广东工业大学硕士学位论文

（工程硕士专业学位）

基于 H.265 的视频教育系统的设计与实现

袁勋

二〇一七年 五月

分类号：　　学校代号：11845

UDC：　　密级：　　学　　号：2111403111

广东工业大学硕士学位论文

（工程硕士专业学位）

基于 H.265 的视频教育系统的设计与实现

袁勋

指导教师姓名、职称：原玲　副教授

企业导师姓名、职称：刘祖华

专业或领域名称：电子与通信工程

学生所属学院：信息工程学院

论文答辩日期：2017 年 5 月

A Dissertation Submitted to Guangdong University of Technology

for the Degree of Master

（Master of Engineering）

Design and Implementation of Video Education System Based on H.265

Candidate: Yuan Xun

Supervisor: Associate Prof. Yuan Ling

May 2017

School of Information Engineering

Guangdong University of Technology

Guangzhou, Guangdong, P. R. China, 510006

摘要

随着网络的普及与流媒体发展，网络教育以一种不同于传统教育的形态出现在日常生活中。其优点也不断地显现出来，大大增加了学生的主动性，且打破传统教育在很多方面的局限性。以校园网为基础的视频教育平台不仅能满足广大师生对名师讲堂、校园活动的需求，还可以进一步拓展为视频会议等其他应用平台。本文设计以校园网为载体的视频教育系统，采用视频直播的形式来实现网络远程教学，系统在对 Live555 开源框架进行二次开发的基础上搭建流媒体服务器，并采用 H.265 视频编码标准以及 RTSP、RTP/RTCP 网络传输策略来实现基于 H.265 的视频教育系统。

本文首先进行系统方案选择并对所用到的相关技术进行简单介绍，其中包括视频编码标准、流媒体协议以及流媒体开源框架。通过对当前主流的视频编码标准编码后的视频进行质量评估，选择 H.265 视频编码标准作为系统的视频编码标准；通过分析常见流媒体服务器开源框架的优缺点，并根据系统所采用的流媒体协议，选择 Live555 作为系统的流媒体服务器。其次，简单地介绍视频教育系统的总体设计，系统主要分为视频采集上传客户端和流媒体服务器以及播放器客户端组成，视频的采集上传客户端采用即插即用的 USB 摄像头作为硬件设备，调用 FFmpeg 库对采集的原始视频流编码成 H.265 码流，为了能适应复杂的、容易丢包的无线网络环境和对后续音频或者字幕的扩展，将编码后的 H.265 码流进行 TS 封装。流媒体服务器是在 Live555 的基础上进行二次开发，采取模块化设计，主要包括流媒体接收模块和转发模块，并采用 RTSP 协议作为交互协议，实现服务器与各个播放器客户端之间的通信。按照 RTSP 会话请求，流媒体服务器从网络接收数据，并对接收的数据进行 RTP 打包与发送，实现流媒体转发功能。播放器客户端选择支持 RTSP 协议的开源播放器，如 VLC 播放器，降低了整个系统的开发成本、节约开发时间。最后，采用 VLC 播放器当作播放器客户端对系统进行了全面的功能测试，并对流媒体服务器转发时延以及系统的时延做出了估计。测试结果表明，系统能够满足实时性要求，视频画面清晰，具有良好的实用价值。

关键字：网络教育；视频编码；Live555；FFmpeg；流媒体协议

ABSTRACT

With the popularity of the network and the development of streaming media, online education in a different from the traditional form of education in daily life. Its advantages are constantly emerging, greatly increasing the student's initiative, and break the traditional education in many aspects of the limitations. The campus network-based video education platform can not only meet the teachers and students on the teacher lectures, campus activities, but also can further expand the video conferencing and other application platforms. This paper designs the video education system with campus network as the carrier, adopts the form of video live to realize the network distance teaching, builds the streaming media server on the basis of the secondary development of Live555, adopts H.265 video coding standard and RTSP, RTP/RTCP network transmission strategy to achieve based on H.265 video education system.

This paper first introduces the system scheme and introduces the related technology, including video coding standard, streaming media protocol and streaming media open source framework. Through the current mainstream video coding standard encoded video quality assessment, select the H.265 video coding standard as the system video coding standard; By analyzing the advantages and disadvantages of common streaming media server open source framework, and according to the system used by the streaming media protocol, select Live555 as a system of streaming media server. Secondly, a brief introduction to the overall design of the video education system, the system is divided into video capture upload client and streaming media server and player client components, Video capture upload client using plug-and-play USB camera as a hardware device, call the FFmpeg library to capture the original video stream encoded into H.265 stream, in order to adapt to complex, easy to packet loss of the wireless network environment and Follow-up audio or subtitles expansion, the encoded H.264 stream will be TS package. Streaming media server is based on the secondary development of Live555, modular design, including streaming media receiving module and forwarding module, use of RTSP protocol as an

interactive protocol to achieve the communication betweent server and various players client. According to the RTSP session request, the streaming media server receives data from the network, and carries out RTP encapsulation and sending of the received data to realize the streaming media forwarding function. The player client chooses an open source player that supports the RTSP protocol, such as a VLC player, which reduces the development cost of the entire system and saves development time. Finally, the VLC player is used as a player client to perform a comprehensive functional test of the system, the delay of the streaming media server and system are estimated. Test results show that the system can meet the real-time requirements, the video screen is clear, with good practical value.

Key words: Online education;Video coding;Live555;FFmpeg;Streaming media protocol

目录

CONTENTS

第一章 绪论

1.1 课题研究背景及意义

随着互联网的发展，作为一种与传统教育不同模式的网络远程教育形式出现在网络上，在我国掀起一股新的浪潮。而远程教育中最基本、最核心的视频教学形式，能够完全体现教师授课现场的图象、语音等信息，这大大增进了学生在学习中的主动性，学生可以选择自己感兴趣的视频内容进行观看学习。在一定程度上，实现了按需教育，打破传统教学模式在教学时间和空间上的局限性，教师和学生可以在不同的地方，在相同或不同的时间进行教学和学习。

视频教育一般所涉及的多媒体文件包括视频、音频以及字幕等文件，尤其是视频文件数据量巨大，因此十分不利于多媒体文件的存储和传输，通过扩大存储空间、加宽线路带宽以及加大传输速度都无法从根本上解决。因此，需要对原始视频文件进行必要的压缩编码处理。另外，现在网络不断发展，计算机也不断更新，视频服务的多样化使得数字视频产业朝着高分辨率、高稳定性以及优越的网络适应能力的方向发展，因此，更高压缩率的视频编码标准成为迫切的需求。这便加速了视频编码标准的发展与更迭，一些旧的、不能满足现状需求的编码标准被压缩率更高、更高效的视频编码标准所取代。目前应用最广范的 H.264 视频编码标准日益显现出不足性，已经无法满足发展迅速的数字视频应用的需求。

基于上述对现有的问题的分析，本文提出了“基于 H.265 的视频教育系统”的解决方案。视频教育系统是通过采用 H.265 视频编码标准的网络直播技术来实现网上教学，与 H.264 相比，H.265 视频编码标准节省带宽，在视频质量相同的情况下，只要原来一半的带宽，并且分辨率越高优势越显著。从而实现高清教学，真正再现课堂，有效提高教学质量。

在学校校园中，特别是在有多个校区的学校中，搭建视频直播平台将会满足教师和学生的各种需求，包括讲课、校园活动等。视频教育系统除了能提供以上服务外，还能扩展为视频监控或者视频会议，这样便大大提高了校园网的利用率。

1.2 国内外研究现状

在我国，各高校网络视频课程建设工作开始于 2011 年，并在同年的 4 月份复旦大学推出国内首场内容为讲座的网络视频课程，紧接着西安交通大学在建校 115 年庆典上，正式对外发布其网络视频公开课，备受外界关注。随着信息化的快速发展，MOOC 作为远程教育的最新发展，在 2012 年起在全球掀起数字教育革命，让任何渴望知识的人们通过网络学习世界上名校的课程。在美国，高校先后推出 Coursera、edX 和 Udicity 三大 MOOC 平台，吸引世界众多知名大学纷纷加盟，向全球开放在线教育资源服务。紧接着 MOOC 在我国也获得快速发展，我国多所“985”知名高校加盟以上 MOOC 平台，与哈佛、斯坦福、耶鲁、麻省理工等世界一流大学共建全球在线课程。2013 年 5 月北大宣布加入 edX，9 月北大与 Coursera 签署合约；2013 年 11 月网易公开课正式上线，与北大合作为北大的 MOOC 项目提供技术支持；2014 年 1 月，由上海交通大学为运营方，另外 C9 高校以及部分其他 985 高校合作建立“好大学在线”MOOC 平台[1]，于此同时，慕课网、中国大学 MOOC 等 MOOC 平台席卷中国。MOOC 的教学和学习是参与与互动，以学习者为中心从学习者兴趣出发，在文献[2]中 Coursera、edX 和 Udicity 以及 Udemy 等平台将视频作为核心的课程信息传输形式，将视频与练习无缝的结合起来。文献[3]中实施翻转课堂主要的环节之一是学生通过教师事先提供的视频学习新知识。因此，结合浏览国内各大 MOOC 平台网站发现，MOOC 平台大多数以 B/S 架构，并且以视频录播的形式实现视频教学，教师事先录制视频通过后期的处理，并配套特有的练习题，然后上传到 MOOC 平台。

本教育系统以视频直播的形式实现实时教学，在文献[4]中提出了一种基于校园网便捷式的视频直播系统是，能够采集老师现场教学的视频和音频信号、老师所用课室电脑屏幕的视频和音频信息以及鼠标的运动等多路信息，并对所采集的信号进行编码，最后通过校园网，利用流媒体服务器转发给学生的客户端播放学习。文献[5][6]提出了基于校园网的视频直播系统并重点分析了视频直播技术的特点以及组成。其中文献[5]将其运用到学术讲座、远程会议、毕业典礼等，在 2014 年暑期，配合学校研究所直播了为期 5 天的精品暑期班，取得了良好的效果。文献[7]以增城学院为例，介绍基于校园网的视频直播系统的实现过程，并用在多媒体课程以及图书馆学术报告厅进行学术报告。文献[8][9][10]设计了一种流媒体直播系统，基于对等网络，在网络中是每个节

点既是消费者又是提供者，能够很好的解决当大规模用户访问服务器时，对网络带宽和服务器并发能力要求高的问题，而且容易部署，扩展性高。

在文献[11]中讲述了利用 3G/4G 移动网络，采用手机摄像头视频拍摄并传输的直播系统，用户可以很方便地利用手机进行直播。文献[12]中分析了现在大多数视频直播的方式，提出采用开源视频库 FFmpeg 对视频压缩编码并使用流媒体服务器转发视频的手机视频直播系统的解决方案。在文献[13]中提出一个移动学习系统，使视频直播应用移动电话。包括实时虚拟教室架构，通过采用流媒体技术，学习者可以访问现场的视频。文献[14]中介绍了一项移动技术提供实时访问视频教育讲座，并在上海交通大学实现两次超过 1000 人的学术讲。然而以上文献只是介绍了将视频直播系统运用到学术讲座以及其他方面，甚至在文献[13][14]中将直播系统应用到移动端，大大提高了使用者、学习者的便捷性。

在文献[15][16]中介绍 H.264 更直接、简洁的视频编码器，增强压缩性能和压缩质量，更重要的一点是能够防止数据的丢失和数据被非法使用。文献[17]介绍利用 DirectShow 技术实现基于 SIP、RTP/RTCP 协议的视频监控系统，使用 SIP 协议来连接客户端和服务器的会话请求，视频图像的处理采用 H.264 视频编码标准，提供了良好的视频质量和适应性的网络条件。以上文献中系统均采用 H.264 标准对视频进行编码，随着视频向高清方向发展，H.264 逐渐显现出不足性，由于 H.264 编码宏块的大小固定，在高清视频是高分辨率下编码宏块个数增长很快，出现大量冗余信息，导致编码效率下降。

为解决以上提出的问题，一种能够满足实时性、高效性的视频编解码技术已经是迫在眉睫。ITU-T 的 VCEG 工作组和 ISO/IEC 的 MPEG 工作组再次合作，在 2010 年成立了联合视频工作组 JCT-VC，展开对新一代编码标准的开发。因此，高效的视频编码标准 HEVC 应运而生[18][19]。

在现实应用场景中，越来越多的行业领导者和企业都接受并研究 HEVC 视频编码标准。各大主流的视频门户和网站都已经能够提供 HEVC 的服务，我国通信行业的领导者华为、中兴以及国外的高通等，它们每年也投入大量经费开始了对 HEVC 的研究，并开发出相关的产品。

2012 年第一个 H.265 编码器是由爱立信公司研发并发布，在这之后又有很多的企业推出自己的 HEVC 编码芯片和相关的解码器。2013 年 5 月，日本广播公司和三菱电

机公司合作开发了首款超高清的 HEVC 实时编码器，在同年的 12 月份，我国也研发出国内首台 HEVC 实时视频编码器。2014 年 4 月，由乐视研发并发布了一款基于 HEVC 的智能电视，融创天下也发布了基于 HEVC 的端到端流媒体系统，推动 HEVC 解决方案在我国的普及。2014 年 11 月，微软官方确认最新发布的 win10 操作系统支持 HEVC 视频编码标准； 2014 年 12 月，由华为研发并发布的荣耀盒子，能够支持 4K 视频的 H.265 编解码，由此可见，新一代视频编码标准已经广泛应用于高清视频领域。

1.3 本文主要研究内容

本文设计是视频教育系统是采用基于 H.265 的视频直播系统实现，在视频直播系统中包含视频采集、视频压缩编码以及流媒体服务器等模块的设计。视频采集设备是采用即插即用的 USB 摄像头来实现对视频数据的捕捉，然后对捕捉的视频进行编码并进行 TS 封装，然后发送至流媒体服务器，响应远端的视频播放器请求并实时播放，以达到视频教育的目的。实现本教育系统主要涉及的关键技术主要包括视频编码技术以及流媒体服务器信令的交互、流媒体转发、流媒体传输技术，主要工作如下：

1. 为了提高视频数据传输的效率和节省有限带宽，系统采用 FFmpeg 库对采集的视频进行压缩编码处理。

2. 为了更好的兼容网络的复杂多变，同时为了未来系统对语音或者字幕等业务的扩展，系统对编码后的 H.265 码流做了进一步的 TS 封装。

3. 基于 Live555 实现流媒体服务器的二次开发，采用 RTSP 协议作为服务器与各个客户端的交互协议，实现流媒体 RTP 数据包的接收与转发。

4. 最后，系统采用具有代表性的 VLC 播放器对系统功能进行测试。

1.4 本文结构安排

本文在研究了网络教育系统的背景下，通过熟悉视频编码原理、各种流媒体协议以及开源服务器框架，提出基于 H.265 的视频教育系统，本文主要分为六个章节来介绍。

第一章：绪论。本章主要介绍了所选课题的研究背景及意义，分析了课题的国内外研究现状，并给出了本文的结构安排。

第二章：系统相关技术介绍。本章主要对系统所采用的视频编码技术、流媒体协议以及流媒体服务器开源框架进行选择，最后，介绍了视频质量评价标准并对相关技术进行简单介绍。

第三章：系统的总体设计。首先介绍了视频教育系统的整体框图，然后分别介绍了视频上传客户端以及流媒体服务框架设计，及其开发平台的选择。

第四章：视频上传客户端的实现。介绍了 FFmpeg 开源库的配置、H.265 编码的特点以及 TS 包结构，并采用 FFmpeg 库对采集的视频编码、TS 封装和发送。

第五章：流媒体服务器的实现。本章简单介绍了 Live555 开源协议，重点介绍了流媒体服务器的搭建，其中包括客户端与服务器的 RTSP 会话请求过程、流媒体服务器数据的接收以及数据的 RTP 打包与发送。

第六章：系统测试。本章采用支持 RTSP 协议的 VLC 播放器对整个系统的功能进行测试，并对系统的延时进行估计。

第二章 系统相关技术介绍

2.1 视频编码技术

2.1.1 视频编码标准

数字视频在我们现实生活中扮演着重要角色，然而，实际生活和实际应用中所接触到的视频，都是经过压缩编码处理过的视频。这是因为未经过压缩编码处理的原始视频数据量是非常大，无法在现实实际中应用与传输。因此，视频的压缩编码处理是视频应用一项重要技术，经过压缩编码后的视频消除了视频数据中的冗余信息，大大减少了视频的数据量。

