推流端将采集到的屏幕数据和麦克风数据编码、封装为 FLV 格式，并将封装好的数据推送至 Nginx 服务器。服务器在接收到推流端发送的数据后，将流媒体数据转发至收流端。收流端软件接收到流媒体数据，对音视频数据解封、解码并进行音视频同步，最后播放视频数据和音频数据。

3.3 硬件平台

树莓派是一款搭载 Linux 操作系统的卡片式计算机，能够为用户提供编辑文档、观看视频、绘制图片，甚至运行程序等操作。以其小巧的身形、不俗的处理能力和低廉的价格被许多研发机构所青睐，在教育、监控、物联网等领域都能看见它的身影。

图 3.2　树莓派 3B 开发板

Fig. 3.2　Raspberry Pi 3B

教育直播系统的服务器搭建在树莓派 3B 硬件平台。树莓派 3B 采用的是基于 ARM 架构的博通 4 核 1.2GHz 处理器，拥有 1GB 大小的 LPDDR2，存储方式为 SD 卡。树莓派机身支持蓝牙 4.1、WiFi 传输功能，配备 1 个以太网接口、4 个 USB 接口以及 1 个 HDMI 接口[32]。树莓派机身如图 3.2 所示。

3.4 流媒体服务器设计

流媒体服务器在系统中的主要功能是当推流端把音视频数据发送过来之后，Nginx 服务器将接收到的数据发送到收流端。在推流端软件发送数据前，要使用 URL 查找到流媒体服务器并建立链接。Nginx 等待服务器与客户端建立连接，在成功建立链接后，服务器创建监听同时调度 Worker 进程等待推流端数据的到来。直播系统工作过程中，Nginx 服务器会将推流端发送的流媒体数据转发到收流端。在系统停止工作后，Nginx 服务器接收到结束消息后，停止 Worker 进程。

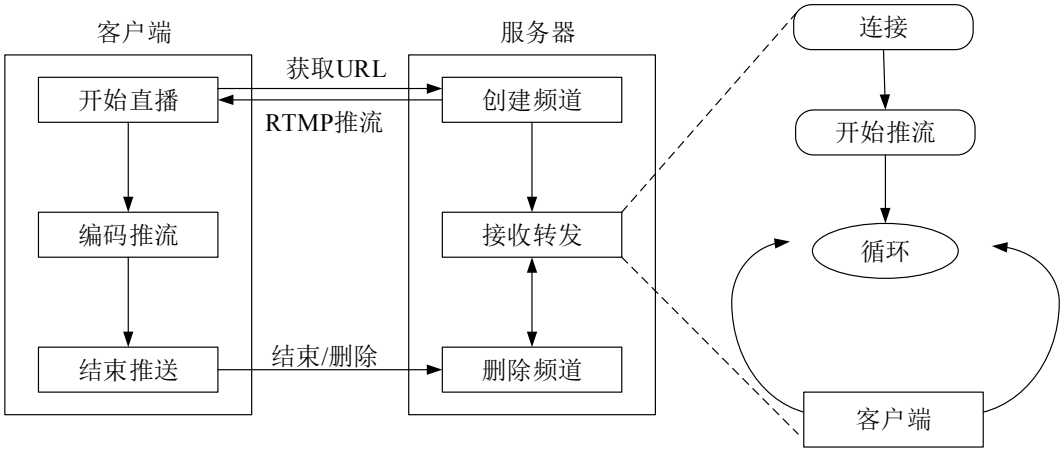

图 3.3　服务器端工作流程

Fig. 3.3　Server-side workflow

3.5　客户端软件设计

本文设计的教育直播系统软件部分由推流端软件和收流端软件两部分组成。推流端软件主要负责采集屏幕数据和麦克风数据，之后将采集到的原始屏幕数据和原始音频频数据分别编码为 H.264 格式和 AAC 格式，因为 RTMP 协议自身对传输内容格式的要求为 FLV 格式，所以编码压缩后的数据要封装为 FLV 格式后，使用 RTMP 协议传输至服务器。收流端软件主要负责接收经由流媒体服务器转发的音视频数据，接收到后将 FLV 封装格式的数据解封装并对 H.264 视频编码格式解码和 AAC 音频编码格式数据解码，之后把解码出来的视频数据和音频数据进行同步处理，最后播放音视频。软件设计流程图如图 3.4 所示，其中图（a）为推流端软件，（b）为收流端软件。

本文在设计推流端软件时，按照功能的不同将软件划分为数据采集、编码、封装等多个模块，并为每个模块开辟一个线程独立运行，模块与模块之间采用临界区的方式进行通信。在采集数据过程中，本论文设计了数据缓存区，有效的提高了在编码过程中的稳定和效率。最后，通过 RTMP 协议将数据传输给服务器。

Windows 客户端收流软件和 Android 客户端收流软件均使用 FFmpeg 处理数据。与推流端软件类似，本文设计收流端软件时，同样按照功能的不同将收流端软件划分为解码、同步、播放等多个模块，每个模块开辟一个子线程独立运行，线程之间的通信使用临界区的方式。在对音视频进行同步之后，进行音视频数据的播放。

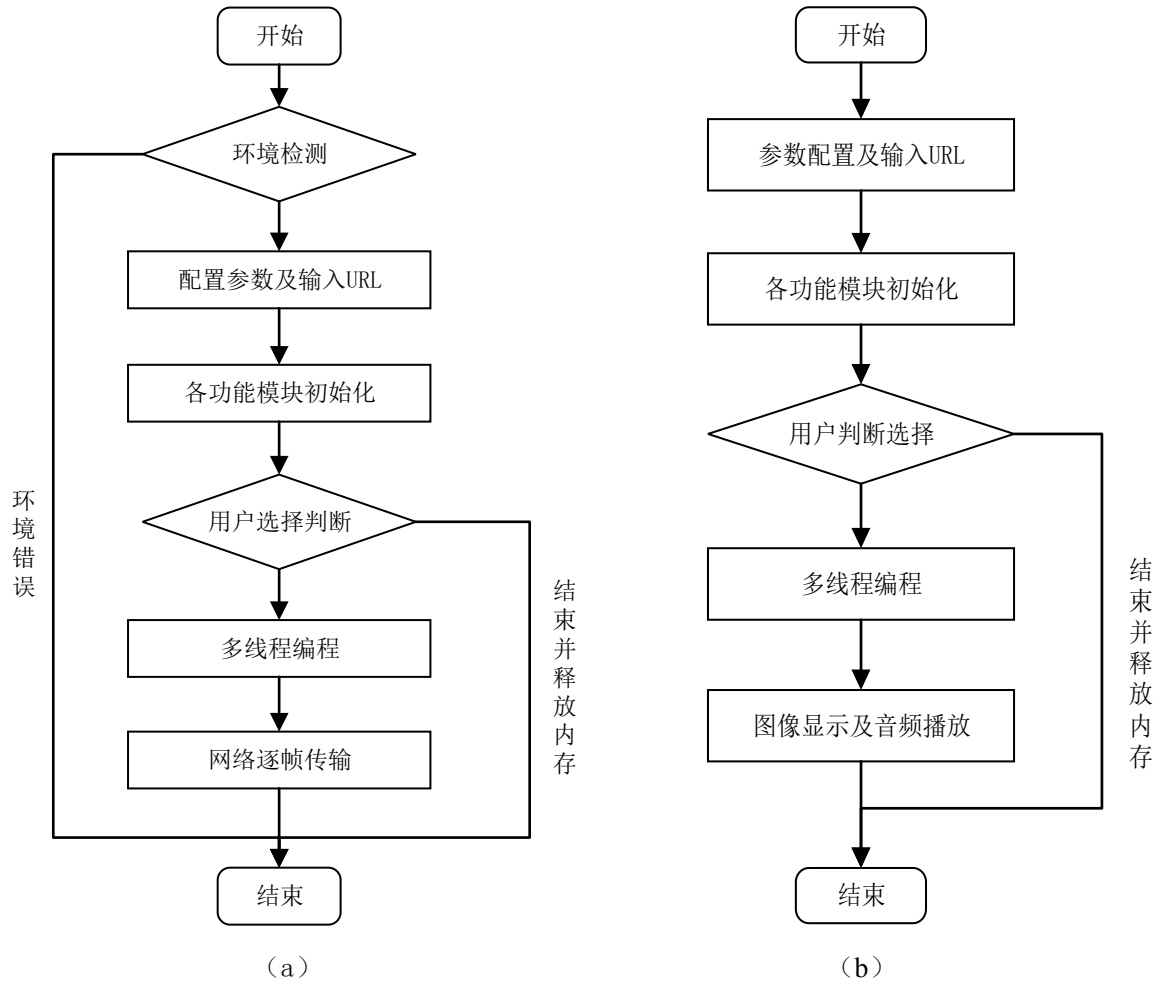

图 3.4　软件流程框图

Fig. 3.4　Software flow chart

3.6　本章小结

本章首先针对教室环境下的实际应用场景，提出了五点功能性的需求，并以系统需求为首要考虑因素，选择部署服务器的硬件设备、设计流媒体服务器应具有的基本功能、设计 Windows 客户端和 Android 客户端的基本架构以及程序工作流程，为后续章节提供了实现指向。

4 教育直播服务器实现

4.1 基于 Nginx 的流媒体服务器实现

在教育直播过程中，通常会面对教室中一位主讲老师，多位学生的情况。在这种情况下，系统服务器要保持良好的数据转发处理能力。本论文选择 Nginx 部署在树莓派 3B 上并作为本系统流媒体服务器的理由是 Nginx 在面对高并发时依旧可以保持优秀的数据转发能力。Nginx 服务器工作性能稳定、内存占用率低且功能丰富可扩展的服务器[33-34]。

Nginx 是俄罗斯人 Igor Sysoev 使用 C 语言发开的开源 web 服务器，在支持高并发链接的情况下，还能拥有性能良好的稳定性。Nginx 支持添加第三方 RTMP 模块，Nginx 主要功能包括：

（1） 支持音视频直播
（2） 支持 flv/mp4 封装格式
（3） 支持两种不同的流分发模式：push 流、pull 流
（4） 能够将直播流保存为 flv 文件
（5） 支持在线转码
（6） 相同条件下，占用较少内存
（7） 能够与 FFmpeg 协同工作
（8） 支持添加外部模块
（9）支持 RTMP 协议

4.1.1 高并发的重要性

互联网兴起的时代，由于网络用户的数量爆炸式的增多，服务器架构开始面对如何妥善处理并发问题。最初，并发问题主要体现在客户端与服务器之间的建立链接的速度较慢，如今，并发问题转变为体量巨大的视频数据、音频数据、文字数据以及图片数据占据了大量的服务器存储空间。面对这样的情况，绝大部分网站使用分离资源存储的方式去解决存储空间占用的问题。然而，随着移动互联网的兴起，移动端以其持久的链接更新等服务对后端服务器提出了更加严苛的需求[35]。

在面对多用户请求、高并发问题时，Nginx 的设计理念不同于传统的基于线程或者进程模型，而是基于事件的模型。这种设计理念的优势在于，面对高并发业务时，Nginx 不会为服务开启新的进程或者线程，这样不会因为频繁地切换或者创建新的进程或者线程而占用过多的 CPU 资源。Nginx 在接收到各个服务请求时，会判断服务类型，通过服

务类型将任务分发到相应的进程处理，这样 Nginx 就能够保持优秀处理能力的同时降低对硬件的需求。

4.1.2 Nginx 服务器架构分析

Nginx 是模块化设计的服务器，这种架构方式在不触碰服务器核心代码的前提下，给予了开发者能够通过添加模块的方式，自由开发服务器的权限。Nginx 的核心组成包括：event 模块、协议模块、负载均衡等等，每个模块的代码在编译时一同进行编译。在 Nginx 核心模块中，封装并提供了内存池操作、网络链接、锁等操作。Nginx 服务器的架构如图 4.1 所示：