视频信号的冗余信息分为两种：数据冗余和视觉冗余。

1. **数据冗余**。数据冗余主要是视频数据内部出现的冗余信息，包括空间上的、时间上的以及结构上的冗余。即视频图像中相邻两帧之间以及一帧图像像素之间都存在这很强的重复性，消除这些重复性也不影响视频质量。因此，这种压缩输入无损压缩。

2. **视觉冗余**。我们人类的眼睛存在某些生理特性，使得人眼对亮度和色度的敏感度不一样，因此，在视频编码的时候引入适量地误差，也不容易被看出来。利用眼睛的视觉这些视觉特征，可以牺牲一定的视频信息换取对视频数据的压缩，这种压缩属于有损压缩。

基于上述两种视频中的冗余信息对数字视频进行压缩编码，视频的数据量得到大幅的减少，对视频保存与传输都更加有利。包括 MPEG1[20]，MPEG2[21]，H.264[22]等视频编码标准以及以后视频编码标准都采用了混合编码的方式，即将变换编码、运动估计与运动补偿以及熵编码三种编码方式相互结合起来使用对视频进行压缩编码。通过变换编码来消除视频图像的帧内冗余，使用运动估计与运动补偿的作用是消除帧间冗余，熵编码来提高压缩效率。下图 2-1 给出了混合编码模型：

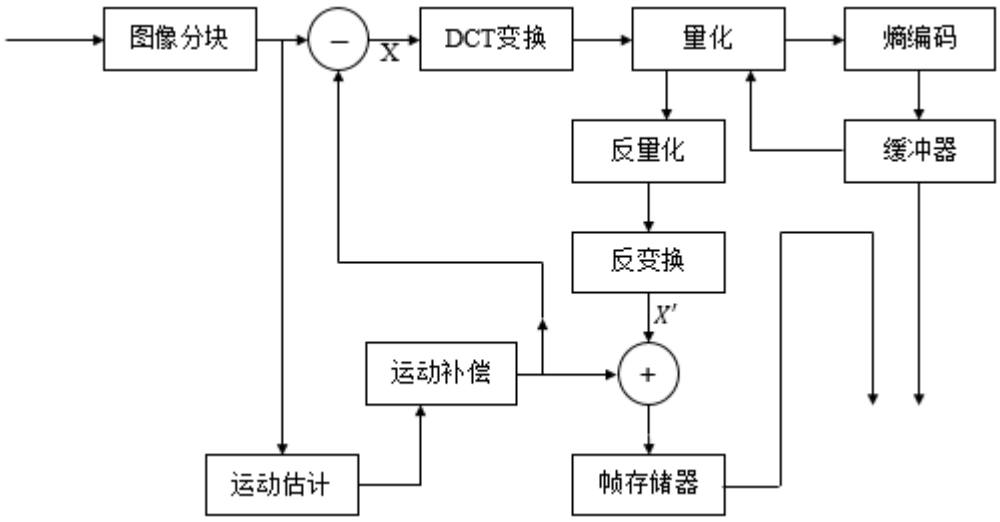

图 2-1 混合编码模型

Figure 2-1 Hybrid coding model

从框图中可以看出，将三种编码方式结合起来起到更好的编码效果。首先，在输入端输入一幅图像经过分块处理，之后再将图像块分为两路，第一路经过运动估计和运动补偿得到预测图像，而另一路与第一路图像做相减运算得到残差图像 X，该残差图像要经过 DTC 变换[23]和量化，输出的数据又分为两路：一路经过熵编码进行编码后送到缓存器，等待传送；另一路进行反量化以及反变换得到的图像与之前的预测图像相加得到新的图像，并送入存储器。

H.265 视频编码标准[24]是 ITU-T VCEG 制定的新一代视频编码标准，H.265 相对于 H.264 以及以往的国际视频编码标准，从理论上说，H.265 视频编码标准的编码框架并没有太大的改变，继续采用混合编码框架。如下图 2-2 所示，包括变换、量化、熵编码、帧内编码、帧间编码以及环路滤波[25]等模块。

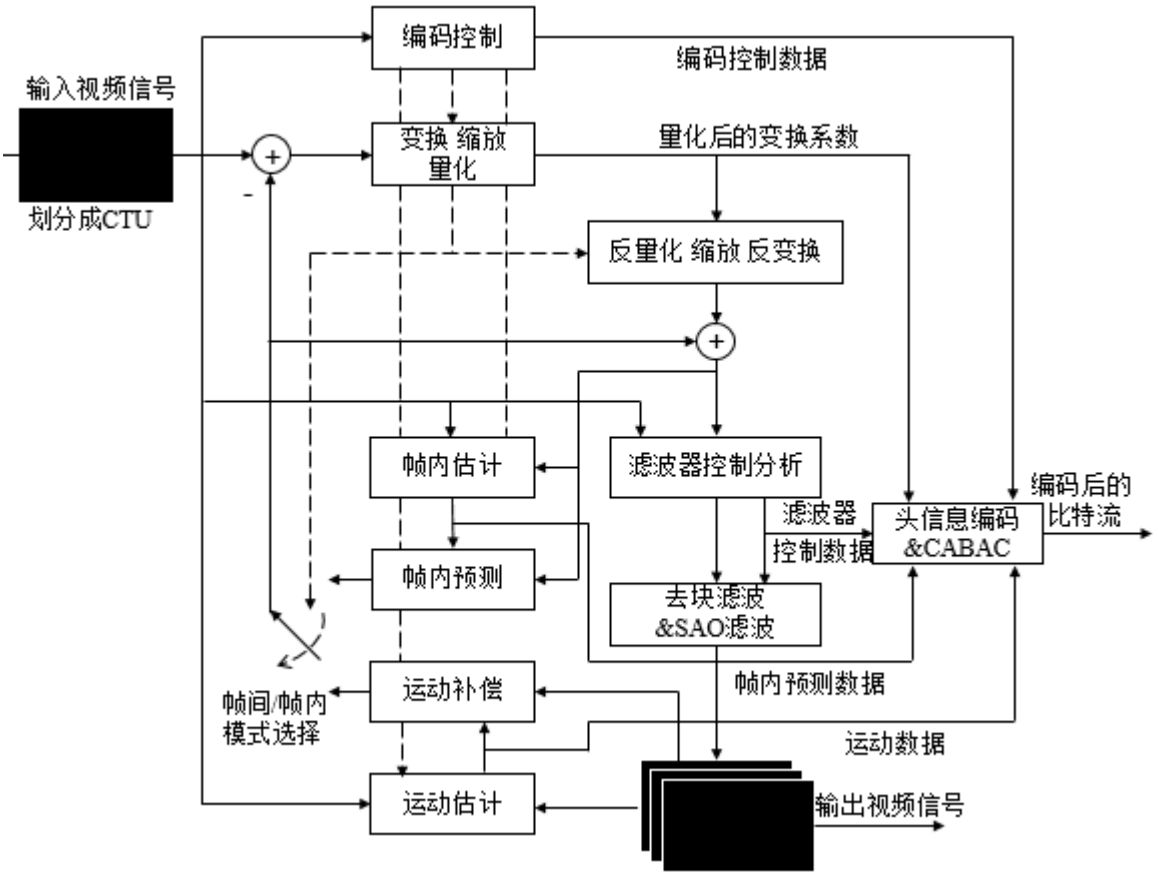

图 2-2 H.265 编码框架

Figure 2-2 H.265 coding framework

H.265/HEVC 的性能有了很大的提升，这源于新编码工具的使用以及具有特色的核心技术。H.265/HEVC 中新的编码技术有很多，例如基于四叉树的宏块分割结构[26]、性能更好的 CABAC[27] 、改进的帧内预测[28]技术、残差四叉树变换(RQT)[29]技术、自适应运动矢量预测 AMVP、合并技术 Merge、可变尺寸的离散余弦变换、模式依赖的离散正弦变换、采样自适应偏移(SAO)[30]技术、以及新的样点自适应补偿滤波器等。

2.1.2 FFmpeg 介绍

FFmpeg 是一个集录制、转换、音/视频编解码功能为一体的完整解决方案，现如今应用十分广泛。FFmpeg 是一个在 Linux 平台上开发的开源项目，但也可以在大多数操作系统中编译和使用，比如支持 Windows、Mac OS X、Linux 等[31]。并且它目前已经支持 AC3、DV、MPEG、FLV 等 40 多种编码，MPEG、ASF、AVI 等 90 多种解码[32]。如今，已经被许多开源项目所使用，比如 Google Chrome、MPlayer、VLC、Quick

Time 的 Perian、Directshow 的 ffdshow 等。FFmpeg 是在 LGPL/GPL 协议下开源发布的，任何人都可以自由使用。

FFmpeg 包含了很多库文件，最常用的包括以下四个库文件：libavformat、libavutil、libswscale 和 libavcodec，其中，libavcodec 是目前领先的音视频编解码库。

下面对 FFmpeg 中的一些主要结构体进行分析：

AVFormatContext：属于整个项目中的根结构体，最基本的结构体之一。其中包含的码流参数比较多的，描述了媒体文件或媒体流的构成和基本信息。

AVCodecContext：这个数据结构对编码器的上下文信息进行了描述，包含了各种编码器需要的信息，包括视频的宽度和高度、音频的采集率和通道数、音频的原始采样格式、时间基数单位、编码器的信息等。

AVIOContext：是 FFmpeg 管理输入输出数据的结构体。

AVStream：在 FFmpeg 中是一个频繁使用的类，它包含了流的一些数据信息，包括帧率、帧数、持续时间、时间基数单位等。

AVFrame：此结构体用于存储原始数据即没有压缩之前的数据，例如对视频来说是 YUV 或者 RGB 格式的视频，对音频来说是 PCM 格式。

AVPacket：FFMPEG 作用这个数据结构来暂时存储解复用后，解码之前的多媒体数据，比如音视频或者字幕数据。

2.1.3 MPEG-2 TS 简介

在本系统只包含视频而未对音频业务做介绍，结合系统的设计思想以及系统的网络环境情况，决定采用 TS 流来传输 H.265 视频帧数据。不仅能够适应复杂的网络环境，而且能够更好的适应未来业务的发展，具有更好的扩展性，例如添加音频业务时就可以很方便的加入系统中。

从 1991 年开始，ITU-T 和 ISO 开始联合制定新一代活动图像编码标准，终于在 1993 年通过名称为“通用活动图像编码”的标准草案，即 MPEG-2。MPEG-2 标准分为三个重要的组成部分，分别是系统层、视频层以及音频层[33]。

MPEG-2 系统层语法扩展较大，主要目的是将一个或者多个音视频以及其他的基本数据流整合成单个数据流，以便适应系统数据的存储和传输。系统层共定义了两类数据传输的码流，即节目流 PS 和传送流 TS。两者都是由基本的码流 PES 组成，而

PES 是将压缩编码后的音频或者视频数据以及一些额外的辅助数据打包而成。两者的适应环境不一样，本文只介绍 TS 流。MPEG-2 系统层原理如图 2-3 所示：

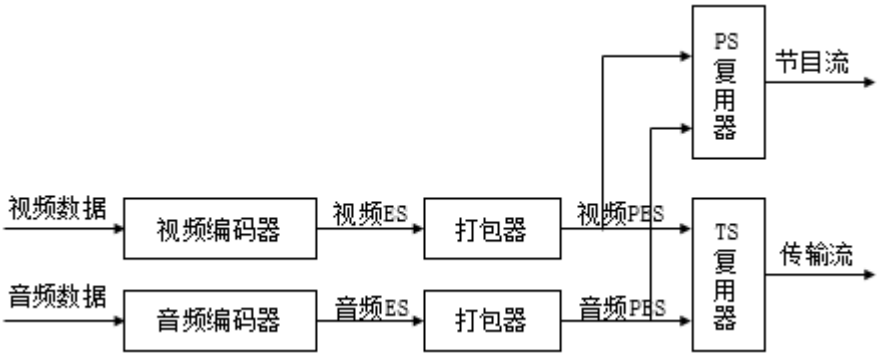

图 2-3 MPEG-2 系统框图

Figure 2-3 MPEG-2 system block diagram

TS 流将有多个时间基点的数据流合成一个数据流，其中属于同一个节目的各个数据流，比如同一段视频中的音频和视频数据流，与打包之后的 PES 数据包具有相同的时间基点。

由于 TS 码流采用了固定长度包结构的缘故，TS 码流常用于信道情况不好、有误码的环境，例如在无线情况下流媒体的传输，当传输有误码、破坏或者丢失了某一处 TS 包的同步信息时，接收端便可以在固定位置上检测包中的同步信息，从而恢复数据流，这样能够做到避免丢失。在恶劣的信道环境下，由于 TS 码流具有较强的抵抗传输误码的能力，因此在传输流媒体中基本上都采用了 TS 码流的格式。

2.2 流媒体协议

本课题选择 RTSP 协议作为流媒体服务器与流媒体客户端之间的信息交互协议，实现资源描述信息的获取、建立连接、播放实时流等功能，而 RTSP 控制的实时流则是通过 RTP/UDP 协议进行传输的。

2.2.1 RTSP 协议

实时流协议[34] 属于应用层协议，在体系结构上位于实时传输协议和实时传输控制协议之上，协议本身并不传输数据，通过 RTP/RTCP 协议来完成流媒体数据的传输。RTSP 协议只是控制基于 RTP 上的实时数据的传送，提供了对流媒体播放、暂停等操

作，它选择并控制多个数据源的传送通道。该协议充当流媒体服务器的控制协议，用于建立和处理播放器客户端和流媒体服务器端之间的实时流会话。开发者可以使用此协议来控制客户端与服务器间的流媒体串流，实现对视频和音频数据传输功能的控制。

RTSP 请求报文的方法包括：OPTIONS、ANNOUNCE、DESCRIBE、SETUP、TEARDOWN、PLAY、RECORD、PAUSE、GET_PARAMETER 和 SET_PARAMETER 等[35]。下表 2-1 是 RTSP 的方法的总览，其中方向代表是由哪一方主动请求哪一方，对象指当前 RTSP 方法用来操作的是表示流，而要求是指为了实现 RTSP 会话，当前方法是否必要。

表 2-1 RTSP 方法介绍

Table 2-1The Introduction of RTSP Methods

方法	方向	对象	要求
OPTIONS	C->,S->C	P,S	必要
DESCRIBE	C->S	P,S	推荐
ANNOUNCE	C->S,S->C	P,S	可选
SETUP	C->S	S	必要
PLAY	C->S	P,S	必要
PAUSE	C->S	P,S	推荐
RECORD	C->S	P,S	可选
REDIRECT	C->S	P,S	可选
SET_PARAMETER	C->S,S->C	P,S	可选
GET_PARAMETER	C->S,S->C	P,S	可选
TEARDOWN	C->S	P,S	必要

2.2.2 RTP 协议

实时传输协议属于传输协议，它主要负责流媒体数据包的实时发送与接收。因为 RTP 传输的是音视频数据，数据量很大，允许少量的丢包，所以传输层一般使用 UDP 协议，以降低资源的消耗。由于 RTP 协议只负责数据的传输而不保证传输的质量，因此需要采用实时控制协议进行流量控制和拥塞控制来保证传输的质量。在流媒体数据的传输过程中的一个重要问题就是无法预料数据包的到达时间，所以在 RTP 的协议中包含了时间标签，发送端在 RTP 打包时可以设置时间标签，接收端就可以按照时间标

签来以正确的速率来还原原始的数据。

RTP 数据包包括协议头和负载两部分。RFC3550 文件中给出了 RTP 协议头字段的详细信息，其中，前 12 个字节是固定的 CRSCidentifiers 字段是可选的，负载部分为要传输的媒体数据。图 2-4 给出了 RTP 报文格式：