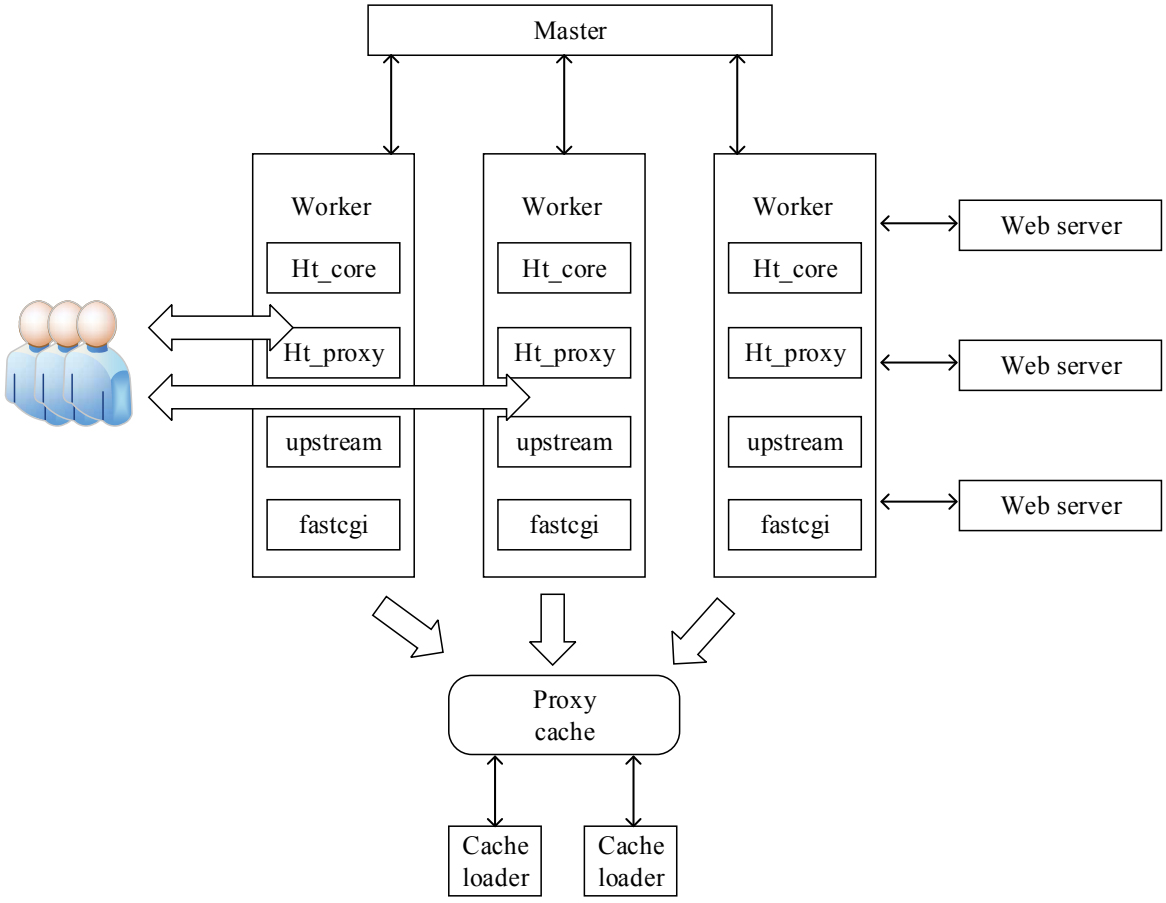

图 4.1 Nginx 服务器架构设计

Fig. 4.1 Architecture design of Nginx server

在 Nginx 服务器启动初始会完成对监听套接字的初始化操作。Nginx 在运行时主要有两类进程负责处理事务分别为 Master 主进程以及 Worker 工作进程。Master 主进程在

Nginx 工作中负责接收外部信号的请求，并将请求分发到 Worker 进程中，由 Worker 进程处理请求。

4.1.3 Nginx 服务器的配置文件

传统的 web 服务器在日常维护配置文件时常常要消耗大量的时间，针对这种问题，Nginx 以简化操作为出发点，将自身的配置文件 nginx.conf 设计成为 C 语言风格，使用者可以通过简便的操作，就能实现对 Nginx 服务器的拓展。

nginx.conf 文件支持正则表达式，配置文件中，主要由以下几个区域组成：main 区、event 区、HTTP 区、server 区以及 location 区等等。Nginx.conf 配置文件的各区域说明如图 4.2 所示：

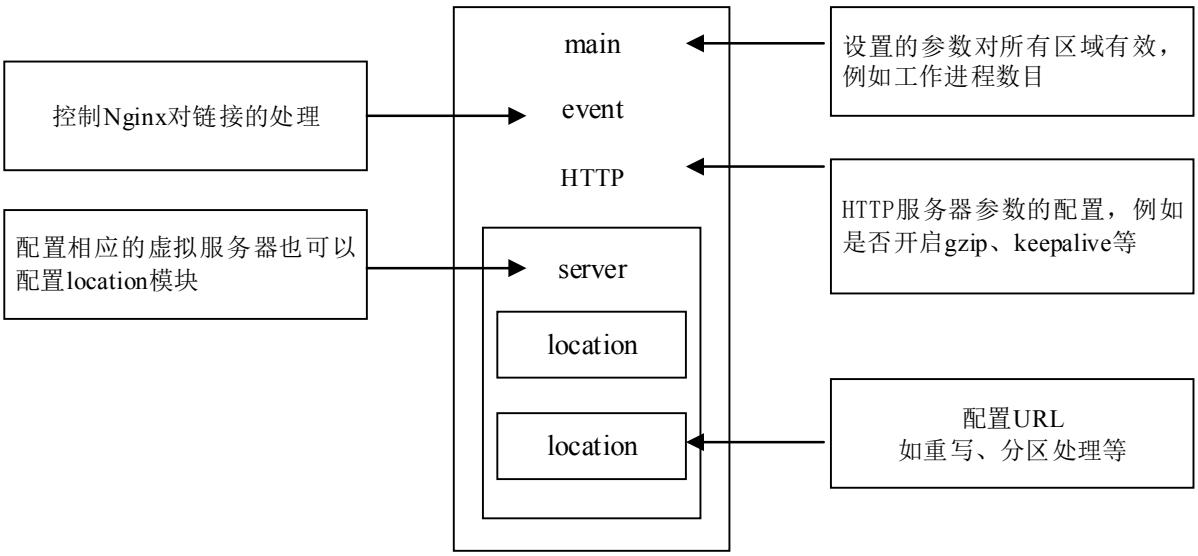

图 4.2 Nginx 配置文件说明

Fig. 4.2 Nginx profile description

4.1.4 Nginx 服务器实现

本文统选取 Nginx 服务器为教育直播系统的流媒体服务器，并在其上部署了 RTMP 模块以实现流媒体数据的转发。对 Nginx+RTMP 服务器的主要搭建步骤为：

（1）下载 Nginx 压缩包以及 RTMP 压缩包并安装，所需文件如图 4.3 所示：

（2）配置 Nginx 安装选项并编译 Nginx。

在 Nginx 默认配置的基础上，使用--add-module 指令添加 RTMP 第三方模块。

（3）启动 Nginx。

使用./nginx 命令启动 Nginx，服务器欢迎界面如图 4.4 所示。

```
drwxr-xr-x. 9 1001 1001   4096 Jan  1  1970 nginx-1.10.0
-rw-r--r--. 1 root root 908954 Sep  7  2016 nginx-1.10.0.tar.gz
-rwxr-xr-x. 1 root root   2725 Jan  1  1970 nginx-install.sh
drwxr-xr-x. 6 root root   4096 Aug  9  2016 nginx-rtmp-module-master
-rw-r--r--. 1 root root 545752 Sep  7  2016 nginx-rtmp-module-master.zip
```

图 4.3　模块安装包

Fig. 4.3　Module installation package

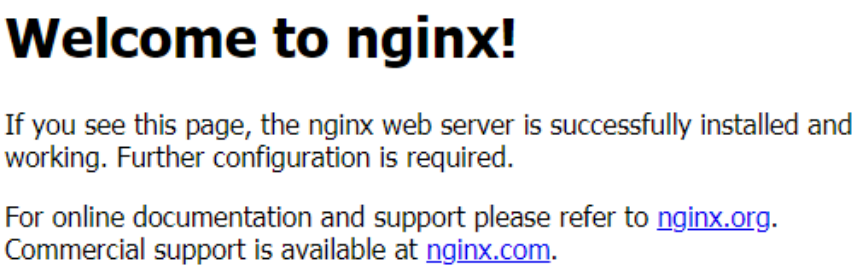

图 4.4　欢迎界面

Fig. 4.4　Welcome Interface

4.1.5　直播配置

Nginx 配置文件中默认的最大并发量为 1024，可以通过 worker_connections 指令修改 Nginx 服务最大并发数量。但是并发量的增多，会增加服务器 CPU 负载。针对本论文所研究的教育直播系统应用场景，未对最大并发量进行修改。

对于直播服务，需要在服务器配置文件中 RTMP 区域的 server 区域添加应用名，本文将应用名称设置为 live，并将监听端口设置为 1936。再次启动 Nginx 服务器，并在浏览器输入查看 RTMP 状态地址，看到如图 4.5 所示的 RTMP 状态监控图，证明配置已经生效。

RTMP	#clients	Video				Audio				In bytes	Out bytes	In bits/s	Out bits/s	State	Time
Accepted: 0		codec	bits/s	size	fps	codec	bits/s	freq	chan	0 KB	0 KB	0 Kb/s	0 Kb/s		2m 15s
live															
live streams	0														
hls															
live streams	0														

图 4.5　服务器状态监控

Fig. 4.5　Server status monitoring

4.2　文件传递设计与实现

教育直播系统设计目的是解决教学过程中信息传递的问题，在教师授课过程中，经常会遇到需要向同学发送 PPT、Word 等文档的情况。针对教师与学生之间传递文件繁

琐的问题，本论文设计了 Nginx 文件下载功能与 Samba 文件共享服务器相结合的方式，简化了教师向学生传递文件、学生获取教学文件的过程，极大地缩短了授课过程中的文件传递时间。

4.2.1 Nginx 服务器文件下载功能实现

本论文使用 Nginx 服务器提供的下载功能模块，避免在系统服务器运行过多进程，造成资源浪费。

Nginx 下载模块具体实现步骤为：

（1）参数配置

创建一个 down.conf 配置文件，该文件用于编写实现文件下载功能 server 模块代码。具体代码配置为：

```
server {
        listen          80;
        server_name    down.mmcl.cn;
        root      /home/mmcl/Nginx/html/down;
        autoindex          on;
        autoindex_exact_size      off;
        autoindex_localtime       on;              }
```

其中/home/mmcl/Nginx/html/down 文件夹是为需要下载的文档指定的保存在服务器上的文件夹，autoindex 表示开起索引，autoindex_exact_size 表示用 kb、mb、gb 表示文件大小，autoindex_localtime 表示显示本地时间。

（2）Nginx 配置文件引入调用

在 Nginx 的配置文件 Nginx.conf 的 http 模块中引入 down.conf 文件。文件具体配置如下。

```
http {
...
     Include      /home/mmcl/Nginx/conf/down.conf;
}
```

（3）访问文件浏览页面

在浏览器地址栏输入 http://localhost.cn，出现文件浏览页面，点击下载即可。

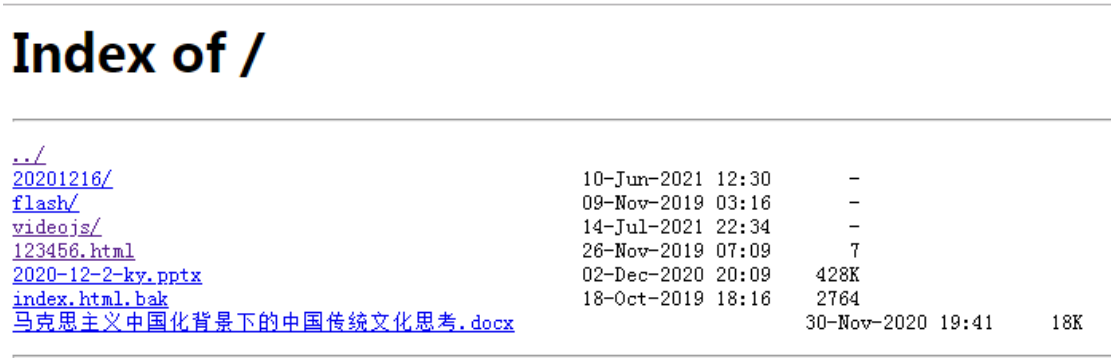

图 4.6　文件下载页面

Fig. 4.6　File download page

4.2.2　Samba 文件共享服务

教育直播系统使用过程中，教师需要和服务器进行文件交互时，为了便于教师对服务器文件夹进行操作，本论文在系统的树莓派 3B 服务器上部署了 Samba 服务。Samba 是用来实现 SMB 协议的软件。SMB 协议是一个网络文件共享协议，主要应用在同一局域网内的终端之间。Samba 工作的基本流程为：