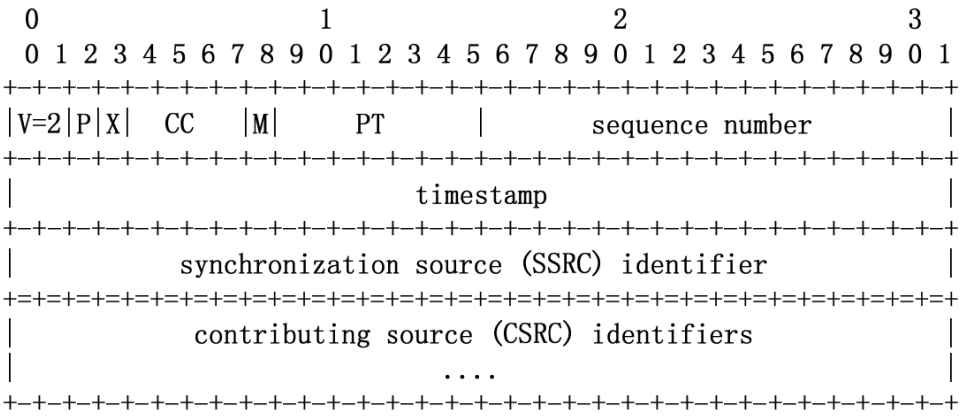

图 2-4 RTP 报文

Figure 2-4 RTP packets

2.2.3 RTCP 协议

实时控制协议[36]与 RTP 协议一起共同定义在 1996 年提出的 RFC1889 中，它是和 RTP 协议一起使用完成网络流量控制和拥塞控制，RTCP 的主要工作是对网络传输质量进行反馈，即统计网络状况参数，把网络状况反馈给发送端，发送端凭借反馈回来的网络情况作出调整，这样能够更好的适应网络状况。RTCP 分组和 RTP 分组一样都是使用 UDP 传输协议进行信息的传输，RTP 分组传输的是数据，而 RTCP 分组传输的是反馈信息，包括发包率、丢包率以及时延等信息。RTCP 通过不同功能的数据报来实现 RTCP 协议的功能，RTCP 数据包[37]有以下 5 种：

(1) **发送端报告包** SR：发送端发出的统计信息。

(2) **接收端报告包** RR：接收端接收的统计信息。

(3) **源描述包** SDES：主要功能是描述与会话源相关的信息。

(4) **离开包** BYE：离开时报告包。

(5) **应用包** APP：解决了 RTCP 的扩展性问题，开发新的应用。

2.3 流媒体服务器开源框架

常见的流媒体服务器包括微软的 Windows Media Service、Real Networks 公司的 Helix Server 以及 Adobe 公司的 Flash Media Server 等，但这些服务器都属于闭源的软件系统，不能按照自己的要求来进行二次的开发，所以并不适合本系统的开发与使用。

在对服务器进行二次开发的基础上，满足要求的主流开源流媒体服务器框架主要有 live555 和 Darwin。Darwin 是由苹果开发的一套流媒体解决方案，视频格式支持较多，但项目复杂、不活跃、信息少。与 Darwin 相比，Live555 代码很容易二次开发，项目更加活跃。Live555 流媒体服务器是基于事件驱动的流媒体解决方案，使用 C++ 语言开发，支持 RTSP、RTP/RTCP 以及 SIP 等高效流媒体协议，具有良好的可移植性。同时它要求的前端播放器软件也更加广泛，只需要支持 RTP/RTSP 协议的播放器就可以无缝地访问这个平台，比如流行的 VLC、MPlayer。

同时，系统采用 Live555 作为流媒体服务器开发平台，主要三个原因：一是拥有强大的开源团队并且经过多年的改进和完善，使得 Live555 已经发展成为一个性能稳定、易于扩展的开发平台；第二、允许无线客户端访问，可以方便地使用手机访问实时；第三、Live555 Media Server 对视频数据流的转发延迟很短，有利于实时视频连接和观看。

2.4 视频质量评价

由于数字视频的数据量非常大，不利于存储和传输。为了降低视频在传输过程中所占的带宽，因此要对视频进行压缩编码处理，本系统选择 H.265 视频编码标准对采集的视频进行编码处理。由于视频的压缩处理是系统的关键，因此，本节介绍主流的视频质量评价方法来对编码后的视频进行质量评估，以此来作为视频好坏的判断标准。

2.4.1 视频质量评价方法

通常，视频质量评价的方法大致分为两种：主观质量评估(SQA)和客观质量评估(OQA) [38]。前者是凭借观看者的主观感觉来评估视频的质量，后者是基于数学模型给出的量化参数来衡量视频的质量。如下图 2-5 所示，主观评价方法又包含 DSIS、DSCQS、

SSM，而客观评价方法也包含全参考模型、部分参考模型以及无参考模型。

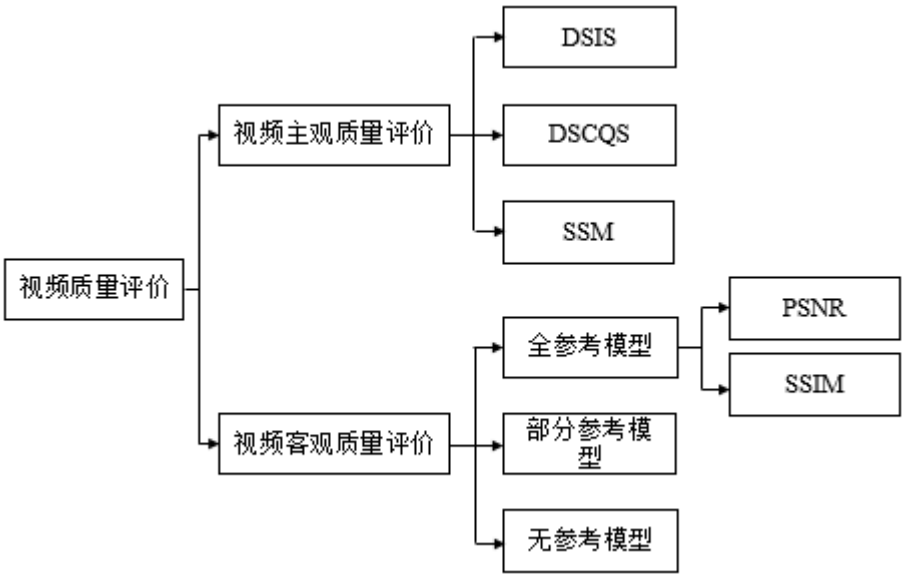

图 2-5 视频质量评价方法

Figure 2-5 Video quality evaluation method

1. 视频质量主观评价

视频质量的主观评价是通过测试者观看视频，凭借测试者的主观判断来评价视频的质量。主观质量评价通常使用连续双刺激质量法[39](double stimulus continuous quality scale，DSCQS)，对于任何观看者提供原始视频图像和压缩编码的失真图像，由观看者根据主观感觉连续地给出分数。

视频主观质量评估是表达人们视觉体验最准确的方式，可以直观准确反映视频的质量好坏，缺点是主观评价方法依赖于人的主观感觉，稳定性非常差，而且实现受制于环境步骤复杂、代价高昂，不利于实时视频通信中的视频质量评估[40]，因此它常被用作客观评价方法的参考和有效性标准。

2. 视频质量客观评价

目前，视频质量专家组提出了三种测试方法，分别是完全参考、部分参考以及无参考[39]视频质量测量方法。如下图 2-6 所示：

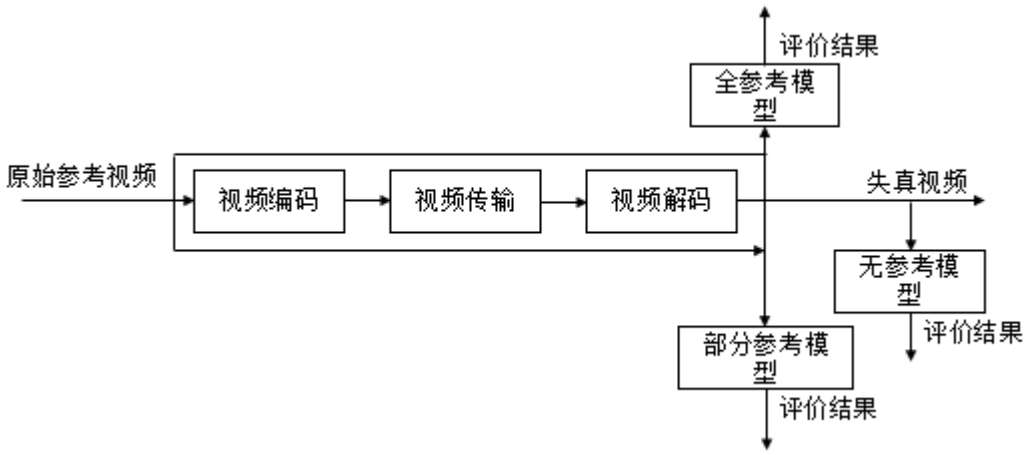

图 2-6 视频客观评价方法

Figure 2-6 objective evaluation of video

在测量视频质量的时候，全参考方法[41]是在假定能够获得原始视频的前提下，将编码后失真的视频与原始视频进行质量比较，其结果是编码后视频相对于原始视频的相似性或者保真度，而不是真是的获得图像的质量。部分参考模型表示在原始视频中提取部分特征信息来衡量视频的失真度，从而能够得到失真视频质量。最后，无参考模型是不需要任何原始视频的任何信息，只是对编码后的失真视频进行时域和频域分析来提取失真视频的特征，以对视频的质量进行评估。

本文主要依赖全参考模型来完成对编码后视频的质量评估工作，目前三类全参考方法分别是基于像素差异(Pixel Difference)的评价方法、基于结构相似度(Structural Similarity)的评价方法和基于视觉感知(Visual Perception)的评价方法[42]。本文主要对前面两种方法进行重点介绍：

(1) 基于像素差异评价方法

基于像素差异的视频质量评价方法是指通过计算比较失真视频与原始视频之间每个像素点的差异，然后联合各个像素点的差异，获得视频质量的评价方法。其中，均方差(mean square error，MSE)和峰值信噪比(peak signal noise ratio，PSNR)[38]是应用最广泛的两种方法，其数学定义分别如式(2.1)、（2.2）所示：

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - r_i)^2 \tag{2.1}$$

$$PSNR = 10log_{10}(\frac{(2^{n-1})^2}{MSE}) \tag{2.2}$$

其中，x_i和r_i分别为失真视频和原始参考视频中的第 i 个像素点的值，N是整个视

频中像素点的个数，n 为像素采样值的比特数。

(2) 基于结构相似度的评价方法

基于结构相似度的评价方法重点是在比较原始参考视频与失真视频的结构信息，通过比较两者的结构相似度，而不是像素点的差异度来进行视频质量评估。由此，学者们提出了基于结构相似度的评价方法。其数学定义式如式(2.3)所示：

$$SSIM(x,y) = l(x,y)C(x,y)S(x,y) \tag{2.3}$$

$$l(x,y) = \frac{2\mu_x\mu_y+C_1}{\mu_X^2+\mu_Y^2+C_1} \tag{2.4}$$

$$C(x,y) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_X^2+\sigma_Y^2+C_2} \tag{2.5}$$

$$S(x,y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3} \tag{2.6}$$

其中，$l(x,y)$代表均值，$C(x,y)$代表，$S(x,y)$代表协方差。

结构相似框图如下图 2-7 所示：

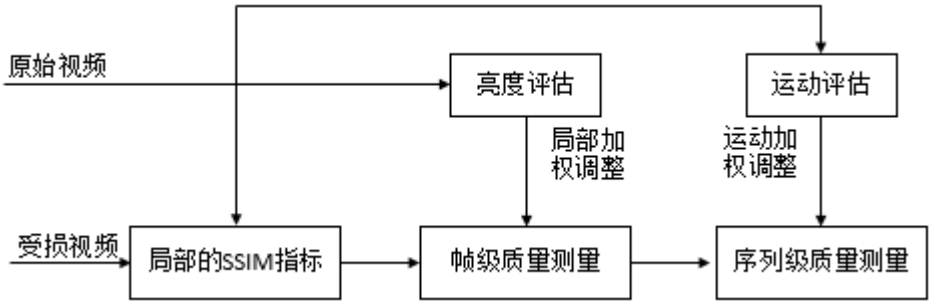

图 2-7 结构相似方法

Figure 2-7 Structure similarity method

基于结构相似评价方法与基于像素差异的评价方法不同是通过感知失真视频与原始视频的结构相似度来衡量视频质量，如果前后两幅图像结构相似，则说明两幅图像质量变化不大，质量损失不严重。

2.4.2 视频质量比较

根据视频质量客观评测方法，通过下面列出的几幅图片，分别为原始视频中的一帧和编码后视频的一帧，来表明视频客观评估所得出的客观值与主观感受之间的关系。测试所用的视频序列均采用相同编码器进行编码，在参数相同的前提下，分别按不同的码率编码并计算出此帧视频的 PSNR 和 SSIM 值。下面的图像均为视频序列的第 25 帧图像。

如下图 2-8 所示，码率为 200kps，PSNR=25.899，SSIM=0.667。下图 2-9 所示，码率为 600kbps，PSNR=31.202，SSIM=0.888。

图 2-8 编码后的一帧图像

Figure2-8 A coded image

2-9 编码后的一帧图像

Figure2-9 A coded image

下图 2-10 所示，码率为 1000kbps，PSNR=33.673，SSIM=0.932。下图 2-11 所示，为视频编码之前的原始视频图像

图 2-10 编码后的一帧图像

Figure2-10 A coded image

图 2-11 原始视频图像

2-11 Original video image

如下图图 2-12、2-13 为整个序列包含 50 帧，下图为整个序列 PSNR 以及 SSIM 比较：

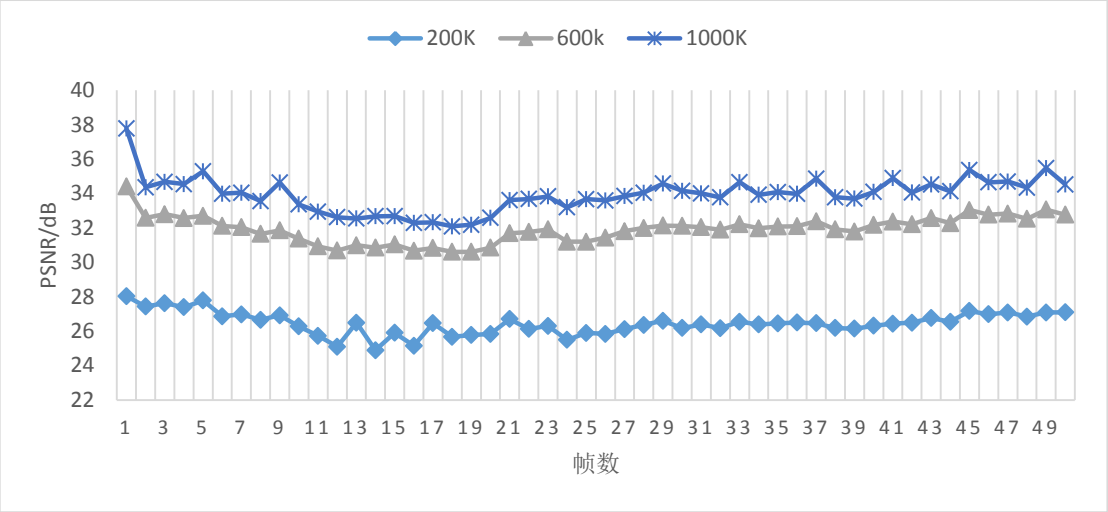

图 2-12 50 帧图像 PSNR 值

Figure2-12 50 frame images PSNR value

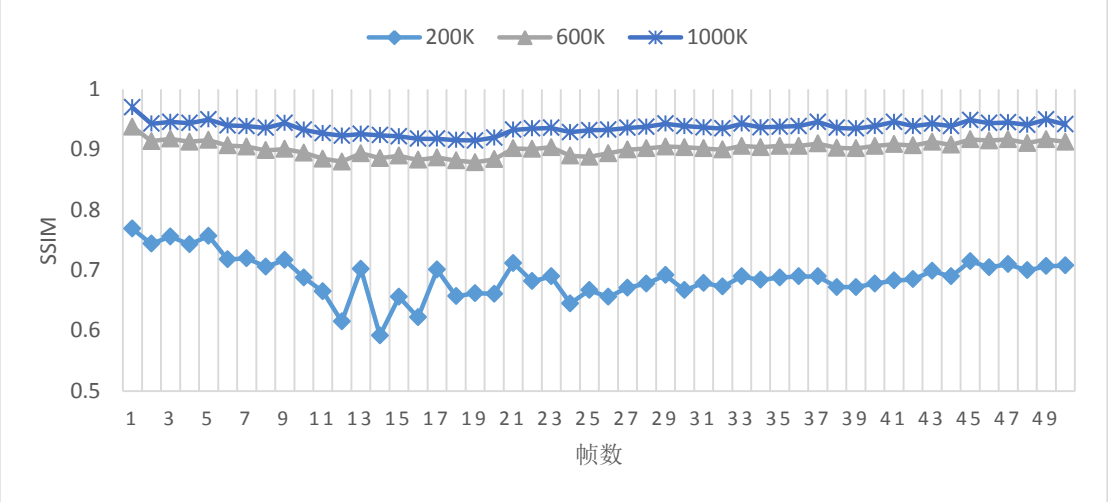

图 2-13 50 帧图像 SSIM 值

Figure2-13 50 frame images SSIM value

PSNR、SSIM 是视频质量客观评价所用的方法，是评价图像的客观标准，单位为 dB(分贝)。通常，图像的 PSNR 值一般在 30~40dB 之间，SSIM 值在 0~1 之间。由以上比较能得到，当 PSNR 与 SSIM 越大时，编码后的视频越清晰，越接近于没有编码的原始视频。也就是说，可以利用图像的 PSNR 和 SSIM 值来代表图像的质量。

2.5 本章小结

本章主要对系统所用到的相关技术进行简单介绍并对系统方案做出选择。首先，

介绍了视频编码相关技术，包括新一代视频编码标准 HEVC 的编码框架以及它所改进的一些关键技术、FFmpeg 开源库的介绍以及对 TS 流进行简单介绍；然后，对系统选用的流媒体协议进行简单介绍，给出系统所采用的流媒体服务器开源框架；最后，对视频质量评价以及所用的方法进行介绍。

第三章 教育系统的总体设计

从国家提出“十五”计划多年来，对高等院校在日常教学研究，日常管理和生活标准化服务相关的所有信息资源，采取先进的网络技术，通信信息得到更好的辅助，最终整合这些数据集成的“数字校园(Digital Campus)[43]”，是高等院校建设信息化项目的重点。中国高等教育机构“数字化校园”是中国教育信息化的三大问题之一。以网络技术为核心，从高效环境的外部条件、信息、活动，实现网络和高效率，这是“数字校园”的一般概念。“数字化校园”的中枢构造是由校园网与其所在的应用系统组成，从而实现整个学校的信息汲取与传输。校园网通俗来讲就是一个为校园里面的师生和其他管理人员提供日常网络服务、日常教学服务、校方科研人员进行研究开发与智能网络服务的基于局域网的多维度网络。

教育系统是以校园网为载体，通过视频直播的方式来实现网上教学，打破传统教育对地点和时间的限制。系统采用视频直播的形式实现远程教育，互联网上现有的视频直播系统都是采用 H.264 视频编码标准，而本系统采用 H.265 视频编码标准。H.265 相对于 H.264 提高了压缩效率、鲁棒性和错误处理能力以及减少实时的延时和接入时延。在相同的视频质量下，H.265 具有更低的码率，可以实现利用 1~2Mbps 的传输速度传送 720P(分辨率 1280*720)普通高清音视频传送。

3.1 系统框架

本系统是基于 C/S 模式进行设计，主要由视频采集上传端、实时视频流媒体转发服务器、RTSP 信令控制平台及播放器组成。视频上传端将采集的原始视频结合一些必要的通信协议及视频处理协议，最终完成端到服务器之间的信令交互、视频的压缩、TS 流封装、RTP 打包和发送。实时视频流媒体服务器配合 RTSP 信令控制平台完成端到端的链路建立、维护链路信息，使得视频流能在端到端之间高效转发。而 RTSP 信令控制平台作为系统通信协议实现的关键部分，管理系统中各个终端，主要负责处理前端设备上下线和播放器的视频播放请求等会话功能，根据不同信令请求来控制实时视频流媒体的转发工作。播放器在本系统中选择支持 RTSP 协议的开源播放器为主，如 VLC 播放器，降低了开发成本。这种平台化的设计既方便了对移动视频采集端和播

放器端的统一管理，又能增强系统整体的负载能力。系统框图如图 3-1 所示：

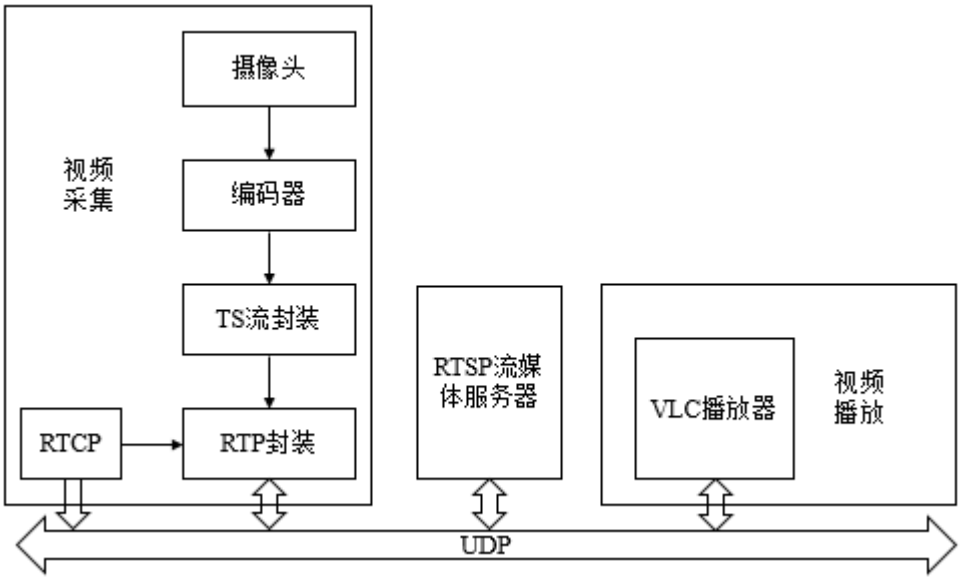

图 3-1 系统总体框图

Figure 3-1 system block diagram

在本系统中对采集的原始视频压缩编码成 H.265 视频流之后，再对编码出来的 H.265 数据做了一次 TS 封装。根据 MPEG-2 TS 标准，将编码出来的 H.265 视频流作为 ES 流，再将 ES 打包成 PES 包，本教育系统选择用 TS 流对编码后的视频进行封装。TS 流能够将一个或者多个音视频以及其他的基本数据流整合成单个数据流，以便适应系统数据的存储和传输，这样能够避免传输过程中由于网络环境引起的丢包不同步问题。本系统未涉及到音频相关业务研究，音频作为今后的拓展业务也可以很方便的接入到本系统中而不需要太多额外的开销，而且能够解决现有互联网视频直播系统分开传输视频与音频可能造成的视频与音频不同步问题。系统选用具有很强的网络穿透能力的 RTSP 协议，能够很好的与已存在的一些主流播放器相结合。

3.2 视频上传客户端设计

视频上传客户端根据系统的需求不同，分解成不同模块来实现，通过内部功能的封装提供底层的实现来完成各个模块之间的调用。

3.2.1 视频采集

视频采集模块主要功能是实现对视频图像的捕捉，本教育系统面向学校，实现基于低成本，能够满足普通用户的需求。因此，在本系统中选择 USB 摄像头做为视频采集的硬件设备。USB 摄像头普及率非常广，而且价格低廉，即插即用十分方便。其次，随着计算机的发展，CPU 的处理能力越来越强大，USB 速度也越来越快，USB2.0 可以达到 480M，USB3.0 更是还要再高 10 倍以上。因此，使用现有 PC 直接处理摄像头数据已经完全没有问题。

3.2.2 视频编码

视频编码是整个系统比较关键的业务之一，系统调用 FFmpeg 对视频进行编码。系统采用 H.265 视频编码标准，大大提高了压缩效率、鲁棒性和错误处理能力以及减少实时的延时和接入时延。

3.2.3 TS 流封装

参照 MPEG-2 标准，将编码后的 H.265 码流封装成负荷数据为 H.265 码流的 TS 流，这样好处是为以后扩展提供方便，将音频和视频数据到一个传输流中，而无需增加太多额外的开发和维护成本，并且 TS 流在比较复杂、容易产生丢包的网络环境中传输比较有优势。

3.3 流媒体服务器设计

流媒体服务器包括两个主要业务：RTSP 交互服务和流媒体转发服务。RTSP 是整个系统的关键部分，主要负责消息处理和视频会话管理系统。例如，播放器的用户认证、回放、暂停、结束请求等操作。流媒体转发服务是采用模块化设计，按功能分为流媒体接收、流媒体发送和 RTCP 控制三个模块。流媒体服务器的模型框图如图 3-2 所示：

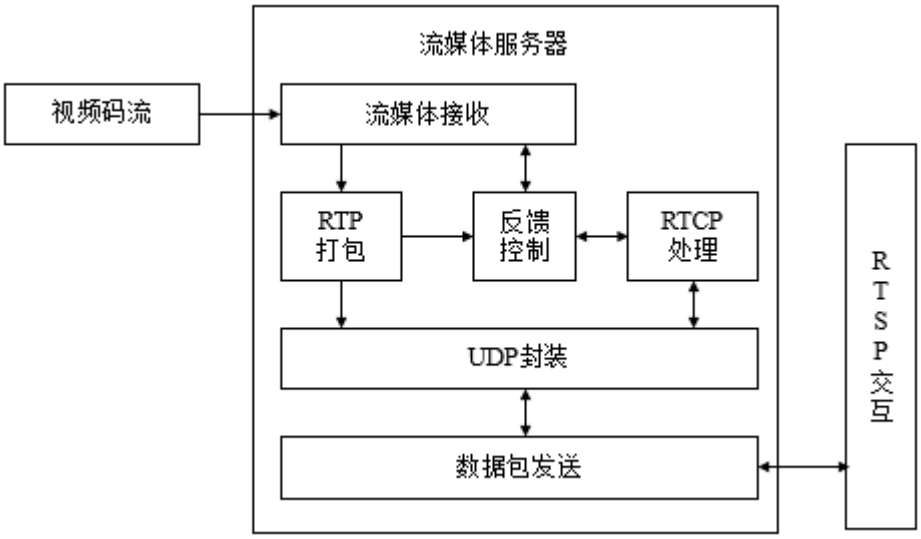

图 3-2 流媒体服务器框图

Figure 3-2 Streaming Media Server Block Diagram

流媒体接收模块可以根据不同媒体格式划分为若干子模块，分别处理来自不同媒体的数据，它只负责媒体数据的来源，不关心媒体数据的去向，不管媒体数据是否被转发，或者是中间被处理、过滤等，数据源可以来自本地文件，也可以是网络数据流。在本系统中，数据源是网络数据流，因此它只需要在允许的缓冲区空间中接收数据并将其提供给其他模块。

流媒体转发只需要负责将流媒体数据发送到对应的播放器，而不用关心数据的类型、格式以及数据来源。系统首先将原始视频压缩为 H.265 流，然后封装 TS，然后执行 RTP 封装，最后执行 UDP 封装。因此，流媒体转发模块负责将输入 TS 流转换为 RTP 分组并发送到对应的播放器以完成任务。

最后，RTCP 模块负责在服务器和播放器客户端之间传输质量控制信息，以确保传输质量的可靠。针对播放器到服务器端以及视频采集端到服务器的网络情况做出反馈，使服务器和客户端做出适当的调整。

3.4 本章小结

本章主要介绍了教育系统的总体构架设计，并且介绍了视频上传客户端所包含的各个模块以及各个模块的功能设计。然后，介绍了本系统流媒体服务器框架设计，以及各模块的功能。

第四章 视频上传客户端设计实现

视频上传客户端在本系统中起到至关重要的作用，主要包括视频的采集、视频的压缩以及对压缩后视频进行 TS 封装。首先系统采用摄像头对视频数据进行捕捉，然后采用 FFmpeg 库对采集的视频进行编码，最后再对编码后的视频数据进行 TS 封装，这样能够提高网络适应性也为方便以后对音频模块的扩展。

4.1 FFmpeg 开源库

4.1.2 FFmpeg 库的配置

FFmpeg 是在 Linux 中开发的开源项目，但在官方网站中有编译好的 windows 版本，共分为三个版本：Static 版、Shared 版以及 Dev 版。其中，Static 版和 Shared 版可以直接利用命令行的形式使用，但它们也是有区别的，Static 版中只包含编译好的三个应用程序：ffmpeg.exe、ffplay.exe、ffprobe.exe，由于相关的链接库都已经编译在应用程序里面了，所以它们体积都很大。Shared 版本中则是将应用程序和动态链接库分开，因此里面除了三个应用程序之外，还包含很对 dll 文件。最后，Dev 版属于开发版本，里面不包含应用程序文件，而是包含库文件以及头文件，用于开发使用。

参照以上三个版本，我们从官方网站上下载 Shared 版和 Dev 版，利用其中的库文件、头文件以及 dll 文件，将它们都拷贝到 VC 工程下就可以使用。

4.2 视频采集

本系统采用 USB 摄像头作为前端视频采集设备，USB 摄像头现在基本已经完全普及，而且价格低廉。利用 FFmpeg 开源库实现视频的采集，在 FFmpeg 中包含有 Libavdevice 类库，其作用是与多媒体设备进行交互，可以读取与电脑相连接的多媒体设备数据。在使用 libavdevice 类库读取多媒体设备数据与直接读取本地数据十分相似，因为系统的设备也被 FFmpeg 认为是一种输入的格式(即 AVInputFormat)。不同在于使用 libavdevice 类库时，首先需要查找与电脑连接的用于输入的多媒体设备，在这里使用 av_find_input_format()完成：

```
AVFormatContext *pFormatCtx = avformat_alloc_context();
AVInputFormat *ifmt=av_find_input_format("vfwcap");
avformat_open_input(&pFormatCtx, 0, ifmt,NULL);
```

上述代码首先指定了 vfw 设备作为输入设备，然后在 URL 中指定打开第 0 个设备（在我自己计算机上即是摄像头设备）。

4.3 视频编码

4.3.1 H.265 编码概述

H.265 是新一代视频编解码标准，它具有高压缩编码效率和高可靠的网络适应性，与 H.264 类似，H.265 将整个编码算法标准分为网络抽象层 NAL 与视频编码层 VCL 两层。其中视频编码层 VCL 通过时域、空域预测和变换编码来实现对视频的压缩，并进行必要的标识，使其可以很好的适应复杂多变的网络环境。而网络抽象层 NAL 则将与网络相关的信息从视频压缩系统中抽象出来，从而使视频编码对于网络来说是透明的，它主要完成对编码后的视频数据进行封装，以提高其鲁棒性、降低误码率。这种分层结构可以实现压缩编码与网络传输分离，有效的解决了视频服务质量和网络服务质量的问题，使视频编码层可以移植到不同结构的网络中，适应不同的网络传输。

视频编码技术的共同目标就是在尽量低的带宽下提供尽量高的图像质量，相对于 H.264，H.265 在很多方面有了革命性的变化。

在 H.265 中，将宏块的大小从 H.264 的 16×16 扩展到了 64×64，以便于高分辨率视频的压缩。同时，采用了更加灵活的编码结构来提高编码效率，包括编码单元（Coding Unit）、预测单元（Predict Unit）和变换单元（Transform Unit）。加入自适应的变换技术，这种思想是对 H.264/AVC 中 ABT（Adaptive Block-size Transform）技术的延伸和扩展。对于帧间编码来说，它允许变换块的大小根据运动补偿块的大小进行自适应的调整；对于帧内编码来说，它允许变换块的大小根据帧内预测残差的特性进行自适应的调整。大块的变换相对于小块的变换，一方面能够提供更好的能量集中效果，并能在量化后保存更多的图像细节，但是另一方面在量化后却会带来更多的振铃效应。

现如今视频业务越来越向高清方向发展，H.264 编码标准的局限性也日益显现出来。H.264 的编码单元是按宏块来划分的，大小固定都是16×16像素。因此，当视频

分辨率不断提高，每个宏块代表的信息量减少了，而经过变换后提高了低频系数的相似程度，这样就造成了信息的冗余，直接导致编码效率下降。其次，宏块个数也会随着分辨率的提高呈现快速增长，导致每个宏块的预测模式、参考帧索引以及运动矢量会占用更多资源，在带宽一定的情况下，图像残差信息所占资源就会减少。

H.265 编码标准与 H.264 相比，采用四叉树编码方式是最主要的改变，自适应的划分宏块大小，H.265 的编码单位可以选择从最小的8×8到最大的64×64。如下图 4-1 与图 4-2 所示，像素变化比较平坦的区域，比如图中的树叶部分，大部分都是绿色，这样的区域编码时划分的宏块就比较大，而图中细节变化比较明显的区域，在图中的人群部分划分的宏块就小而且多。这样做的目的是能够按照像素的细节变化来重点编码，从而提高编码效率。