（1）SMB 类型协商

客户端准备使用 Samba 服务器的共享资源前，需要和服务器进行协议版本认证。认证流程为客户端首先发送自身能够使用的协议版本，Samba 在接收到数据包后，相应请求并回复通信要使用的协议版本。如果服务器支持的 SMB 协议版本与客户端不相匹配，则返回 0XFFFFH 表示结束通信。

（2）链接建立

在确定 SMB 协议版本后，客户端会向 Samba 服务器发送携带身份信息的数据包。服务器对接收到的信息进行认证，并返回一个应答数据包来允许或者拒绝本次链接请求。SMB 协议的客户端认证方式共有四种包括：

（a）Share：不需要输入用户名和密码，这台 Samba 服务器的文件资源可以被局域网内所有计算机访问。

（b）User：需要这台 Samba 服务器对申请访问资源的客户端进行身份验证。身份验证的内容包括 username 和 password。

（c）Server：需要另外一台 Samba 服务器对申请访问资源的客户端进行身份验证。身份验证的内容为 username 和 password。

（d）Domain：客户端身份验证环节交由域控制器执行。

（3）访问资源

当客户端与服务器完成（1）、（2）步骤的磋商和认证之后，客户端会向服务器发送自己想要访问的资源名称，之后服务器会发送一个应答数据包回应此次资源访问是否被允许。

本论文以教师使用便利文件存储安全为首要考虑目标，选择 User 认证模式，为教师提供安全可靠的用户名和密码。

4.3 推流鉴权实现

在设计教育直播系统的过程中，针对不良用户将恶意信息推送至流媒体服务器，影响教学的问题，本论文采用了推流鉴权方案，有效地防止了恶意信息流入课堂的问题。

Nginx 推流鉴权主要与 on_publish 指令有关，在系统运行开始阶段，推流端向服务器申请建立链接并发送数据，服务器接收到推流端的请求后，将推流端携带的身份信息传递给 on_publish 指定的地址。该地址一般指向一个信息处理脚本文件，文件的主要作用是进行推流用户的身份信息校验，在校验完成后会对 Nginx 服务器返回一个状态码作为校验结果，其中 2xx 代表验证成功，允许该推流端向服务器推送信息；3xx 代表地址重定向，代表将推流端推送的信息转发到其他地址；若返回其他的状态码，则代表拒绝该推流端推送流媒体数据。具体流程如图 4.7 所示。

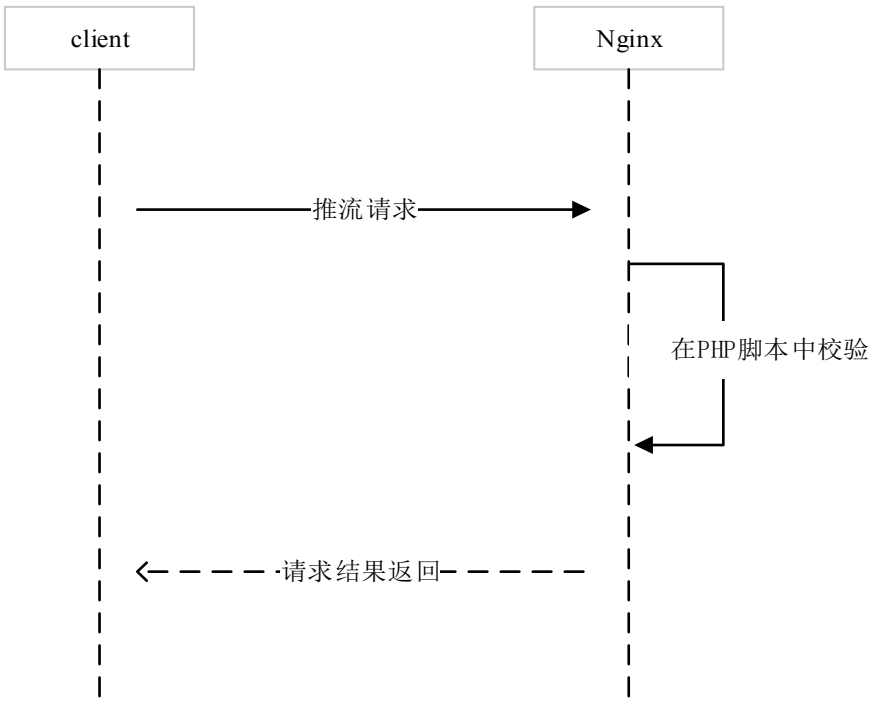

图 4.7 Nginx 鉴权流程图

Fig. 4.7 Nginx authentication flow chart

本文 Nginx 鉴权实现流程为：

（1）编写身份校验 php 脚本。脚本首先获取推流端携带的 ip 和 password 身份校验信息，然后根据脚本内存储的合法身份信息对获取到的信息进行校验。如果校验信息合格，则向 Nginx 服务器返回 200 状态码，代表推流端身份信息正确，允许向 Nginx 服务器推送数据。如果校验信息不合格，则向 Nginx 服务器返回 404 状态码，代表推流端身份信息错误，禁止向 Nginx 服务器推送数据。脚本校验身份信息的流程框图如图 4.8 所示。

（2）修改 Nginx.conf 文件。在 rtmp 模块的 server 部分增加 on_publish 指令，并将 on_publish 的校验地址指向步骤（1）中编写的 php 脚本文件。并在 Nginx.conf 文件中增加 php 模块，设置 Nginx 与 php-fpm 间通信套接字为 127.0.0.1:9000，同时修改 php-fpm.conf 文件，将 listen 监听同样设置为 127.0.0.1:9000。

（3）重新启动 Nginx 和 php-fpm。

推流端分别使用正确和错误的 id、password 进行校验，通过图 4.9Nginx 服务器 access.log 日志可以看到，推流鉴权功能成功实现。

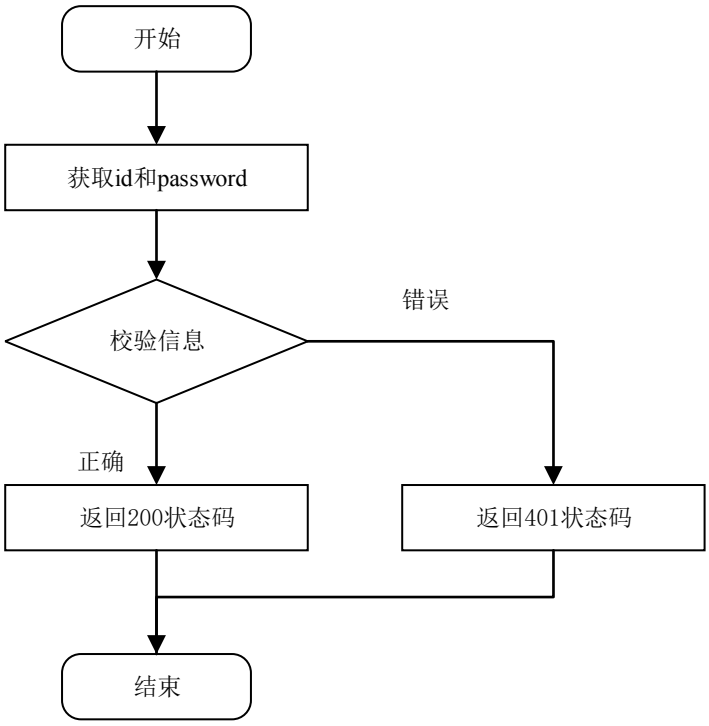

图 4. 8　鉴权脚本框图

Fig. 4.8　Authentication script block diagram

图 4.9　服务器日志

Fig. 4.9　Server log

4.4　本章小结

本章主要完成对教育直播系统服务器的实现。本章首先详细阐述了 Nginx 服务器的架构和配置文件，之后对 Nginx 服务器进行了实现，并在其上部署了 RTMP 模块，最后针对系统使用中出现的问题提出了解决方案，包括 Samba 服务加 Nginx 文件下载功能结合的文件传递方式、Nginx 鉴权功能防止恶意信息流入等。

5 教育直播系统软件实现

本论文设计的教育直播系统主要由服务器和软件两部分组成，其中软件分为推流端软件和收流端软件两种。Windows 端主要包括推流端和收流端的功能，Android 端软件仅实现收流端功能。本章重点介绍软件重点功能的实现，以及针对音视频播放不同步、收流端音视频打开时间慢等问题提出的解决方案。

5.1 软件实现平台搭建

5.1.1 PC 端开发环境搭建

基于 FFmpeg 的教育直播系统的 PC 端开发系统为 Windows 7，FFmpeg4.0.2 版本，集成开发环境为 Visual Studio 2010。VS2010 下的 FFmpeg 快速配置的具体流程为：

（1）在 FFmpeg 官方网站下载 4.0.2 版本。

FFmpeg 共包括三个版本：Static、Shared 以及 Dev。FFMpeg 的 Static 版本在编译过程中就已经将需要用到的库文件编译到可执行文件中，因此这个版本的每个可执行文件都很大，Static 包含的可执行文件包括：FFMpeg.exe，ffprobe.exe，ffplay.exe。FFMpeg 的 Shared 版本同 Static 版本不同，在编译过程中未将需要用到的库文件编译进可执行文件当中，而是在运行时调用需要的库文件，因此 Shared 版本的可执行文件体积都很小，除了含有可执行文件之外，Shared 版本还带有库文件，例如 avcodec-54.dll 等。FFMpeg 的 Dev 版本是专门为开发者提供的版本，这个版本仅包含了开发 FFMpeg 项目所需要使用到的库文件和头文件，不提供可执行文件。

本文选择下载的版本为 Dev 和 Shared 版本。相较于 Static 版本，Shared 版本的 bin 文件夹较小一些，其中包含了我们需要用到的 FFMpeg.exe，ffplay.exe 以及 ffprobe.exe。

（2）配置所需 FFmpeg 文件

将从官网保存至本地的 Shared 版本和 Dev 版本压缩包解压，在解压后的文件中分别将 Shared 版本中的 bin 文件夹以及 Dev 版本中的 include、lib 文件夹拷贝至新建文件夹中。

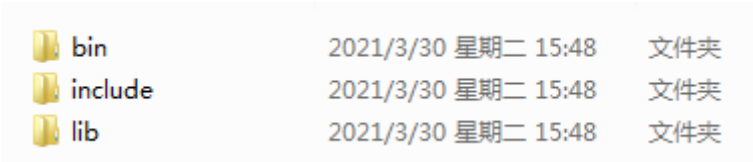

图 5.1 FFmpeg 关键文件夹

Fig. 5.1 FFmpeg key folders

（3）计算机 Path 变量配置

打开计算机的环境变量窗口，查找并编辑 Path，在 Path 中添加 bin 文件夹的绝对路径，要使用“;”将 bin 文件夹路径与前面的路径变量分隔。

（4）配置 VS2010 下的 FFMpeg 开发环境

新建 Win32 控制台工程，在工程的属性管理器中创建新的属性表并命名为 FFMpeg。打开并编辑 FFMpeg 属性表的 include 目录和 lib 目录并添加步骤（2）配置好的 include 和 lib 文件夹。最终结果如图 5.2 所示：