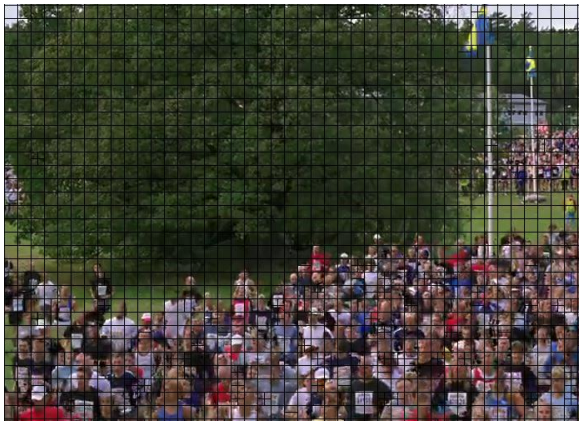

图 4-1 H.264 编码一帧示意图

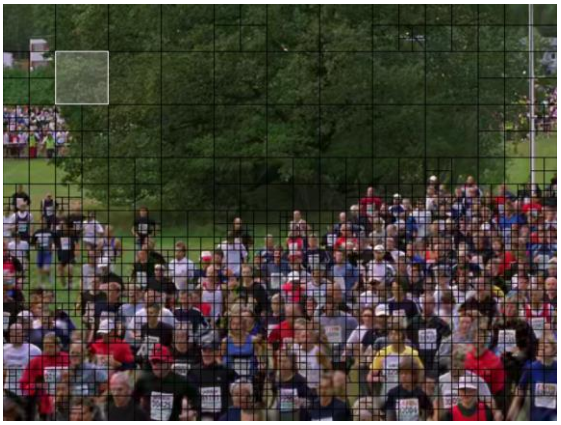

图 4-2 H.265 编码一帧示意图

Figure 4-1 H.264 encoding a frame diagram Figure 4-2 H.265 encoding a frame diagram

4.3.2 FFmpeg 编码视频

本系统调用 FFmpeg 开源库对采集的原始视频数据进行压缩编码，FFmpeg 支持流媒体的压缩编码、解码、格式转换以及流媒体的录制，为视音频开发提供一种完整的解决方案。FFmpeg 中包含 libavcodec 组件，其中包含了多种编码和解码模块，可以完成对视频的编码工作。FFmpeg 支持调用 libx265 将原始视频编码成 H.265 码流，编码流程如下图 4-3 所示：

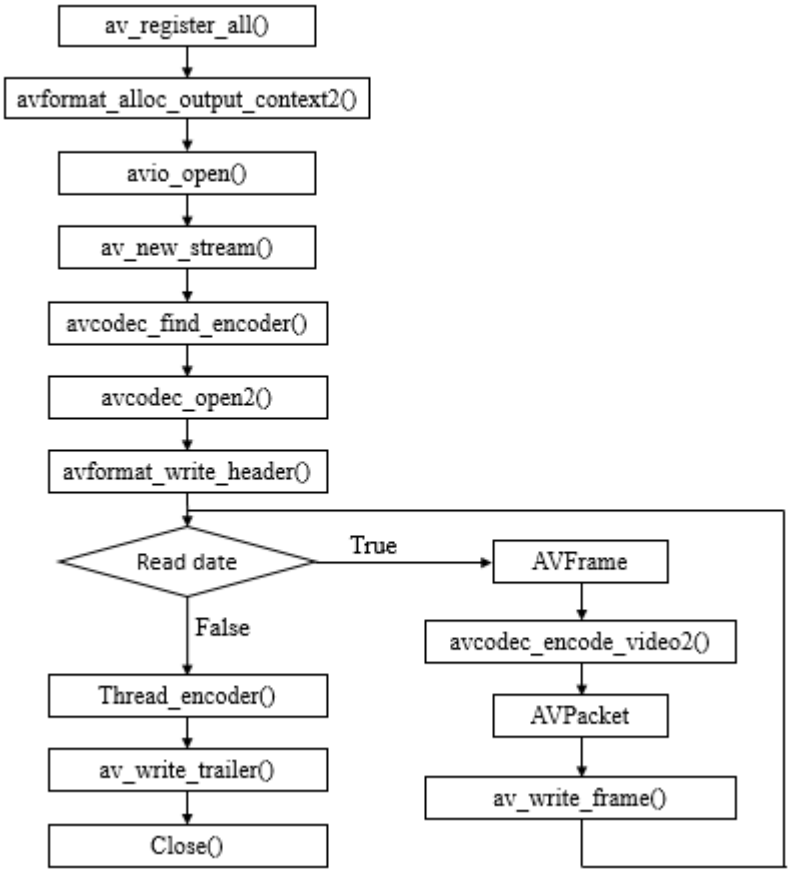

图 4-3 FFmpeg 编码流程

Figure 4-3 FFmpeg encoding process

流程图中主要的函数如下所示：

av_register_all()：注册 FFmpeg 所有编解码器。

avformat_alloc_output_context2()：初始化输出码流的 AVFormatContext。

avio_open()：打开输出文件。

av_new_stream()：创建输出码流的 AVStream。

avcodec_find_encoder()：查找编码器。

avcodec_open2()：打开编码器。

avformat_write_header()：写文件头。

avcodec_encode_video2()：编码一帧视频。

av_write_frame()：将编码后的视频码流写入文件。

av_write_trailer()：写文件尾。

在调用 FFmpeg 库对采集的视频进行编码之前，首先要初始化 FFmpeg 所有编码

器，注册编码器，设置编码器参数。然后查找编码器并打开编码器，打开编码器之后才可以对视频进行编码。通过设置结构体 AVCodecContext 中成员函数来完成参数是设置，例如，通过设置 AVCodecContext->bit_rate 来设置视频的码率，通过设置 AVCodecContext->width 和 AVCodecContext->hight 来设置视频的宽度和高度等参数，通过设置 AVCodecContext->PIX_FMT_YUV420P 来设置视频为 YUV420 格式。

由以上分析看出编码的核心函数为 avcodec_encode_video2()，系统每采集一帧视频数据就传给 avcodec_encode_video2()函数编码成 H.265 视频流。该函数调用关系如下图 4-4 所示：

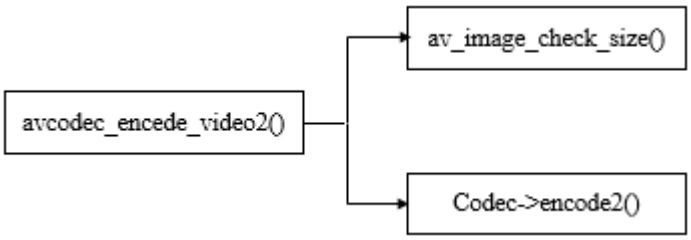

图 4-4 函数调用关系图

Figure 4-4 Function call diagram

首先调用 av_image_check_size()检查设置的宽高参数是否合理，然后调用了 AVCodec 的 encode2()调用具体的编码器。encode2()是一个指针函数，指向特定的编码函数，本系统采用 H.265 编码，则是调用 libx265，对应的 AVCodec 的结构体的定义，如下所示：

```
AVCodec ff_libx265_encoder = {
    name            = "libx265";
    id              = AV_CODEC_ID_HEVC;
    init            = libx265_encode_init;
    encode2         = libx265_encode_frame;
    defaults        = x265_defaults;
    …… };
```

由以上结构体定义可以看出 encode2()函数指向 libx265_encode_frame()函数，在 libx265_encode_frame()函数中，VCL 层编码过程主要在 x265_encoder_encode()函数中完成，分成四步：第一步相关参数的初始化，包括帧类型获取、写码流结构的初始化、

码流控制获取当前帧编码量化步长、写片头序列参数集(SPS)、图像参数集(PPS)；第二步进行以宏块为单位对图像帧数据进行编码；第三步为保存编码过程的中间产物信息，以保存编码器的状态信息；第四步对已经编码完成的一帧图像进行去块效应、环路滤波等处理。VCL 编码最核心的部分就是以宏块为单位对图像进行压缩编码，需要大量的矩阵运算，是整个编码过程中最耗时的部分。

NAL 主要是将已经完成的 H.265 视频帧进行打包封装，不让 VCL 视频编码与网络相关的信息相关联，提高码流的健壮性以及更能适应网络传输，降低误码率。在 x265 库中，可以通过库函数 x265_nal_encode()来实现 NAL 打包。流程图如下图 4-5 所示：

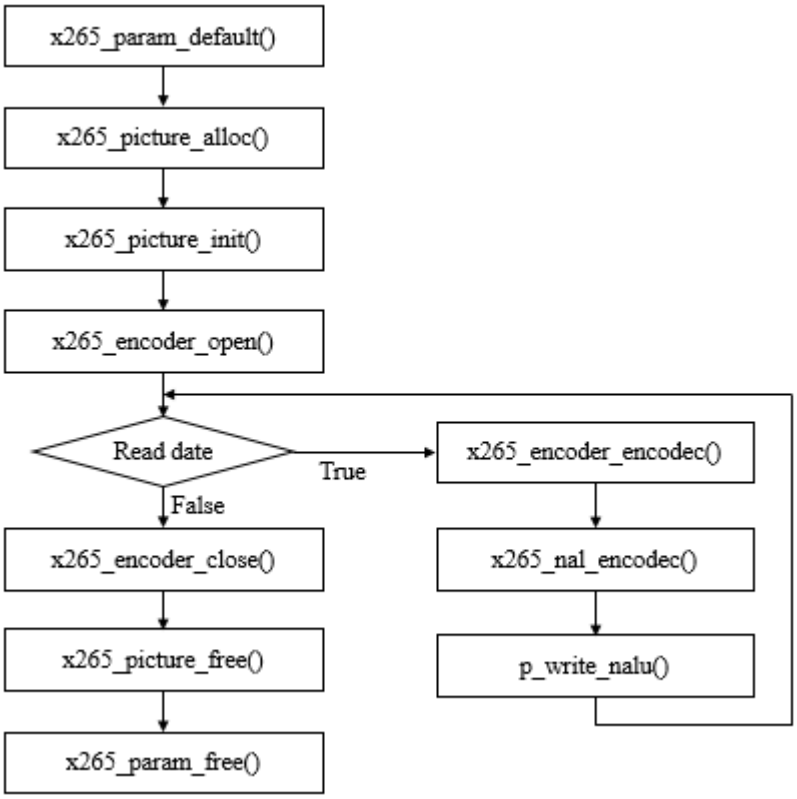

图 4-5 x265 编码流程图

Figure 4-5 x265 coding flow chart

4.4 TS 封装

4.4.1 TS 包结构分析

1. PES 包结构

PES 是 Packetized Elementary Stream 的简称，是用来传递 ES 的一种数据结构。PES

流是将 ES 流经过 PES 打包器打包后形成的，在打包过程中完成的基本操作包括 ES 流分组和加入包头信息等操作。PES 流的基本单位是 PES 包，PES 包的开头为 32 位码字，主要作用是辨别 PES 包中码流的性质。PES 包的组成结构如图 4-6 所示：

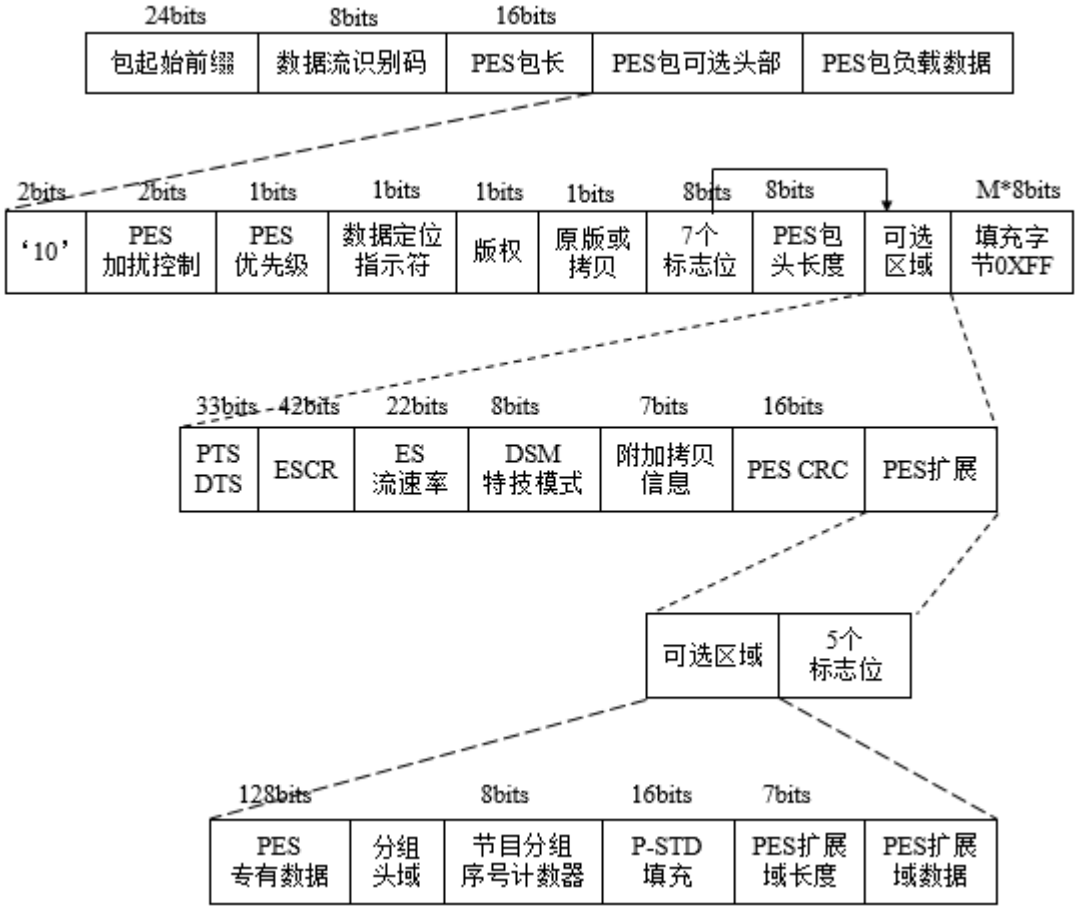

图 4-6 PES 组成结构

Figure 4-6 PES composition structure

2. TS 流

TS 流的传输包长度固定，一般为 188 字节。每个 TS 数据包都由包头、自适应区和有效负载三部分组成，而 PES 包则是作为 TS 包的有效负载加以传输，传输流数据包有效负载的第一个字节也就是每个 PES 包的包头的第一个字节。TS 包结构如图 4-7 所示：

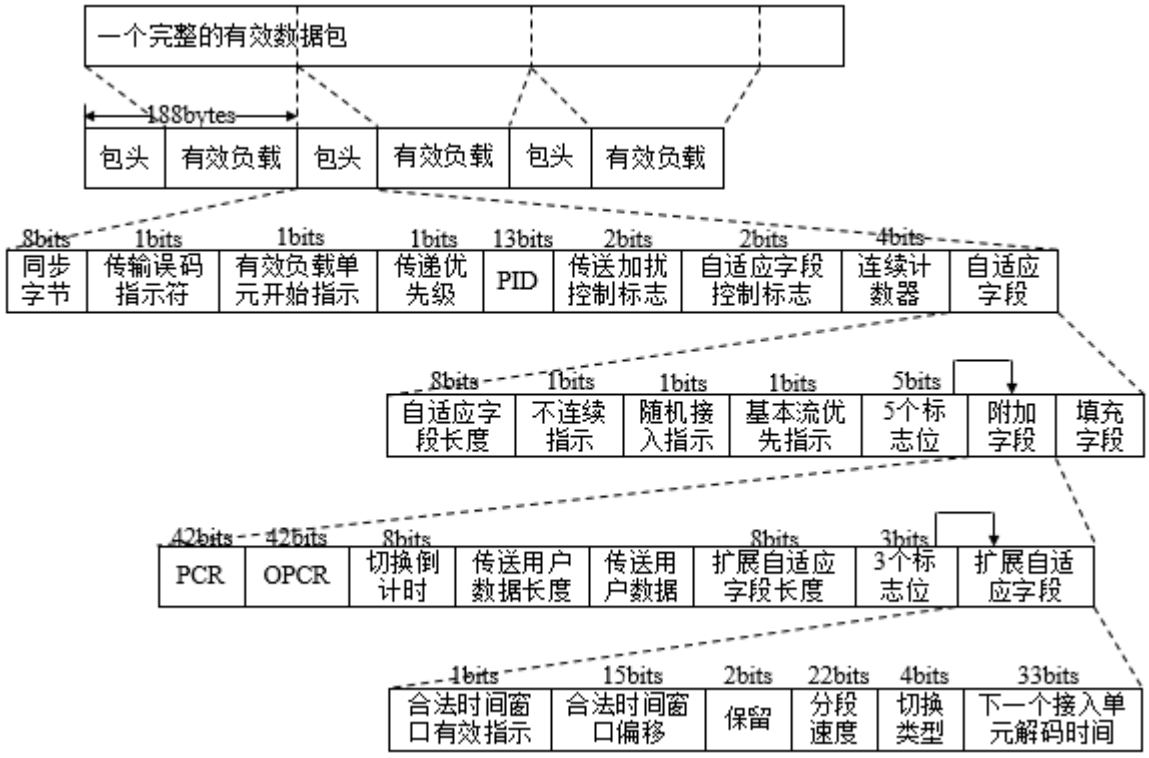

图 4-7 TS 包结构图

Figure 4-7 TS package structure

TS 流的形成过程：

1. 对原始数据流进行压缩编码，形成一个连续的 ES（基本码流）。
2. 将 ES 进行打包形成 PES。
3. 在 PES 包中加入定时信息(PTS/DTS)。
4. 将 PES 包分配到一系列固定长度的传输包中。
5. 在传输包中加入定时信息(PCR)。
6. 在传输包中加入节目专用信息(PSI) 。
7. 连续输出的传输包形成具有恒定比特率的 MPEG-TS 流。

4.4.2 FFmpeg 实现 TS 流打包

系统将原始视频编码成 H.265 码流，然后将 H.265 码流进行 TS 打包发送，而不是直接将其打包成 RTP 包发送，这样便于网络传输和业务扩展。在 FFmpeg 库中给出几种文件格式转换，但本系统需要转换的是实时的输入 H.265 视频帧数据，所以需要对读入文件时的操作更改成读取 H.265 视频帧，此功能由 init_packet()函数完成；封装

完之后写入到文件操作时，还需要将 TS 数据包发送到网络服务器，send_RTP_WithTS-Packet()完成发送。转换流程图如 4-8 所示：

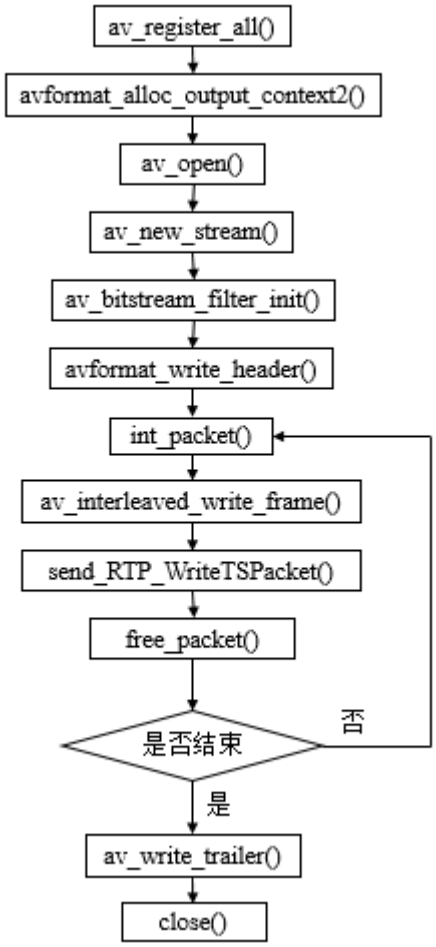

图 4-8 ES 流封装成 TS 流

Figure 4-8 ES stream encapsulated into TS stream

4.5 本章小结

本章主要介绍视频采集端详细设计实现，包括视频的采集、视频编码以及对编码后的视频进行 TS 封装。首先对所用到的 FFmpeg 开源库进行介绍以及使用前的配置，本系统采用 USB 摄像头采集视频流，然后利用 FFmpeg 开源库对视频进行编码，然后介绍了 TS 包结构分析并对编码过的视频数据进行 TS 封装，完成视频采集端工作。

第五章 流媒体服务器的实现

流媒体服务器是基于 Live555 开源库进行二次开发，实现满足系统业务需要的流媒体服务器是本章的重点。在系统传输中，流媒体服务器主要负责从视频上传客户端获取视频数据，并对获取后的数据流进行 RTP 包处理。另外，流媒体服务器使用 RTSP 交互协议来控制服务器端与播放客户端的会话请求，当收到来自客户端的请求时，流媒体服务器将打包好的 RTP 数据包经过 Socket 套接字发送出去。

5.1 Live555 介绍

随着计算机技术和流媒体技术的发展，不断出现各种流媒体服务器开源库，但 Live555 是其中代码最精简的流媒体服务器，而且支持 RTSP、RTP/RTCP 等流媒体协议，并且可以很方便地应用到嵌入式当中。Live555 是开源的、跨平台的流媒体完美解决方案，支持多种音视频编码格式的多媒体数据的流化、接收和处理，包括 H.265、H.264、MPEG 等视频和多种音频编码格式[44]。而且由于 Live555 良好的结构化设计模式，能够很容易扩展到其他的多媒体格式。Live555 开源协议栈中主要包括四个基本组件[45]，分别为 Usage Environment、BasicUsageEnvrionment、groupsock、LiveMedia，Live555 的基本框架如图 5-1 所示：

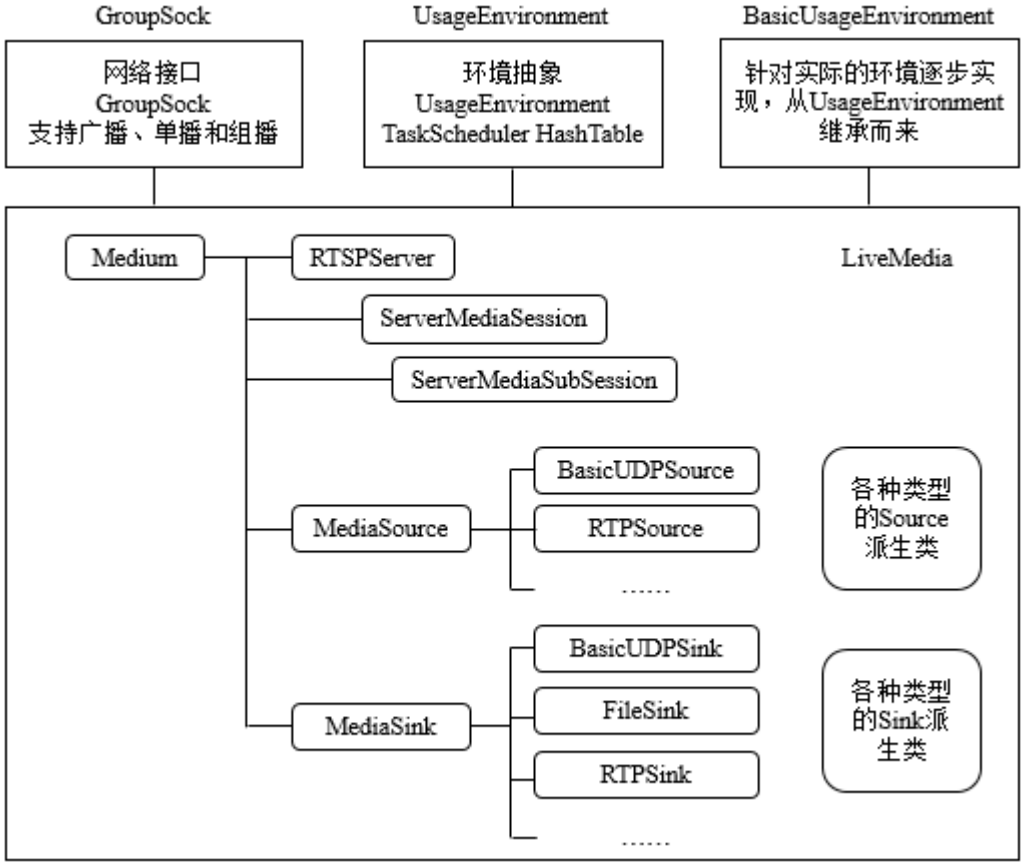

图 5-1 Live555 基本框架图

Figure 5-1 Live555 basic frame diagram

GroupSock 里面封装了一系列的网络接口，包括网络地址类、数据包的发送类等等，它原本是为了支持组播，但也可以支持单播功能。

在 UsageEnvironment 中主要包括了 UsageEnvironment 类、TaskScheduler 类和 HashTable 类。UsageEnvironment 用于输入输出操作和错误信息的处理。TaskScheduler 类主要负责整体事件的任务调度，包括回调函数的注册和异步事件的处理。TaskScheduler 类通过 DelayQueue 来实现事件的延时调度，以控制好数据包的发送速率。

BasicUsageEnvironment 类是从 UsageEnvironment 继承而来，针对实际的环境逐步实现具体的输入、输出以及任务调度的处理操作。

LiveMedia 是 Live555 开源库的核心组件，它定义了一个最基本的抽象基类 Medium，可以通过继承这个基类实现不同业务模块。MediaSession、MediaSource、MediaSink、RTSPClient 和 RTCPInstance 等都是派生自这个类。MediaSession 主要负

责建立相关流媒体的 RTP 会话，MediaSession 中可能会包含多个 MediaSubsession，MediaSubsession 为子会话，可以是音频或者视频子会话。MediaSource 作为媒体数据源的抽象，负责从网络或文件获取数据流；MediaSink 属于 Live555 中的基类，所有 Sink 都是派生自 MediaSink 类。Sink 就是数据消费的对象，比如要把接收到的视频或音频数据存储到文件，这个文件就叫做 Sink。Source 就是生产数据的对象，将 Source 获取的数据发送到网络或存储到文件，数据流经过多个 source 和 sink，下面是一个示例：

'source1' -> 'source2' (a filter) -> 'source3' (a filter) -> 'sink'

从其它 Source 接收数据的 source 也叫做"filters"。Module 是一个 sink 或者一个 filter。RTSPClient 主要作用是负责处理 RTSP 交互过程中播放器客户端的请求；RTCPInstance 实现 RTCP 协议控制消息监控网络。因此，LiveMedia 在 Live555 整个系统框架中起着至关重要的作用。

5.2 RTSP 服务器实现

5.2.1 服务器的搭建

Live555 流媒体服务器不仅支持常见的 RTSP、RTP/RTCP 以及 SDP 等流媒体协议，而且支持多种常用多媒体格式的流化与传输，是目前多种流媒体服务器中实现 RTSP 协议最精简的开源代码。Live555 使用 C++语言开发，由于 C++项目跨平台特性，Live555 可以很方便地移植到嵌入式系统中。本教育系统是通过视频直播的方式来实现远程视频教育，因此要实现视频直播需要在 Live555 源代码的基础上，通过继承相关类和重写相关方法来实现我们所需要的功能。

首先，通过继承 FramedSource 类来创建 liveFramedSource 类，并利用创建 liveFramedSource 对象来获取实时数据源。系统的 RTSP 服务由 RTSPServer 类实现，RTSPServer 类用于构建 RTSP 服务器，同时在内部定义 RTSPClientConnection 类来处理各个客户端会话。首先创建 RTSP 服务器(具体实现类是 DynamicRTSPServer)，在创建过程中，先建立套接字在 TCP 的 554 端口进行监听，然后处理函数(RTSPServer::incomingConnectionHandlerHTTP)和套接字句柄连接到任务调度器(TaskScheduler)。TaskScheduler 将套接字句柄放入 select 调用中使用的套接字句柄集中，

并将套接字句柄与 incomingConnectionHandlerHTTP 句柄相关联。然后，主程序进入任务调度主循环，调用 select 系统阻塞函数，等待网络连接。

当播放器客户端输入 URL 向服务器发送会话请求时，select 返回相应的 scoket，然后根据以前保存的对应关系，找到相应的处理函数句柄 incomingConnectionHandler-OnSocket。然后 incomingConnectionHandlerOnSocket 调用 createNewClientConnection() 来创建 RTSPClientConnection，开始处理客户端的会话。RTSP 服务器搭建流程图如下图 5-2 所示：

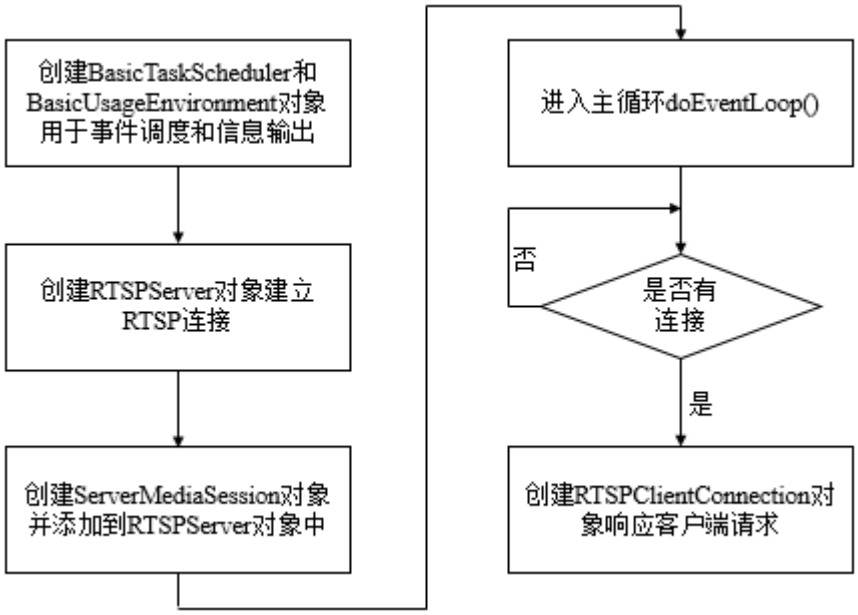

图 5-2 流媒体服务器搭建流程图

Figure 5-2 Streaming media server flow chart

1. **任务调度**

首先，在构建 RTSP 服务之前，需要创建两个基本对象：TaskScheduler 和 Usage-Environment 派生类对象，TaskScheduler 类和 UsageEnvironment 类都是虚基类，这两个类构建整个 live555 框架。TaskScheduler 类是 Live555 任务调度中心，整个程序运行的引擎，负责任务的调度和执行。而 UsageEnvironment 类定义了系统的操作环境，负责控制台的输入和错误输出。通过调用以下方法创建：

```
TaskScheduler* scheduler = BasicTaskScheduler::createNew();
UsageEnvironment* env = BasicUsageEnvironment::createNew(*scheduler);
```

BasicTaskScheduler 可以用于服务器设置各种调度任务，任务将被添加到延迟队列

中，选择适当的时间进行调度以确保 RTP 的正确传输。然后将 scheduler 的引用传递给 BasicUsageEnvironment 对象，以便可以控制任务类的执行。

2. 建立 RTSPServer

TaskScheduler 任务调度系统任务调度过程非常简单明了，这种设计将不同的业务逻辑最直接的解耦和，只需要将需要处理的业务逻辑加入到调度引擎中，即可高效运行。构建 RTSP 服务利用 SocketHandler 事件队列可以很好的处理来自网络的 socket 请求，实现本系统要求的播放器端到服务端的 RTSP 交互。下面描述如何在 TaskScheduler 任务调度系统上构建 RTSP 服务。

RTSP 服务需要监控播放器和视频数据采集端的 RTSP 连接，使用 TaskScheduler 任务调度机制，需要创建一个 SocketHandler 添加到任务调度系统中处理 RTSP 请求，系统还要对来自网络的 RTSP 请求必要的用户认证，还需要一个用户数据认证管理中心来初始化这个 SocketHandler，以下是创建 SocketHandler 过程。

(1) 初始化用户数据认证管理中心：

```
UserAuthenticationDatabase* authDB= new UserAuthenticationDatabase ("Name");
authDB->addUserRecord ("username", "passwd123");
```

(2) 创建 RTSPServer 对象：

首先，创建了 RTSPServer 类，并选用了标准的 554 和 8554 端口。

```
RTSPServer* rtspServer;
portNumBits rtspServerPortNum = 554;
rtspServer = DynamicRTSPServer::createNew(*env, rtspServerPortNum, authDB);
if (rtspServer == NULL)
{
        rtspServerPortNum = 8554;
        rtspServer = DynamicRTSPServer::createNew(*env,
                rtspServerPortNum, authDB);
}
```

RTSPServer 类负责所有的 RTSP 服务器相关的操作。

```
DynamicRTSPServer::createNew (UsageEnvironment& env, Port ourPort ,
UserAuthenticationDatabase* authDatabase ,unsigned reclamationTestSeconds)
```

```
{
        int ourSocket = setUpOurSocket(env, ourPort);
        ……
}
```

创建 RTSPServer 对象，主要通过 setUpOurSocket()函数创建一个 Socket 连接，并为 Socket 连接设置发送缓冲区的大小，调用系统函数 listen()函数来监听这个端口；为 RTSPServer 对象设置最大并发连接数 LISTEN_MAX_SIZE，默认为 20；把连接处理回掉句柄 incomingConnectionHandlerHTTP 和 Socket 句柄 serverSocket 传递给 taskScheduler 任务调度系统，与此 Socket 处理器相关联的处理事件添加到 fHandlers 中。这样就可以灵活地将 RTSP 服务添加到主程序任务调度系统中，等待 RTSP 客户端连接。