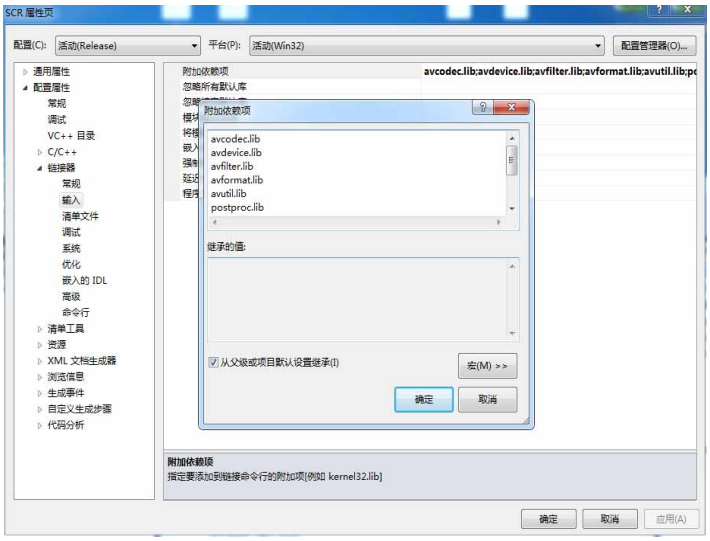

图 5.2　设置附加依赖项

Fig. 5.2　Set additional dependencies

至此，VS2010 下 FFmpeg 开发环境配置完成。

5.1.2　Android 端开发环境搭建

Android 端开发有两种开发工具，分别是 Eclipse 和 Android studio。Eclipse 集成了 ADT（Android Development Tools）插件，Android studio 是 Android 官方提供的集成开发环境[36]。Android studio 相较于 Eclipse 具有以下优势：

（1）集成 Android 软件开发包。

（2）提供了更为便捷的图形布局设计方式。

（3）在文本编辑方面，相较于 Eclipse，Android studio 更加完善和智能。

（4）Android studio 提供了 Gradle 构建工具，该工具同时具备了 Ant 与 Maven 的功能。

（5）Android studio 为用户提供了性能分析工具，让用户可以更直观地分析项目性能等。

（6）Android studio 配备了内置终端，让用户在开发过程中无需打开新的命令行就可以执行操作。

根据上述 Android studio 所具有的优势，本文选择 Android studio 作为教育直播系统 Android 客户端的开发环境。

不同于在 Windows 端使用 FFmpeg，FFmpeg 并未提供可以直接在 Android studio 上使用的 FFmpeg 版本。因此，在 Android 环境下使用 FFmpeg 需要先进行交叉编译。本地编译是指在平台编译出本平台的可执行文件，而交叉编译恰恰相反是指因本平台的系统硬件性能较差，所以需要借助其他平台去编译本平台的代码并生成可执行文件。可以说，嵌入式系统的进步带动了交叉编译的发展，因为嵌入式系统的处理能力、内存等均有限，所以有时候需要在其他平台上编译好后导入嵌入式系统中[37]，此时就需要交叉编译。交叉编译最主要的是环境，即交叉编译链；对本文来说，编译 FFmpeg 需要准备 NDK，NDK 中提供了交叉编译链。本文选用的 FFmpeg 版本是 FFmpeg 4.0.1，NDK 版本是 Android-ndk-r15c，交叉编译的平台是 Linux 系统。具体步骤如下：

（1）在进行交叉编译之前，需要进入到系统环境变量文件夹中，向文件夹中添加 NDK 路径。

（2）交叉编译得到的 FFMpeg 库文件的默认文件后缀为.so.FFMpeg 版本，为了得到 Android 项目支持的.so 文件后缀，本文提前对 FFMpeg 配置文件进行了修改，修改其生成库文件名的格式;编辑 FFmpeg 目录下的 configure 文件，修改如下：

```
# 将configure文件中的:
SLIBNAME_WITH_MAJOR='$(SLIBNAME).$(LIBMAJOR)'
LIB_INSTALL_EXTRA_CMD='$$(RANLIB) "$(LIBDIR)/$(LIBNAME)"'
SLIB_INSTALL_NAME='$(SLIBNAME_WITH_VERSION)'
SLIB_INSTALL_LINKS='$(SLIBNAME_WITH_MAJOR) $(SLIBNAME)'

#替换为:
SLIBNAME_WITH_MAJOR='$(SLIBPREF)$(FULLNAME)-$(LIBMAJOR)$(SLIBSUF)'
LIB_INSTALL_EXTRA_CMD='$$(RANLIB) "$(LIBDIR)/$(LIBNAME)"'
SLIB_INSTALL_NAME='$(SLIBNAME_WITH_MAJOR)'
SLIB_INSTALL_LINKS='$(SLIBNAME)'
```

图 5.3　FFmpeg 文件修改

Fig. 5.3　FFmpeg file modification

为了减小 APK 大小，仅选择系统需要的功能。本论文编写了一份脚本文件 build-script.sh，为脚本开启用户可执行权限并运行脚本。

至此，本研究需要的 Android FFmpeg 库文件已交叉编译成功。

在 FFmpeg 开始接入之前，需要预先配置 Android studio 环境。主要是在 SDK Manager 中下载并配置 CMake 以及与交叉编译版本相一致的 NDK。将交叉编译生成的.so 文件拷贝到 libs 目录下，同时把 include 里面的头文件拷贝到 libs 目录下。并在项目的 build.gradle 文件 defaultConfig 这个范围中配置 sourceSets。最后在项目文件 CMakeLists.txt 中配置 add_library、set_target_properties 和 target_link_libraries。

5.2 数据采集

本文使用 FFMpeg 的 libavdevice 捕获计算机的屏幕数据以及麦克风数据，然后将数据输出到指定的结构体对象中。本系统数据输出端口是编码器的入口，通过程序循序读入设备音视频数据，将捕获的原始数据交递给编码器模块，完成音视频数据的采集功能。

在采集数据前，需要获取到计算机屏幕和麦克风设备的名称。本论文在使用 FFmpeg.exe 在命令行模式下获取 DirectShow 的设备信息[38]。

图 5.4 获取设备名称

Fig. 5.4 Get device name

因为命令行编码的原因，在出现中文名称时会发生乱码的现象。使用 chcp 65001 命令，将编码设置为支持中文的 UTF-8 代码页。

在本论文代码中，音视频均采用 dshow 输入设备格式进行捕获。屏幕采集设备名称为：screen-capture-recorder；音频采集设备需要对获取到的设备名称转换为 UTF-8 类型，使用的类型转换方法为调用 Win32 SDK 下的 WideCharToMultiByte()函数，核心代码为：

```
char*    to_utf8(wchar_t* buffer){
    char* str=NULL;
    int maxlen=WideCharToMultiByte(CP_UTF8，0，buffer，-1，0，0，0，0);
    str=(char*)av_malloc(maxlen);
    if(str)
        WideCharToMultiByte(CP_UTF8，0，buffer，-1，str，maxlen，0，0);
    return str; };
```

将转换后的麦克风设备名称保存在 char 类型指针中：

char* psDevName=dup_wchar_to_utf8(L"audio=麦克风(High Definition Audio 设备)");

5.3 音视频编码

在流媒体数据传输过程中，如果采用未经编码过的 YUV 或 PCM 数据，不仅会对网络带宽造成很沉重的负担，而且也会增加系统存储的压力。因此，为了提高教育直播系统的运行效率，对采集到的原始音视频数据进行编码是重要的实现环节[39]。

常用的视频编码标准包括 H.26x 系列标准和 MPEG 系列标准，相较于 MPEG 系列编码标准，H.26x 有着良好的压缩性能。因此本文选取了 H.26x 系列中的 H.264 视频编码标准作为推流端软件的视频编码格式。H.264 规定了数据单位为 NALU（Network Abstraction Layer Unit，网络抽象层单元）。NALU 结构如图 5.5 所示。

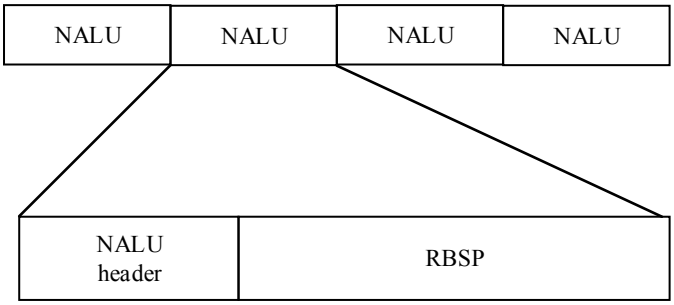

图 5.5 NALU 结构图

Fig 5.5 Nalu structure chart

其中 NALU 头信息为一个字节，存放 NALU 的类型和优先级，RBSP 存放的是与编码相关的信息包括视频压缩数据和参数集等信息

目前，主流的音频编码标准包括 MP3 标准和 AAC 标准。AAC 标准具有更高效的音频压缩率，以及在高压缩率的同时还能保证优秀的音质，因此本文选择 AAC 音频编码标准作为推流软软件的音频编码格式。AAC 编码标准共有两种不同的格式分别为 ADIF（音频数据交换格式）和 ADTS（音频数据传输流）。ADTS 更适合用于在流媒体传输，在 ADTS 结构中包含一个 syncword，在直播系统里对数据处理更加便捷。ADTS 数据结构如图 5.6 所示,其中每一个同步字 syncword 之后都会包含一个头信息、误差校验信息和数据块信息。

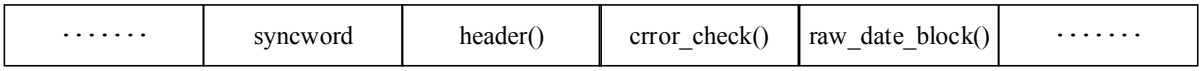

图 5.6　ADTS 结构图

Fig. 5.6　ADTS struct chart

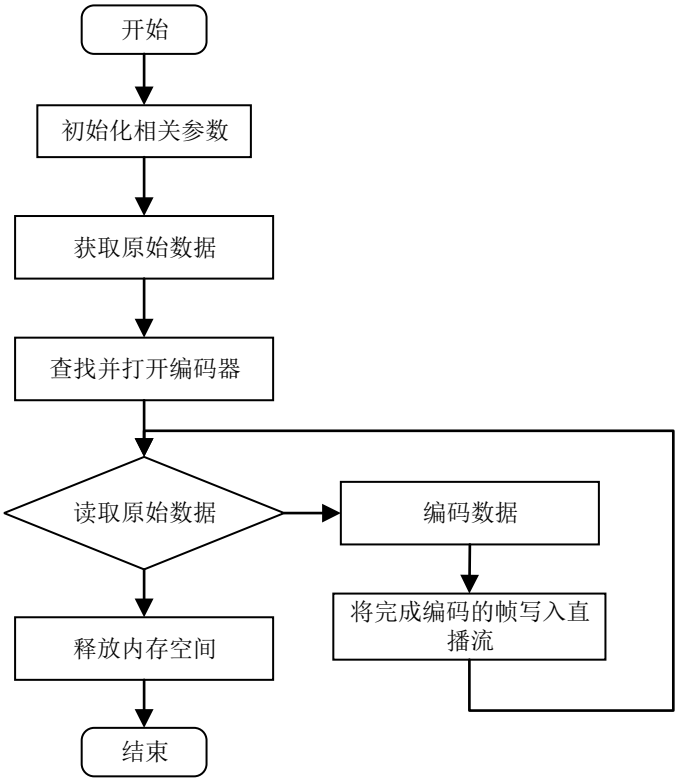

图 5.7　FFMpeg 编码流程图

Fig 5.7　Ffmpeg coding flow chart

本文使用 FFMpeg 音视频编码技术对采集到的原始 YUV 数据和 PCM 数据进行编码压缩，首先初始化相关参数，在获取到原始数据的同时配置视频编码参数，之后查找并打开解码器，最后将编码后的数据存放至 AVPacket 结构体对象中。具体 FFMpeg 编码流程图如图 5.7 所示。

5.4　视频解码

视频解码的逻辑顺序为获取到输入的视频编码数据，之后根据编码信息查找并打开解码器，最后将输入数据循环输入到解码器当中进行解码。

FFmpeg 解码模块首先调用流信息分析函数 avformat_find_stream_info()，将一部分数据流传输进该函数中，函数会通过输入的数据流得到相关的视频信息。通过得到的视

频信息，为 AVStream 结构体对象赋值，并为查找和打开解码器等操作提供所需数据信息。

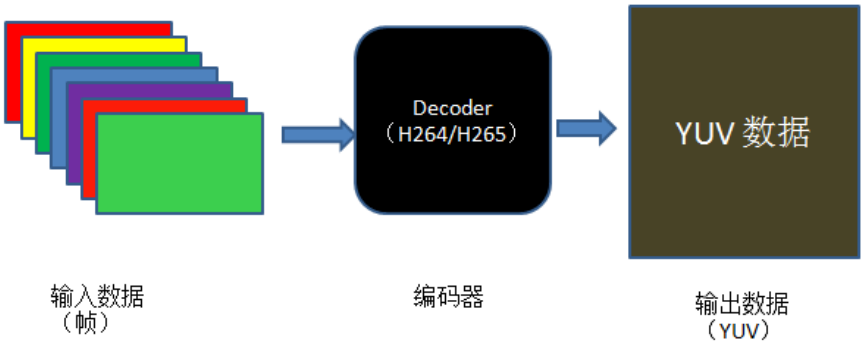

图 5.8　视频解码流程

Fig. 5.8　Video decoding process

为接收到的视频数据在系统中申请缓存空间，本论文将申请空间大小设置为 25 帧视频帧的大小。然后，通过获取到的视频流信息，为 H.264 视频流查找相应的解码器，并进行 H.264 解码操作。为解码结构体对象缓存分配内存空间、初始化操作，之后打开解码器，获取视频流数据，进入循环解码。

经过解码，获取到原始视频 YUV 数据。使用 AVFrame 结构体对象存储解码后的原始视频 YUV 数据，AVFrame 结构体对象在 FFMpeg 框架中用于存放原始数据包括视频 YUV 数据，音频 PCM 数据等，结构体中还存放一些和数据相关的信息比如在解码的时候存储了 QP 表等。

存放于 AVFrame 结构体中的 QP 表指向一块内存，里面存储的是每个宏块的 QP 值。宏块的标号是从左往右，一行一行的来的。每个宏块对应 1 个 QP。qscale_table[0]表示为 1 行 1 列的宏块 QP 值；qscale_table[1]表示为 1 行 2 列的宏块 QP 值；qscale_table[2]表示为 1 行 3 列的宏块 QP 值。宏块的大小是 16 行 16 列，宏块每行数量的计算公式为：

$$m_{stride} = width/16+1 \tag{5.1}$$

则总宏块数量为：

$$m_{sum} = ((height+15)>>4)\times width/16+1 \tag{5.2}$$

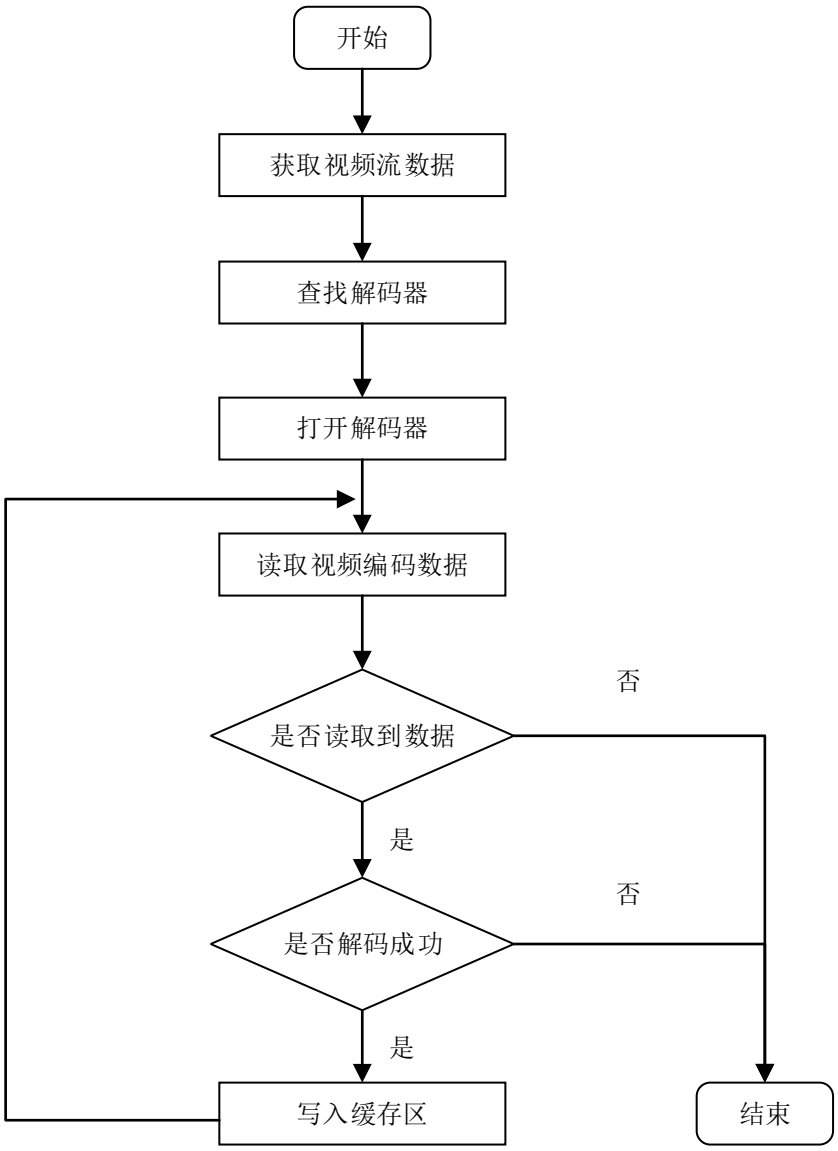

图 5.9　FFmpeg 视频解码流程图

Fig. 5.9　FFmpeg video decoding flow chart

5.5　音频解码

FFmpeg 音频解码单元，根据获取到的流信息查找并打开解码器，之后将音频 AAC 编码数据作为输入循环送入解码器当中进行解码。解码器解码后的 PCM 数据被存放在 AVFrame 结构体对象，最后将数据堆入解码缓存区，为之后进行的音视频同步、音频播放等操作提供音频数据。

本论文使用的是 4.0.1 版本的 FFmpeg。在该版本中，音频解码方面有了较大的改动，相较于之前的版本，该版本的 avcodec_decode_audio4()函数输出的解码原始音频数据的格式为 AV_SAMPLE_FMT_FLTP（float，planar），这种输出会产生杂音现象。针对这种问题，本论文使用 SwrContext 对解码出来的原始音频 PCM 数据进行重采样，再将数据存储到缓存结构体对象中。

音频重采样实现核心代码如下：

（1）SwrContext 结构体申请

```
struct SwrContext *au_convert_ctx=swr_alloc();
```

（2）配置音频重采样各项参数

```
uint64_t    out_channel_layout=AV_CH_LAYOUT_STEREO;
int             out_nb_samples=PCodeCtx_aac->farme_size;
int             out_sample_rate=44100;
int             out_channels=av_get_channel_layout_nb_channels(out_channel_layout);
int             out_buffer_size=av_samples_get_buffer_size(NULL，out_channels，
out_nb_samples，out_sample_fmt，1);
AVSampleFormat       out_sample_fmt=AV_SAMPLE_FMT_S16;
uint8_t     *out_buffer=(uint8_t*)av_malloc(MAX_AUDIO_FRAME_SIZE*2);
int64_t     in_channel_layout=
av_get_default_channel_layout(pCodeCtx_aac->channels);
```

（3）配置参数并初始化结构体

```
au_convert_ctx=swr_alloc_set_opts(au_convert_ctx，out_channel_layout，
out_sample_fmt，out_sample_rate，in_channel_layout，pCodeCtx_aac->sample_fmt，
pCodeCtx_aac->sample_rate，0，NULL);
swr_init(au_convert_ctx);
```

（4）开始音频重采样

```
swr_convert(au_convert_ctx，&out_buffer，MAX_AUDIO_FRAME_SIZE，
(const uint8_t**)pFrame->date，pFrame->nb_samples);
```

（5）释放结构体并设置为 NULL

```
swr_free(&au_convert_ctx);
```

5.6 音视频同步

在收流端软件播放视频和音频过程中，假设任由视频按照帧率播放，音频也按照自身的采用率播放，随着系统运行时间的拉长，就会出现视频和音频播放速率不一致的现象。产生这种问题的原因在于播放速度的差异，随着时间的累计会导致视频和音频播放

进度的差距越来越大。针对这种问题，本文采用了音视频同步策略，对视频和音频的播放进行矫正，保证了画面与声音的一致性[40]。

在处理音视频同步的方法中，基本的设计思想是设定播放进度的参考坐标，让视频播放进度和音频播放进度参考该坐标对自身播放进度进行调整。一般来讲共有三种不同的播放策略：视频播放进度设定为参考坐标，音频播放根据视频播放进行调控；音频播放进度作为参考坐标，视频播放进度根据音频播放进度进行调控；采用外部时钟作为参考坐标，音视频的播放进度均参看该时钟。

本论文采用以音频播放进度作为参考坐标，音频的播放时间戳 PTS 为坐标刻度，视频播放进度以音频为基准，实现音视频同步。

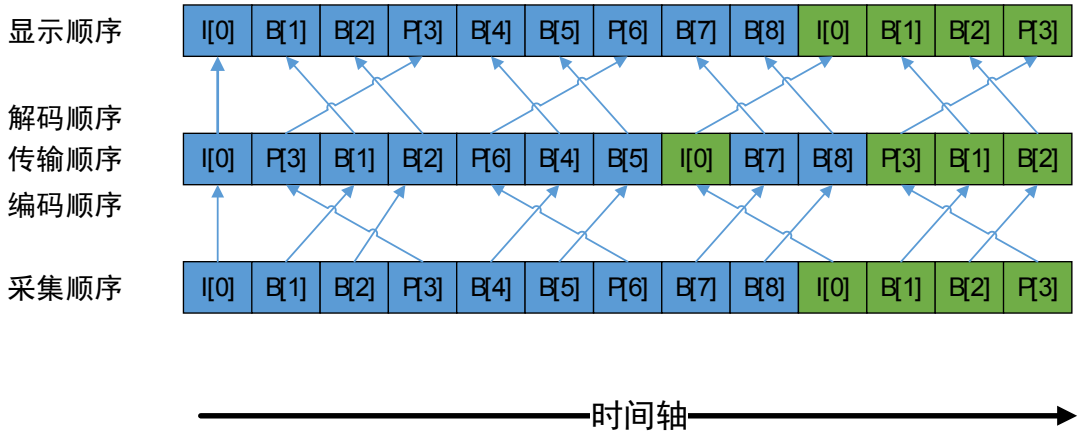

图 5.10　视频解码、显示顺序

Fig. 5.10　Video decoding and display sequence

在音视频数据中包含了两个表示时间的信息分别为解码时间戳 DTS 和显示时间戳 PTS。音频数据的 DTS 与 PTS 是相同的，但是在视频数据中，由于 B 帧的存在，导致 PTS 与 DTS 不同。视频解码、显示顺序如图 5.10 所示。

本系统视频同步到音频的基本运行逻辑是：通过计算 PTS 与时间基的乘积，进而获得当前音频的播放进度与视频的播放进度，通过播放进度我们能够获取到视频相对于音频是否播放过快或者过慢，是否需要调整当前帧的播放进度。定义 delay 为实际播放时长，duration 为理想播放时长，具体实现算法为：

（1）计算视频播放进度相较于音频播放进度的差值 diff，计算方法为利用视频的 PTS 与时间基的乘积减去音频 PTS 与时间基的乘积。

（2）计算同步阈值。同步阈值的作用是：若视频播放进度与音频播放进度的差值小于同步阈值，则认为音视频是同步的，不需要矫正 delay；若差异值大于同步阈值，则认为音视频不同步，需要矫正 delay，其中同步阈值的计算方法为：