3. 创建 ServerMediaSession

按照不同的流媒体类型选择不一样的 Source 源，转发给相应的 Sink 去消费，形成一条通道。Live555 开源协议中提供了可扩展的框架结构，ServerMediaSession 对象就代表就是一次会话、一个实际的流媒体，可以同时包含视频、音频等信息。ServerMediaSession 对象创建如下：

```
ServerMediaSession*sm = ServerMediaSession::createNew(*env,streamName,
                                                    streamName, descriptionString);
```

同时提供一个 ServerMediaSubSession 子会话，管理着单个数据流，它负责将 liveFramedSource 中获取的采集数据源进行 RTP 封包。然后将 ServerMediaSession 对象添加到 RTSPServer 对象中。代码如下所示：

```
sms->addSubsession(H265VideoFileServerMediaSubsession::createNew(*env,
                        inputFileName,reuseFirstSource));
rtspServer->addServerMediaSession(sms);
```

在 RTSPServer 收到来自客户端的 SETUP 消息时，他会和播放客户端协商数据流通信的细节，如通信协议、地址等。在服务器收到来自播放客户端的 PLAY 请求时，开始启动 RTP 数据流传输。

4. 进入大循环处理事件

```
env->taskScheduler().doEventLoop();
```

doEventLoop()属于消息循环，一旦进入循环就会不断查找延迟事件。当获取到延迟事件的句柄时，便调用相关的处理函数来处理相应的任务。RTSP 服务器的主循环负责相应 RTSP 会话和处理 RTP 传输，实际上调用 BasicTaskScheduler 的 SingleStep()函数来实现。

当主循环 doEventLoop()轮询播放器客户端发来的连接请求时，RTSPServer 对象保存着代表每个播放器客户端的新套接字 clientSocket，然后使用这个套接字与相应的播放器客户端通信。每个客户端将对应一个 RTP 会话，并且每个客户端的 RTSP 请求仅控制其自己的RTP会话，因此需要创建表示每个客户端的RTSP会话的会话类。RTSP服务器调用 createNewClientConnection()函数创建一个 RTSPClientConnect 对象来保存套接字相关属性，以便它能够与 RTSP 客户端通信。以下是 createNewClientConnection()函数的创建过程：

```
RTSPServer::createNewClientConnection(int clientSocket, struct
                                        sockaddr_in clientAddr);
    return new RTSPClientConnection(*this, clientSocket, clientAddr);
```

RTSPClientConnection 对象在收到 RTSP 客户端发来的不同消息后，先将 RTSP 消息解析，获取每个消息名称、请求头域以及请求体。当接收到 DESCRIBE 消息后，它响应客户端的 RTSP 地址、验证用户等。当您收到 SETUP 请求消息时，需要建立会话信息。当接收到 PLAY 时，开始流媒体转发，Sink 会从 Source 源中取数据开始转发。收到 TEARDOWN 消息后，关闭流会话并停止转发作业。doEventLoop()工作流程如下图 5-3 所示：

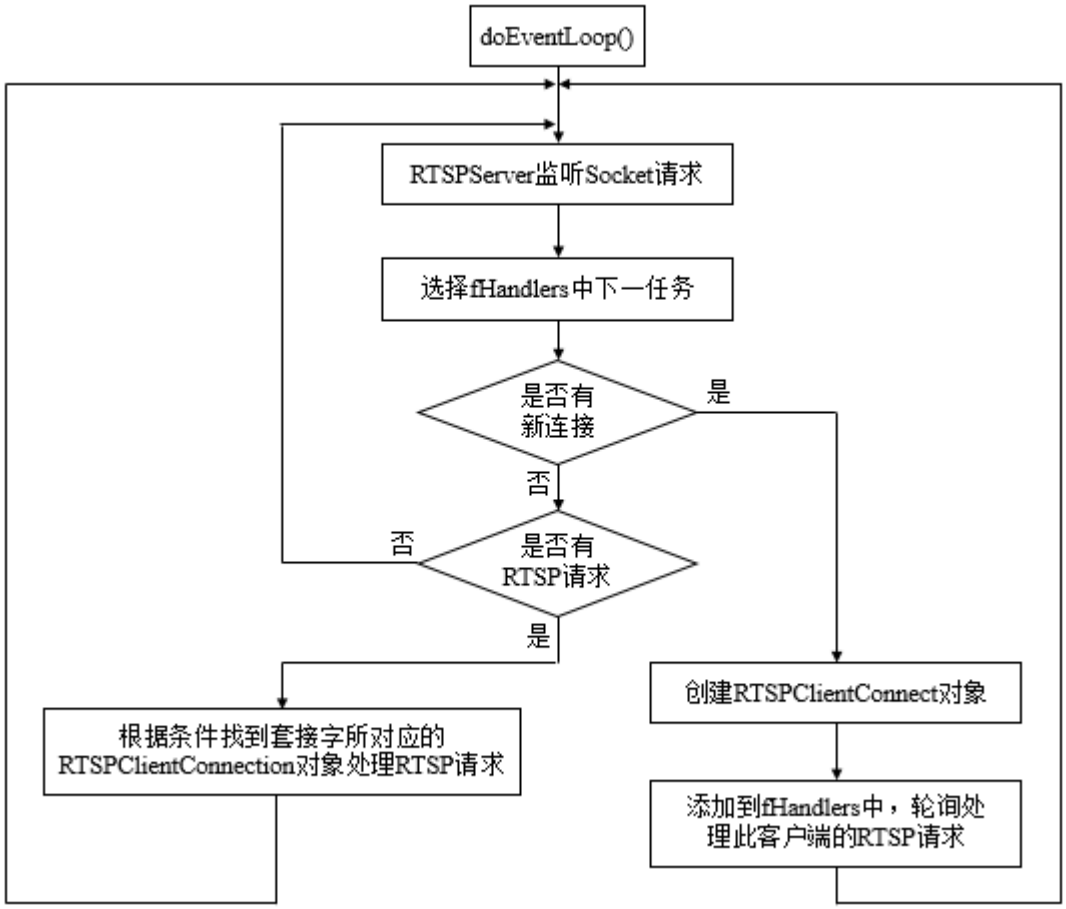

图 5-3 doEventLoop()工作流程图

Figure 5-3 doEventLoop() Workflow diagram

5.2.2 RTSP 交互

本教育系统采用 RTSP 协议作为控制交互消息的流媒体协议，实时流媒体协议用于 C/S 框架模式，它本身不传输数据，只是负责实现连接、播放以及停止等操作。RTSP 协议与 HTTP 协议一样都基于文本的协议，而不同点在于 RTSP 是一种双向协议，服务器端不仅要时刻监听客户端发来的 RTSP 消息，也可以作为客户端向视频采集端发送 RTSP 消息，而视频采集端也可以作为服务器接受来自服务器端发来的 RTSP 消息。播放客户端与服务器交互连接时是通过客户端向流媒体服务器发送的 OPTION、DESCRIBE、SETUP、PLAY、PAUSE、TEARDOWN 等请求来进行交互连接[46]。

常见的交互请求方法[47]如下：

1. 视频上传端向流媒体服务器发送摘要认证信息，成为在线视频直播的前端设备。
2. 播放器客户端向流媒体服务器发起 OPTION 请求，等待服务器端回应，得到

流媒体服务器能够提供的方法。

3. 当流媒体服务器收到客户端发来的 DESCRIBE 请求后，要求对播放器客户端进行用户认证，通过认证服务之后，服务器向客户端返回绘画描述信息，包括所请求的媒体名称、媒体协议、媒体封装格式等。如果未通过用户认证，则返回失败信息并结束请求。

4. 播放客户端向流媒体服务器发送 SETUP 请求，表示播放器客户端希望与流媒体服务器建立会话连接，要求流媒体服务器为会话设置属性。同时包含播放器客户端到流媒体服务端的会话和流媒体服务端到视频上传端的会话，会话内容主要包括：媒体传输方式、流媒体发送端 IP 地址和端口号及会话的唯一标识号等。

5. 流媒体服务器回应播放客户端，请求等待。

6. 播放客户端发送 PLAY 请求，流媒体服务器将来自视频上传端的视频数据转发到相应的 Session 对应的播放端进行直播，并返回流媒体传输信息，包括：RTP 报信息、序列号以及时间戳等。

7. 最后播放客户端向流媒体服务器发送 TEARDOWN 请求，要求停止给播放器客户端发送数据，并释放所占用的资源。

5.3 流媒体转发

5.3.1 流媒体接收

在 Live555 中，MediaSource 属于基类，所有 Source 类都派生自 MediaSource 类，并通过各个派生类实现对各种流媒体类型、格式以及流媒体编码类型的支持。其中 FramedSource 就是派生自 MediaSource，并且内部定义一个虚函数 doGetNextFrame()，用于让其子类各自复写此函数来实现流媒体帧的获取。系统通过继承 FramedSource 类自定义一个 liveFramedSource 类来获取数据源。

当 Sink 实例从 Source 实例索取数据源时，会重复地循环调用 doGetNextFrame()来获取实时数据，并将获得的数据间接传递到 Sink 的缓冲区。最后调用 FramedSource :: afterGetting()函数通知 Sink 端进行处理。等待数据被获取，然后由 PackFrame()函数进行 RTP 打包并发送出去。其流程图如下图 5-4 所示：

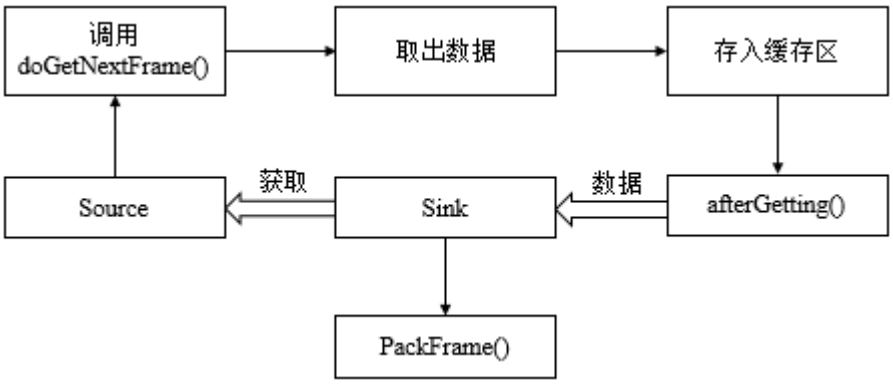

图 5-4 流媒体接收流程图

Figure 5-4 Streaming Media Receive Flowchart

5.3.2 流媒体打包与发送

当播放器客户端向流媒体服务器发送 SETUP 请求报文时，提示流媒体服务器与客户端建立会话以及确定 RTP 包的传输模式，在本系统中默认的是使用 UDP 传输方式。当流媒体服务器收到播放器客户端发来的 SETUP 请求后，服务器创建新的子会话，子会话又会创建新的数据源 Source 和 RTPSink。RTP 数据包的传输开始于播放客户端向服务器发送 play 请求之后，每当 Source 检测到数据时，就发送给 RTPSink。首先调用子会话的流启动函数 startPlaying()，startPlaying()函数内部调用函数 continuePlaying()，接着调用打包函数 buildAndSendPacket()设置好 RTP 包头，然后调用帧填空函数 packFrame()，进而调用 afterGettingFrame1()函数处理好帧的分片，最后调用 sendPacketIfNecessary()来真正实现 RTP 包的传输。具体流程如下：

1. startPlaying()

当客户端发送 Play 请求时，服务器端会调用 handleCmd_Play()方法来响应请求，startPlaying()方法就是在 handleCmd_Play()函数中进行调用，开始触发 RTP 的传输。函数里面还设置了 source 源，以便从其中获取数据。其代码如下：

```
MediaSink::startPlaying(MediaSource& source,afterPlayingFunc* afterFunc,
                        void* afterClientData)
{
    ……
    return continuePlaying();
```

```
}
```

2. continuePlaying()

```
MultiFramedRTPSink::continuePlaying()
{
    buildAndSendPacket(True);
    return True;
}
```

流启动函数startPlaying()返回定义在MediaSink类中的纯虚函数continuePlaying()，此函数在特定的媒体sink子类中实现，因此系统会调用H264or5VideoRTPSink::continuePlaying()函数进行特殊的处理，其代码如下：

```
Boolean H264or5VideoRTPSink::continuePlaying()
{
    if (fOurFragmenter == NULL)
        fOurFragmenter = new H264or5Fragmenter(fHNumber, envir(), fSource,
                                    OutPacketBuffer::maxSize,ourMaxPacketSize() - 12);
    return MultiFramedRTPSink::continuePlaying();
}
```

该函数中创建了一个辅助类 H264or5Fragmenter，主要负责 RTP 封包工作。continuePlaying()函数里面调用 MultiFramedTPSink 类的 buildAndSendPacket()函数，MultiFramedTPSink 是与帧有关的类，它每次从 source 获得一帧的数据。其代码如下：

```
Boolean MultiFramedRTPSink::continuePlaying()
{
    buildAndSendPacket(True);
    return True;
}
```

3. buildAndSendPacket()

由上述代码看出，buildAndSendPacket()函数完全替代 continuePlaying()函数来完成完成 RTP 的打包与发送工作，首先判断传入的参数是否为第一帧，以此来进行后续相应处理工作而。buildAndSendPacket()的代码如下：

```
MultiFramedRTPSink::buildAndSendPacket(Boolean isFirstPacket)
{
    nextTask() = NULL;
    fIsFirstPacket = isFirstPacket;
    unsigned rtpHdr = 0x80000000;
    rtpHdr |= (fRTPPayloadType<<16);
    rtpHdr |= fSeqNo;
    ……
    packFrame();
}
```

如果是第一帧的话就以当前时间作为发送时间，然后填充 RTP 包头。函数中设置了 RTP 包头的版本号、序列号、负载类型以及 SSRC 等信息，预留了时间戳和标志位的位置，它们的填充将由 doSpecialFramehandling()函数来完成。

4. packFrame()、getNextFrame()

packFrame()用于填充真正的数据帧。其函数代码如下：

```
MultiFramedRTPSink::packFrame()
{
    if (fOutBuf->haveOverflowData())
    {
        ……
            afterGettingFrame1(frameSize, 0, presentationTime,
                               durationInMicroseconds);
    }else
    {
        if (fSource == NULL)    return;
            fSource->getNextFrame(fOutBuf->curPtr();
            fOutBuf->totalBytesAvailable();
        afterGettingFrame, this, ourHandleClosure, this);
    }
```

```
}
```

首先判断上次打包的 buffer 中是否有剩下的帧数据，因为一个包可能容纳不下一帧，如果有就使用剩下的数据，并调用 afterGettingFrame1()函数向包中填充数据。如果没有的话就跟 source 要数据，调用 getNextFrame()再去取数据。

5. afterGettingFrame1()

afterGettingFrame1()函数负责向包中填充数据，如果一帧的缓冲不够大，就会发生截断一帧数据的现象，如果包已经打入帧数据了，并且不能再向这个包中加数据了，就要把新获得的帧数据保存，此外还要计算获取的帧中有多少数据可以打到当前包中，并且把剩下的数据作为 overflow 数据保存，对有的一些负载格式还需要做特殊处理。代码如下：

```
MultiFramedRTPSink::afterGettingFrame1()
{
    if (numFrameBytesToUse == 0 && frameSize > 0)
        sendPacketIfNecessary();
}
```

6. sendPacketIfNecessary()

```
MultiFramedRTPSink::sendPacketIfNecessary()
{
    fRTPInterface.sendPacket(fOutBuf->packet(), fOutBuf->curPacketSize());
    ……
    nextTask() =envir().taskScheduler().scheduleDelayedTask(uSecondsToGo，
                                        (Task Func*)sendNext,this);
}
```

如果 packet 中有数据则调用 fRTPInterface.sendPacket ()发送 RTP 包，并计算发送 RTP 包所需要的发送时间，把 MultiFramedRTPSink::sendNext()作为一个 DelayTask 添加到 TaskScheduler 任务调度器中，作为一个延时事件调度，等发送完当前 RTP 时才去发送下一个 RTP 包，以减小网络拥塞，保证发送速率的平滑性。RTP 打包与发送的流程图如图 5-5 所示：