若 duration < AV_SYNC_THRESHOLD_MIN，则同步阈值为

AV_SYNC_THRESHOLD_MIN；

若 duration > AV_SYNC_THRESHOLD_MAX，则同步阈值为：

AV_SYNC_THRESHOLD_MAX；

若 AV_SYNC_THRESHOLD_MIN < duration < AV_SYNC_THRESHOLD_MAX，则同步阈值为 duration。

（3）delay 的矫正策略如下：

视频时钟落后于音频同步时钟且落后值超过同步阈值：

若当前帧播放时刻落后于同步时钟 delay 与 diff 之和小于 0，则 delay 等于 0；否则令 delay 等于 duration 与 diff 之和。

视频时钟超前于同步时钟且超过同步域值：

一帧播放时长过长，超过最大值，仅校正为 delay=duration+diff；否则令 delay 等于 2 倍的 duration，使视频播放滞缓，等待音频播放进度。

视频时钟与音频时钟的差异在同步域值内，表明音视频处于同步状态，不校正 delay，则 delay 等于 duration。

5.7 首帧延迟优化

在无线局域网环境中，当收流端与服务器建立链接并接收流媒体数据时，收流端会出现等待播放时间过长的问题。对 FFmpeg 进行研究发现，本论文设计的系统是实时直播流数据，当收流端以随机时间点接收数据并准备播放时，需要对接收到的数据进行信息读取，信息读取时间的长短首先取决于 FFmpeg 框架中的 avformat_find_stream_info()函数。因此，研究 avformat_find_stream_info()函数，对函数代码进行优化调整，可以在较大程度上缩短等待播放时长，提升使用体验。

该函数的主要功能是读取一定大小的流媒体数据并对读取到的视频流和音频流进行解析，将解析到的数据赋值给 AVCodecContext。在 avformat_find_stream_info()函数内部，包含了查找音视频编码数据的解码器、将解码器打开、读取音视频帧和解码读取到的数据等工作。函数的运行逻辑具体如下所示：

（1）定义参数。其中包括：probesize，该参数是读入流媒体数据的大小，默认值为 500Kb；orig_nb_streams，该参数表示流媒体数据中流的数目。

（2）两次循环遍历输入流数据。执行的操作包括打开解析器，将获取到的信息赋值给 AVCodecContext，查找数据的解码器，初始化 streaminfo 的 fps_first_dts、fps_last_dts。

（3）for(;;)循环读取流媒体数据，填充 AVCodecContext。在该无限 for 循环内部，使用 read_frame_internal 读取数据，将读取到的值填充到每一个 AVStream 的 AVCodecContext。for 循环的终止条件为超出 probsize 数据大小限制。

（4）清空解码器。经过（2）、（3）步的三次循环，AVCodecCOntext 已经填充完成，此步骤将清空解码器，把读取到的数据存放至队列中。

（5）清理内存空间。将申请到的内存空间进行清理，防止内存泄漏。

通过对 avformat_find_stream_info()函数运行逻辑的分析，发现该函数在解析流媒体数据时，运行时间与读取到的数据量大小成正相关。函数默认使用 500K 数据量读入大小的目的在于保证可以完整的分析出流媒体数据中包含的视频和音频信息，为之后的 avcodec_find_decoder()等函数提供可靠地保障。本论文使用的数据封装格式为 flv，该封装格式具有体积小、加载速度快等优势，使用 avformat_find_stream_info()函数默认提供的读取数据量大小，会对整体运行造成时间上的冗余。因此，本论文以减少 FFmpeg 首帧打开时间，去除 avformat_find_stream_info()函数内部循环冗余时间为目的，针对 probesize 参数进行了优化，设计了优化实验，优化后将首帧打开时间从 4~5s 缩减至 1~2s。本次实验的设备硬件条件如表所示：

表 5.1　设备配置

Tab. 5.1　Equipment configuration

类型	系统配置	参数
Windows 客户端	操作系统	Windows 7
	CPU	i5-6700
	内存	4G
	硬盘	1T
服务器	操作系统	Linux

实验的具体流程为：

（1）收流端设置 probesize 参数。

（2）推流端点击开始推送。

（3）收流端点击开始播放按钮，同时开始计时，单位为毫秒。

（4）收流端开始播放流媒体数据时，停止计时。

（5）重复执行步骤（2）~（4）30 次，并将计时器时间记录保存。

按照上述实验条件和步骤进行多次实验，实验结果如图 5.11 所示。将 probesize 数值调整为 140Kb 时，出现黑屏现象，表明 avformat_find_stream_info()在读取数据大小为 140Kb 的条件下，无法成功解析本系统的流媒体数据。

通过对 avformat_find_stream_info()函数的分析，并以改变 probesize 参数为方法，设计多组实验，最终将本系统收流端首帧打开延迟减低至 1~2s，优化了用户体验。

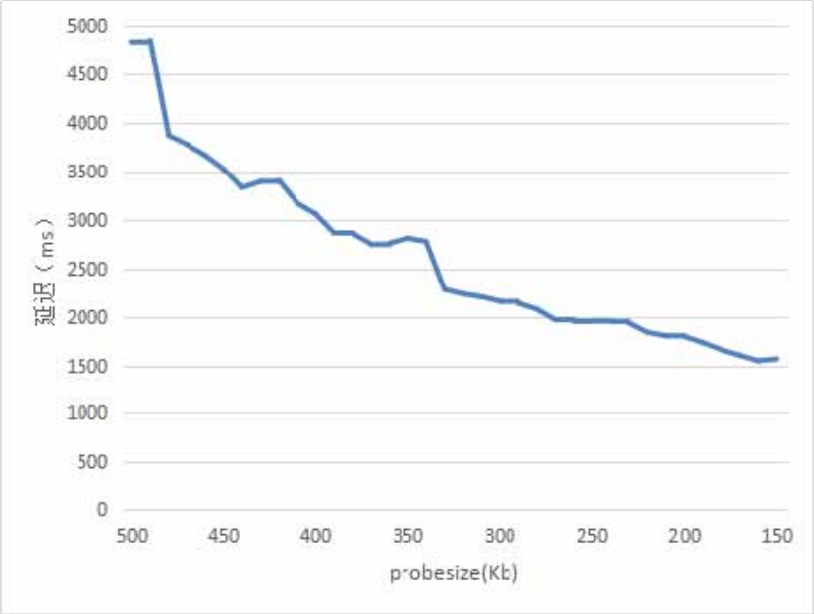

图 5.11　首帧优化实验结果

Fig. 5.11　Experimental results of first frame optimization

5.8　本章小结

本章完成了客户端软件的实现。本章首先对 FFmpeg 在 Windows 端和 Android 端的编译移植与客户端软件开发环境的配置进行了详细介绍，之后对教育直播软件中音视频采集、音视频编码、音视频解码等技术进行了介绍与实现，并且对系统实现过程中出现的问题提出了解决方案，包括音视频同步、首帧打开延迟等。

6 系统测试与性能分析

完成基于 FFmpeg 的教室直播系统之后，针对教育直播系统的各项功能设计了相关实验并进行评估。在不断的进行实验的过程中，发现并解决系统在设计与实现上的不完善的地方和缺点，整体系统内部一些参数和变量在进行多次实验的基础上，不断进行修改，对教育直播系统的系统功能、网络延迟等方面进行测试，以达到最佳的效果。

6.1 系统基本功能测试

本节主要是对教育直播系统的基本功能进行了测试，检验系统是否能够满足教学过程中的基本需求。测试过程使用的系统硬件设备如下表所示：

表 6.1 设备配置

Tab. 6.1 Equipment configuration

类型	系统配置	参数
Windows 客户端	操作系统	Windows 7
	CPU	i7-6700
	内存	8G
	硬盘	1T
服务器	操作系统	Linux
Android 客户端	操作系统	Android 8.1.0
	处理器	2.0GHz 八核
	运行内存	4G

6.1.1 软件安装测试流程

基于 FFmpeg 的教育直播系统通过 InstallShield Wizard 对 Windows 端软件进行发布以及使用 Android studio 直接生成 Android 端 apk，其中包含了软件需要用到的所有文件。

在安装 Windows 软件时，用户可自行选择安装文件夹，安装结束后会在用户桌面创建快捷方式。

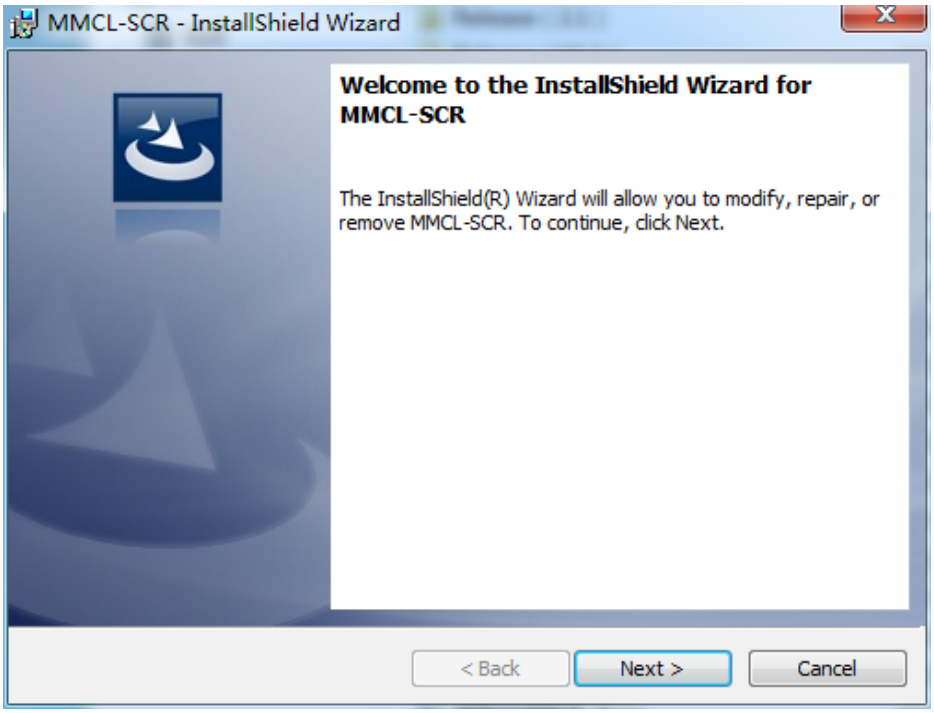

图 6.1　安装界面

Fig. 6.1　The installation interface

在安装 Android 端 app 的过程中，系统会询问用户是否对 app 进行授权，用户均点击同意即可。

图 6.2　安装界面

Fig. 6.2　The installation interface

6.1.2 系统基本功能测试

点击运行 Windows 端推流软件后输入推流地址，点击推流软件上的开始按钮，之后软件开始推送屏幕和麦克风数据。客户端链接到服务器后，客户端就可以收到由服务器转发的 RTMP 流媒体数据。

系统的推流和收流基本功能测试的结果如图 6.3 所示。在图中可以看到，教育直播系统满足了设计之初的基本需求，可以完成流媒体数据的传输。市场上的多媒体传输方式，大多是以牺牲图像、音频质量为代价换取较低延迟，或者以传输时延过大为前提，提供较好的图像、音频质量。本论文所研究的教育直播系统，在保证了低延迟的前提下，提供了较好的图像、音频质量。在教学方面，低延迟是授课时首要考虑的先决条件。另外，本系统服务器的搭建在树莓派 3B 上，该设备体积较小，具有便捷性并且对于时间、地点等没有约束。

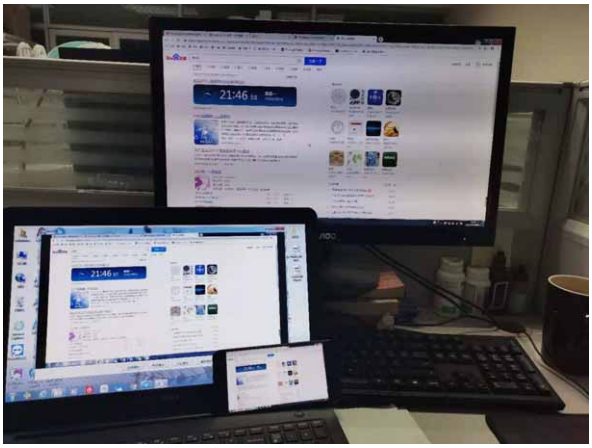

图 6.3 基本功能测试

Fig. 6.3 Basic function test

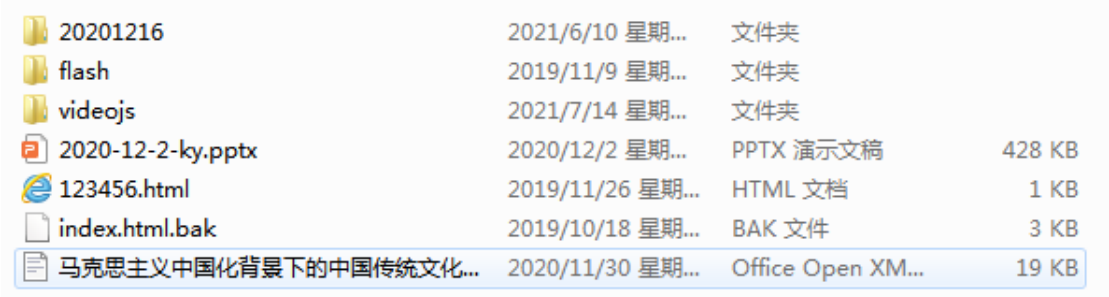

图 6.4 共享服务器文件夹

Fig. 6.4 Shared server folder

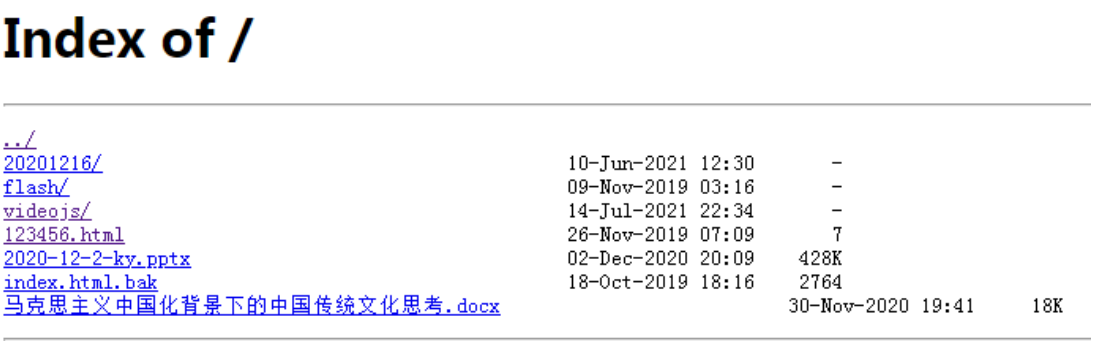

图 6.5　文件下载页面

Fig. 6.5　File download page

系统的文件上传和下载基本功能如图 6.4 和图 6.5。教师可通过电脑共享操作服务器文件夹，将文档、ppt 等资料上传至服务器。学生可通过浏览器浏览服务器上可下载的文件，点击进行下载。

6.2　系统稳定性测试

为了确保教育直播系统可以在承担一定业务压力的条件下仍正常工作，本论文对系统进行了稳定性测试，使用一定数量的设备模拟业务压力，对系统功能进行测试。

为了完成系统的稳定性测试，本论文提出并实现了两种不同的测试方案，分别为：

（1）Windows 单机推送，Windows、Android。

（2）Windows 多机推送，Windows、Android。

具体的测试方法为，在实验室环境下，搭建整体教育直播系统，使系统在不同数量不同的测试方案下连续工作 12 小时。

6.2.1　单机推流和单机收流

表 6.2　Windows 测试结果

Tab. 6.2　Windows test result

序号	播放时长	延迟时间	稳定性
1	1h	<1000ms	稳定
2	2h	<1000ms	稳定
3	3h	<1000ms	稳定
4	6h	<1000ms	稳定
5	9h	<1000ms	稳定
6	12h	<1000ms	稳定

表 6.3　Android 测试结果

Tab. 6.3　Android test result

序号	播放时长	延迟时间	稳定性
1	1h	<1000ms	稳定
2	2h	<1000ms	稳定
3	3h	<1000ms	稳定
4	6h	<1000ms	稳定
5	9h	<1000ms	稳定
6	12h	<1000ms	稳定

本论文的实验环境为 xx 学校实验室，面积为 7m×9m。一台 Windows 系统设备作为推流设备。一台 Windows 系统设备，为 Windows 软件收流。一台 Android 设备，作为 Android app 收流。实验结果如表 6.2 和 6.3 所示。

从表格中可以看出，系统整体时延均在 1s 左右，可以满足日常业务需求。

6.2.2　多机测试

表 6.4　Windows 测试结果

Tab. 6.4　Windows test result

序号	播放时长	延迟时间	稳定性
1	1h	<1000ms	稳定
2	2h	<1000ms	稳定
3	3h	<1000ms	稳定
4	6h	<1000ms	稳定
5	9h	<1000ms	稳定
6	12h	<1000ms	稳定

表 6.5　Android 测试结果

Tab. 6.5　Android test result

序号	播放时长	延迟时间	稳定性
1	1h	<1000ms	稳定
2	2h	<1000ms	稳定
3	3h	<1000ms	稳定
4	6h	<1000ms	稳定
5	9h	<1000ms	稳定
6	12h	<1000ms	稳定

本论文的实验环境为 xx 学校实验室，面积为 7m×9m。在实验过程中，一台 Windows 设备作为推流设备，一台 Windows 设备为预备轮换推流设备，两台 Windows 为收流设备，两台 Android 设备为收流设备。在实验过程中，每隔一段时间会切换推流设备。

从表格中可以看出，教育直播系统经历 12 小时连续工作后，仍旧保持稳定的工作状态。在测试期间，系统整体时延均在 1s 左右，可以满足日常业务需求。

6.3 推流用户切换测试

教育直播系统在使用过程中，会遇到主讲人频繁更换的情况。在这种情况下，推流设备也会随着主讲人的更换而改变。用户切换测试，是在系统工作的情况下，通过更换推流设备来测试系统的流畅程度。测试的结果切换时间越短，说明系统的性能越好。

用户切换测试的实验场地是 xx 学校实验室，面积为 7m×9m。选取两台 Windows 设备为轮流切换设备，一台 Windows 设备为收流设备，一台 Android 设备为手机端收流设备，实验结果如图 6.6。

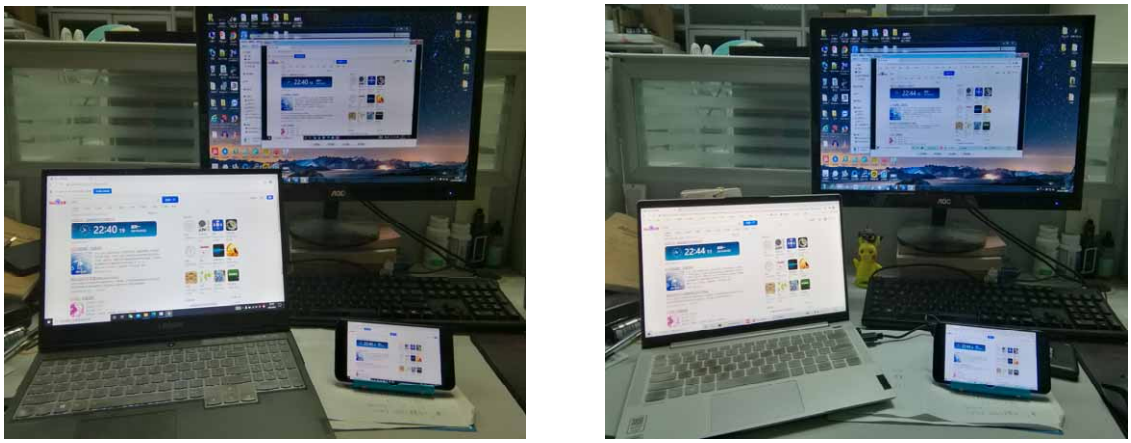

图 6.6 用户切换测试

Fig. 6.6 User switching test

通过用户切换测试实验，可以得到以下结论：在多主讲人情况下，收流端软件能够根据系统推流端软件频繁更换做出流畅的切换，可以满足日常授课的使用需求。

6.4 系统延迟测试

系统延迟的定义是在流媒体数据传输过程中，推流端发送数据到收流端播放数据所需要的时间，这个时间的长短与系统性能相关，通常越小的系统延迟，系统性能也就越好，用户的体验程度也越好。为了检测本系统的延迟情况，本论文设计了两组不同用户量的延迟测试实验。

（1）单一用户收流实验

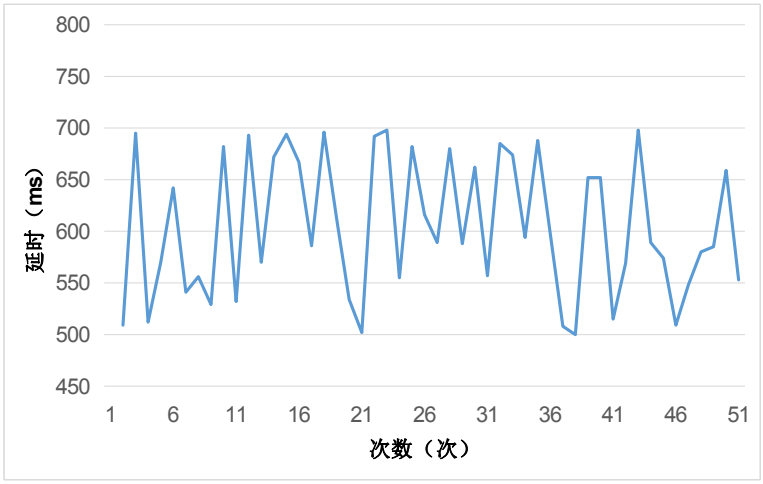

图 6.7　Windows 软件延迟测试结果

Fig. 6.7　Windows software delay test results

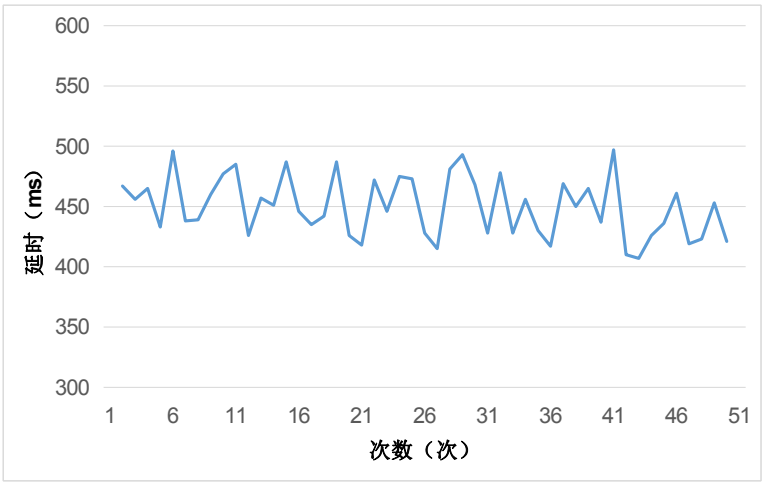

图 6.8　Android 软件延迟测试结果

Fig. 6.8　Android software delay test results

本次实验在 xx 学校实验室，面积为 7m×9m。实验过程中使用的设备为一台 Windows 设备作为推流设备，一台 Windows 设备为收流设备，一台 Android 手机为收流设备。具体的设备信息如下表所示。在实验过程中，连续播放 PPT 页面并记录切换延迟。从图 6.7 和 6.8 中可以看出，教育直播系统的延迟能够稳定在 1s 以内，可以满足正常使用。

（2）多用户收流实验

标准小型教室的实验场地为某校教室，面积为 6m×8m。设置本实验的目的在于测量在真实的教学环境下教育直播系统的稳定程度与延时情况。根据实验结果，在多人授课情景下，本论文研究的教育直播系统能够满足正常的使用需求。

表 6.6　测试结果

Tab. 6.6　Test Result

编号	系统	处理器	内存	平均延时
1	Windows 7	i3-3120M	4G	600ms
2	Windows 7	i5-3230M	4G	570ms
3	Windows 10	i7-3632QM	4G	580ms
4	Windows 10	i7-5500U	4G	610ms
5	Windows 7	i5-4200U	4G	490ms
6	Android 8.1	armeabi-v7a	4G	500ms
7	Android 9.0	armeabi-v7a	4G	550ms
8	Android 10.0	armeabi-v7a	4G	530ms
9	Android 9.0	armeabi-v7a	4G	470ms
10	Android 8.0	armeabi-v7a	4G	550ms

6.5　本章小结

本章完成了对系统基本功能的测试，以及针对系统重要性能指标设计了多组实验包括：稳定性测试、推流用户切换测试和延迟测试。实验结果表明本文设计的教育直播系统可以满足日常教学需求。

结　论

使用投影仪进行授课的传统教学方式存在着有效投影范围小、受物理空间局限、互动性较差、信息传递不便等问题。为了克服这些问题，更好地将互联网应用于大教室、多教室的授课过程中，本文深入分析教师与学生的需求，设计并实现了基于 FFmpeg 的教育直播系统，并对系统使用过程中的首帧打开延迟、流畅性能等问题进行了优化，最终提高了系统的使用体验。本文主要完成了以下工作：

（1）针对本系统面向的校园应用场景，设计了系统的整体架构，并详细介绍了系统在实现过程中使用到的技术包括：FFMpeg、Nginx 服务器、RTMP 传输协议、H.264 编码标准以及 AAC 编码标准。

（2）完成了流媒体服务器的搭建。以 Raspberry Pi 3B 为硬件设备，成功搭建 Nginx 服务器，并在 Nginx 服务器上添加 RTMP 模块来实现音视频数据的传输，极大程度降低了数据传输过程中的延迟。同时，实现了 Nginx 下载功能和 Samba 服务的组合，大大简化了教师和学生之间传递文件的流程。

（3）成功在 VS2010 和 Android studio 开发软件中配置了 FFMpeg 开发环境。深入研究 FFMpeg 在处理音视频过程中使用到的函数以及结构体，针对系统需求进行了优化。采用了 H.264 视频编码标准和 AAC 音频编码标准对获取到的屏幕、麦克风数据进行进行压缩编码，大大减少了系统在传输过程中对网络环境的依赖。采取了视频同步到音频的音视频同步策略，解决了音视频播放进度参差不齐的问题。

设计了多组实验，对本系统的各项指标进行评估。结果表明，本文设计并实现的教育直播系统能够满足在校园环境中的授课需求。

本文在流媒体数据处理效率、教师学生间文件传递方式、网络延迟等方面取得了一些成功，但考虑到教育直播系统的复杂程度，本系统还能够在以下几个方面进行更进一步的研究。

（1）在服务器方面，增添服务器对流媒体数据保存的功能，方便学生回放教师授课信息。

（2）本系统完成了防止恶意信息流入的推流鉴权功能，但是在数据传输过程中，数据容易被截获。应对传输的流媒体数据加密，防止教学信息被不法分子盗用。

（3）本系统完成了 Windows 平台和 Android 平台的软件实现，但是在 4G 以及新兴的 5G 网络环境下，应增加网络环境检测功能，软件根据网络环境自适应调整传输视频的像素格式。

参 考 文 献

[1] 黄诗文.基于 FFmpeg 的高性能高清流媒体播放器软件设计[D].杭州:浙江大学,2012.

[2] 李安民.流媒体发展背景下的网络教学资源建设研究[J].计算机产品与流通,2020(9):79+87.

[3] 杨明. 在线教育直播平台设计与实现[D].北京:北京交通大学,2019.

[4] 任衡.基于互动课堂的屏幕直播系统延迟优化与设计实现[D].北京:北京工业大学.2019.

[5] 申宇欢.视频编码技术的研究与优化[D].成都:电子科技大学,2019.

[6] 席文强.基于 FFmpeg 的高清实时直播系统设计与实现[D].西安:长安大学.2017.

[7] 李闻斌,庞璐宁,黄晟.实时流媒体传输系统中关键技术的研究与实现[J].信息通信,2020(12):159-161.

[8] 戴伟.基于 Nginx 高性能 Web 服务器的理论研究与性能改进[D].南京邮电大学,2019.

[9] 李翊翔.在线音乐直播的商业模式探究[D].厦门:厦门大学,2017.

[10] VASEK M, WADLEIGH J, MOORE T. Hacking Is Not Random: A Case-Control Study of Webserver-Compromise Risk[J]. IEEE Transactions on Dependable & Secure Computing, 2016, 13(2):206-219.

[11] 阿不都克里木·玉素甫,王亮亮.基于自主可控平台的 FFmpeg 在线视频转换系统[J].计算机与现代化,2020(01):81-84+116.

[12] 辛长春,娄小平,吕乃光.基于 FFmpeg 的远程视频监控系统编解码[J].电子技术, 2013, 40(1):3-5.

[13] 黄松涛,夏挺.基于 FFmpeg 的远程监控调试系统[J].电子世界,2019(1):177-178.

[14] 雷霄骅,姜秀华,王彩虹.基于 RTMP 协议的流媒体技术的原理与应用[J].中国传媒大学学报(自然科学版),2013(6):59-64.

[15] 邓正良.基于 FFmpeg 和 SDL 的视频流播放存储研究综述[J].现代计算机,2019(22):47-50.

[16] 郝朝,刘升护.基于 FFmpeg 和 SDL 的遥测视频解析技术[J].计算机技术与发展,2019,29(4):191-194.

[17] BYEON JOOHYUNG, et al. Fast Thumbnail Extraction for H.264/AVC, HEVC and VP9[J]. Applied Sciences,2021,11(4):1844-1844.

[18] BOONTHEP N, CHAMNONGTHAI K. Correction to: A Method of MotionEstimationBased H.264 Video Coding Using Optimal SearchRange[J]. Wireless Personal Communications, 2021, 115(4):2833-2850.

[19] 杨景炜.网络视频监控图像接入软件系统的设计与实现[D].成都:电子科技大学,2012.

[20] 王雯,马宁.基于 H.264 和 DM6437 的煤矿井下视频监控系统设计[J].机械工程与自动化,2017(5):167-168.

[21] CHEN, YI, WANG, et al. An adaptive data hiding algorithm with low bitrate growth for H.264/AVC video stream[J]. Multimedia tools and applications, 2018, 77(15): 20157- 20175.

[22] 严羽,王永众,杨来邦.基于无人机的实时图传系统[J].智能计算机与应用,2019,9(4):65-70.

[23] 熊翘楚,蓝海,陈晶.一种面向 AAC 音频 MDCT 系数域的隐写分析方法[J].广州大学学报(自然科学版),2020,19(2):69-77.

[24] 吴杨.基于 RTMP 协议的实时视频监控系统的设计与实现[D].杭州:浙江工业大学,2019.
[25] 夏俊锋.HTML5 下的视频会议系统中基于 RTMP 的直播解决方案的设计与实现[D].广州:华南理工大学.2015.
[26] 王丽静.巴二小英语教学互动管理系统设计与实现[D].大连:大连理工大学,2018.
[27] FRENCH H, JIE L, PHAN T, et al. Real Time Video Qoe Analysis of RTMP Streams[C]// Performance Computing & Communications Conference. Orlando: 2011.
[28] HUANG J, WU D M, LIU X P. Implementation of the RTMP Server based on Embedded System[C]// International Conference on Computer Science & Information Processing. Xi'An: 2012.
[29] NURROHMAN A, ABDUROHMAN M. High Performance Streaming Based on H264 and Real Time Messaging Protocol (RTMP)[C]// 2018 6th International Conference on Information and Communication Technology (ICoICT). Bandung: 2018.
[30] LEI X, JIANG X, WANG C. Design and Implementation of Streaming Media Processing Software based on RTMP[C]//International Congress on Image & Signal Processing. Hang Zhou: 2013.
[31] 崔营营,刘洋,刘志强.基于 RTMP 协议的桥梁视频监控关键技术研究[J].物联网技术, 2020, 10(11):12-14+17.
[32] 张美平,吴德平,王灿杰等.基于树莓派的智能家居设计与实现[J].计算机系统应用,2019, 28(8):109-114.
[33] 葛钰,李洪赭,李赛飞.一种 Web 服务器集群自适应动态负载均衡设计与实现[J].计算机与数字工程,2020,48(12):3002-3007.
[34] 刘金秀,陈怡华,谷长乐.基于 Nginx 的高可用 Web 系统的架构研究与设计[J].现代信息科技,2019,03(11):94-97.
[35] 王利萍.基于 Nginx 服务器集群负载均衡技术的研究与改进[D].济南:山东大学,2015.
[36] 余亮,王红,王元航.基于 Android Studio 的智慧校园多媒体管理 App 设计[J].电子世界,2020(12):114-115.
[37] 张欢庆,高丽,宋承祥.基于 ARM 的嵌入式 Linux 交叉编译环境的研究与实现[J].计算机与数字工程,2012(2):157-159.
[38] 王新蕾,刘乃丰,夏济海.基于 DirectShow 的视频处理 Filter 组件设计与实现[J].现代电子技术,2016,39(13):46-50.
[39] 王宗志.基于 H.264 编码的常态化录播教学系统的设计[J].电脑知识与技术,2020,16(11): 65-66+97.
[40] 成敏,谢杉杉,徐强,宋占伟.基于 Android 的音视频同步交互系统[J].吉林大学学报(信息科学版),2016,34(4):507-514.

致　　谢

三年紧张而又充实的研究生生活马上就要结束了，回首望去，我收获的不仅仅是知识，还有从身边师长、同门、朋友身上学习到了对待事情的态度，让我在面对困难时多了一分从容和淡定。

首先感谢我的研究生导师王洪玉教授。王老师不仅是一位兢兢业业、专心于学术的老师，更教导了我学习和生活中的道理。王老师对待学术一丝不苟，当我们遇到科研困难时，他总是会耐心地指导我们，与我们讨论，为我们提供思路，教授我们方法。在这三年里，承蒙老师教导，我才能在学术科研中有一丝长进。

其次，我想感谢我的同门苗鑫师兄、席铮师兄和蒯越师妹，在教研室的日子是充实且快乐的，我们共同努力，相互帮助，在学习和生活中我们都是彼此可靠的伙伴。愿诸位在接下来的日子里，百尺高楼更进一步，能够实现自己的梦想。

最后，我想感谢大连理工大学，这一切都离不开学校对我的栽培，是学校为我提供了丰富的教学资源，成长的平台，祝学校越来越好。

时光飞逝，日月如梭，三年的时间很快，但是读研期间收获到的成长是我一生的财富，感恩！

大连理工大学学位论文独创性声明

作者郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用内容和致谢的地方外，本论文不包含其他个人或集体已经发表的研究成果，也不包含其他已申请学位或其他用途使用过的成果。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

若有不实之处，本人愿意承担相关法律责任。

学位论文题目：基于FFMpeg的教育直播系统设计与实现

作 者 签 名：陶奎印　　日期：2021 年 6 月 10 日

大连理工大学学位论文版权使用授权书

本人完全了解学校有关学位论文知识产权的规定，在校攻读学位期间论文工作的知识产权属于大连理工大学，允许论文被查阅和借阅。学校有权保留论文并向国家有关部门或机构送交论文的复印件和电子版，可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印、或扫描等复制手段保存和汇编本学位论文。

学位论文题目：基于FFMpeg的教育直播系统设计与实现

作者签名：陶建印　　日期：2021 年 6 月 10 日

导师签名：[signature]　　日期：2021 年 6 月 10 日