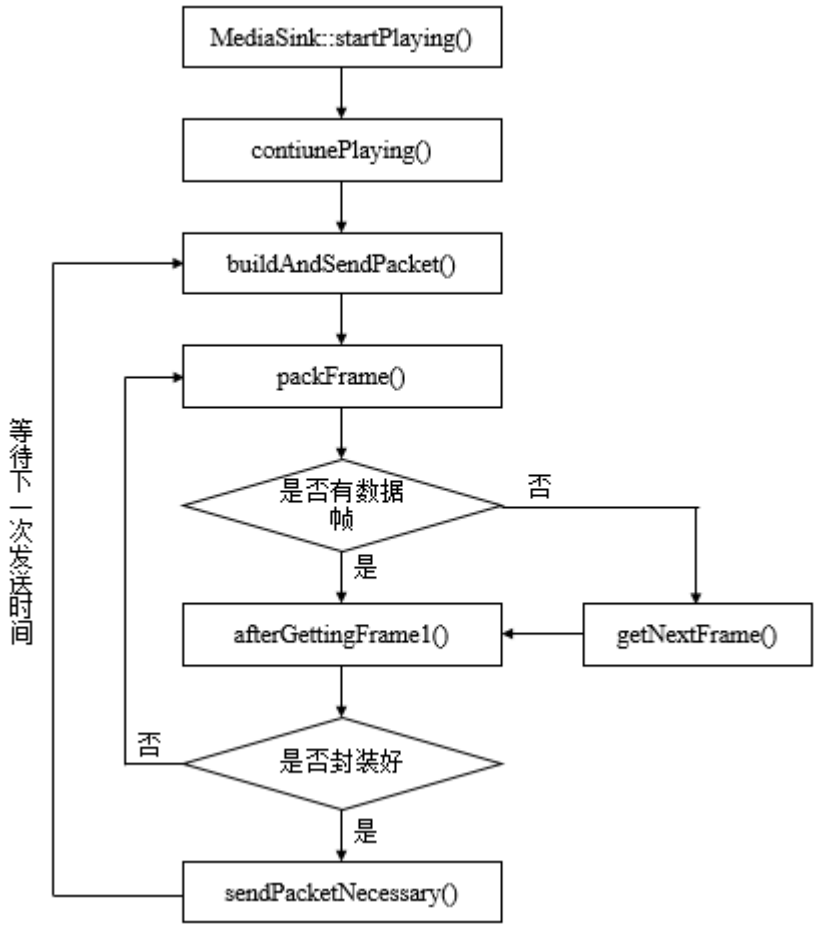

图 5-5 RTP 打包与发送的流程图

Figure 5-5 RTP package and send the flow chart

5.4 本章小结

本章详细的介绍了流媒体服务器的搭建，首先介绍了流媒体服务器搭建的大致流程，然后介绍 RTSP 协议的交互方式，再根据播放客户端对服务器的 RTSP 请求实现流媒体接收模块以及流媒体转发模块的设计。

第六章 系统测试

以上章节介绍了系统的整体设计及详细的开发过程,本章对系统的运行进行介绍，并完成功能测试，最后对系统功能测试的结果进行分析。

6.1 系统的软硬件环境

本系统采用微软 VS2010 进行开发，以 VLC 开源播放器作为 RTSP 网络串流播放器代表进行测试。

硬件配置：视频上传客户端采用联想笔记本，因特尔双核处理器，主频 1GHz，内存 4GB，操作系统为 windows10，视频采集利用笔记本自带的 720P 摄像头来捕捉视频。服务器端也是采用联想笔记本，因特尔双核处理器，主频 2.1GHz，内存 4GB，操作系统为 windows10 最新版。

6.2 视频编码质量评估

6.2.1 PSNR 值测试比较

本教育系统采用 H.265 视频编码标准，采用上文所介绍的视频客观评价标准对经过 H.265 编码后的视频以及目前主流视频直播系统所采用的 H.264 标准编码后的视频进行质量评估。

为了方便掌握所采集视频的帧数，本次评价将系统视频采集的原始视频先不进行编码，而是直接保存在测试所用的电脑上。通过测试电脑摄像头采集四段视频保存在本地，然后利用 YUV 播放器将所采集的四段 YUV 视频剪切成相同的 100 帧，将这四段视频进行编号为序列 1、序列 2、序列 3 以及序列 4。然后利用 FFmpeg 调用 libx265 和 libx264 分别对四段视频进行编码，由于 x265 和 x264 是属于同源编码器，因此 FFmpeg 调用 libx265 和 libx264 在默认参数下的编码效果最为接近，所以 HEVC 同 H.264 的比较，我们选择在默认参数，默认速度下编码并进行比较。对每段视频分别按码率为 1M 至 6M 进行编码，然后计算出各个码率的 PSNR 值。下图 6-1、6-2、6-3、6-4 分别为四段视频序列在不同码率下的 PSNR 比较。

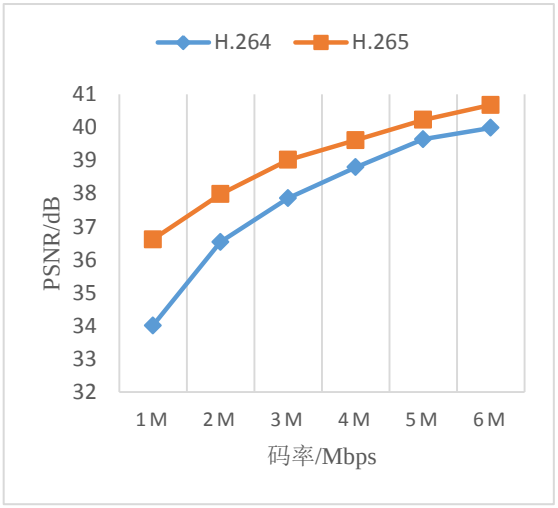

图 6-1 序列 1PSNR 比较

Figure 6-1 Sequence 1PSNR comparison

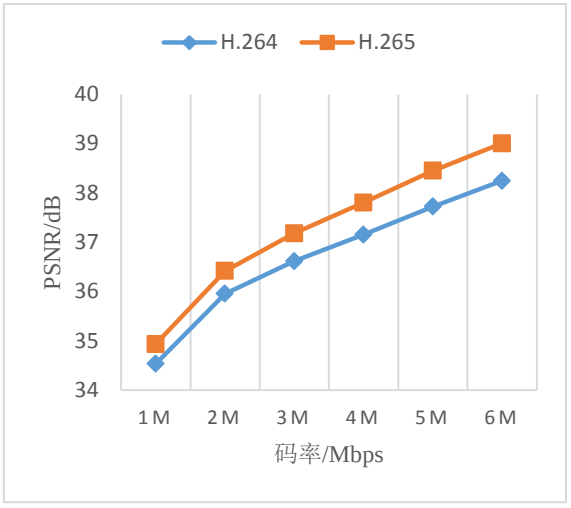

图 6-2 序列 2 PSNR 比较

Figure 6-2 Sequence 2PSNR comparison

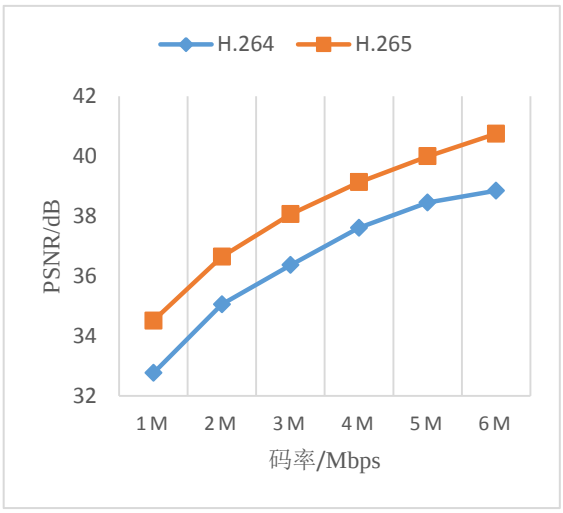

图 6-3 序列 3PSNR 比较

Figure 6-3 Sequence 3PSNR comparison

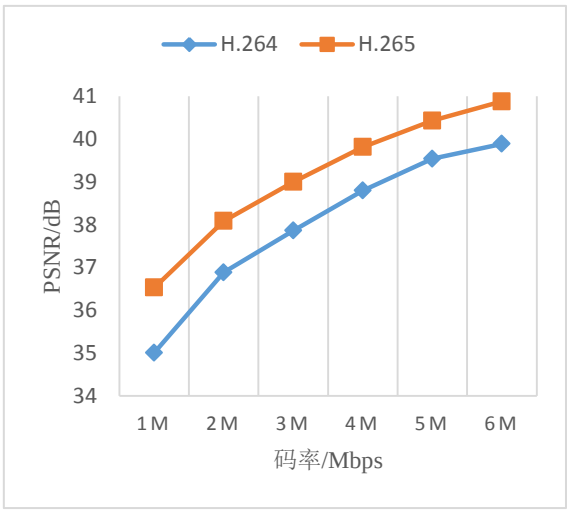

图 6-4 序列 4PSNR 比较

Figure 6-4 Sequence 4PSNR comparison

6.2.2 测试比较结果

在以上测试中，H.264 以编码码率 5M 为基准，通过查询以上各图中 H.265 的 PSNR 值，在表 6-1 中列出：

表 6-1 H.265 与 H.264PSNR 比较

Table 6-1 Comparison of H.265 and H.264PSNR

测试序列	H.265 码流	H.264 码流	码率节省
序列 1	3.9M	5M	22%
序列 2	3.7M	5M	26%
序列 3	3.2M	5M	34%
序列 4	3.6M	5M	28%

由上表数据能够看出，在相同的 PSNR 下，即相同的视频质量下，H.265 码率下降约 20%-35%左右。尽管本次测试在一台电脑上完成，有一定的局限性，但能够表明本系统采用 H.265 视频编码标准比目前采用 H.264 视频编码标准的网络直播系统在码率上有节省。

6.3 功能测试

6.3.1 服务器转发测试

首先，对流媒体服务器功能进行测试，测试视频序列为编码后的 H.265 数据。系统通过调试后，流媒体服务器根据用户的媒体名称，为用户添加媒体直播请求地址，如下图 6-5 所示：

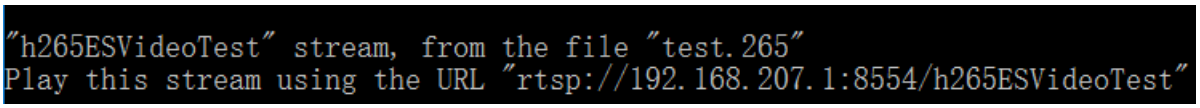

图 6-5 媒体直播请求地址

Figure 6-5 Media Live Request Address

通过 VLC 播放器通过打开 RTSP 网络串流的方式并通过用户验证，请求媒体流进行实时播放，如图 6-6 所示：

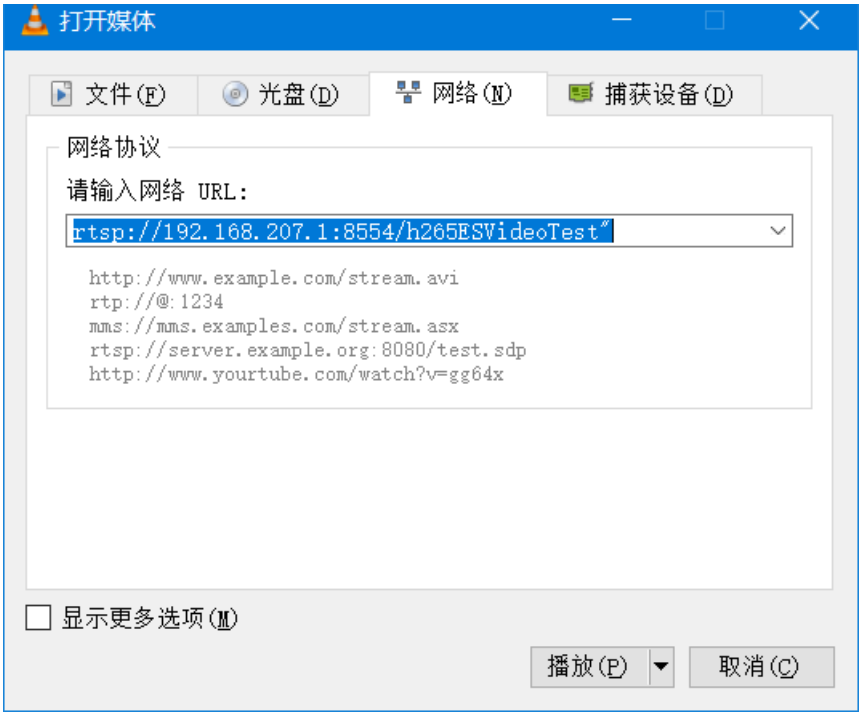

图 6-6 RTSP 网络串流

Figure 6-6 RTSP network streaming

客户端输入 RTSP 协议播放地址后，向服务器请求实时视频数据，服务器则从前端读取视频数据并进行实时播放，如下图 6-7 所示：

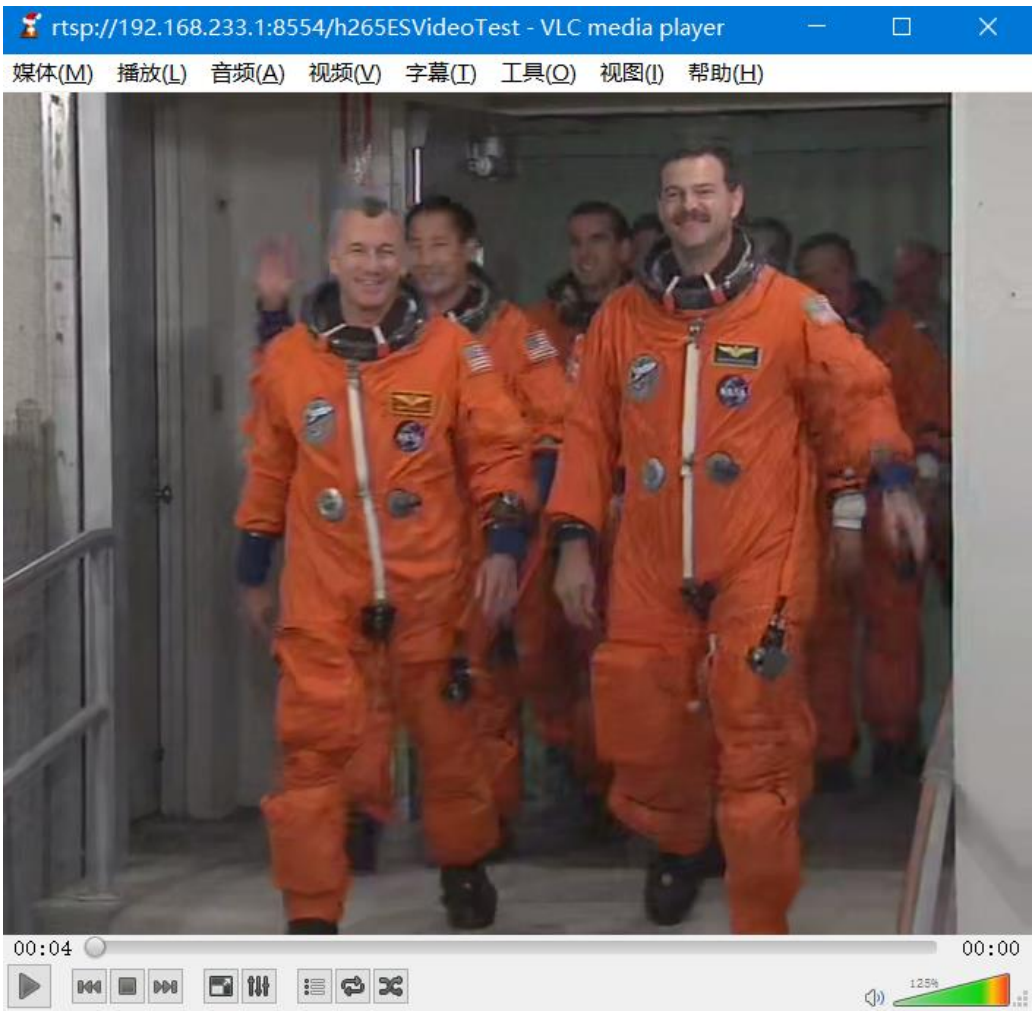

图 6-7 视频直播界面

Figure 6-7 video live interface

上图为流媒体服务器实时转发视频并通过 VLC 播放器播放的视频画面，测试结果表明，流媒体服务器能够正常接收并转发数据，能够正常工作。

6.3.2 直播测试

系统通过调试后，流媒体服务器根据用户的媒体名称，为用户添加媒体直播请求地址，并通过 VLC 播放器通过打开 RTSP 网络串流的方式，请求媒体流进行实时播放，客户端输入 RTSP 协议播放地址后，向服务器请求实时视频数据，服务器则从前端读取视频数据并进行实时播放，如下图 6-8 所示：

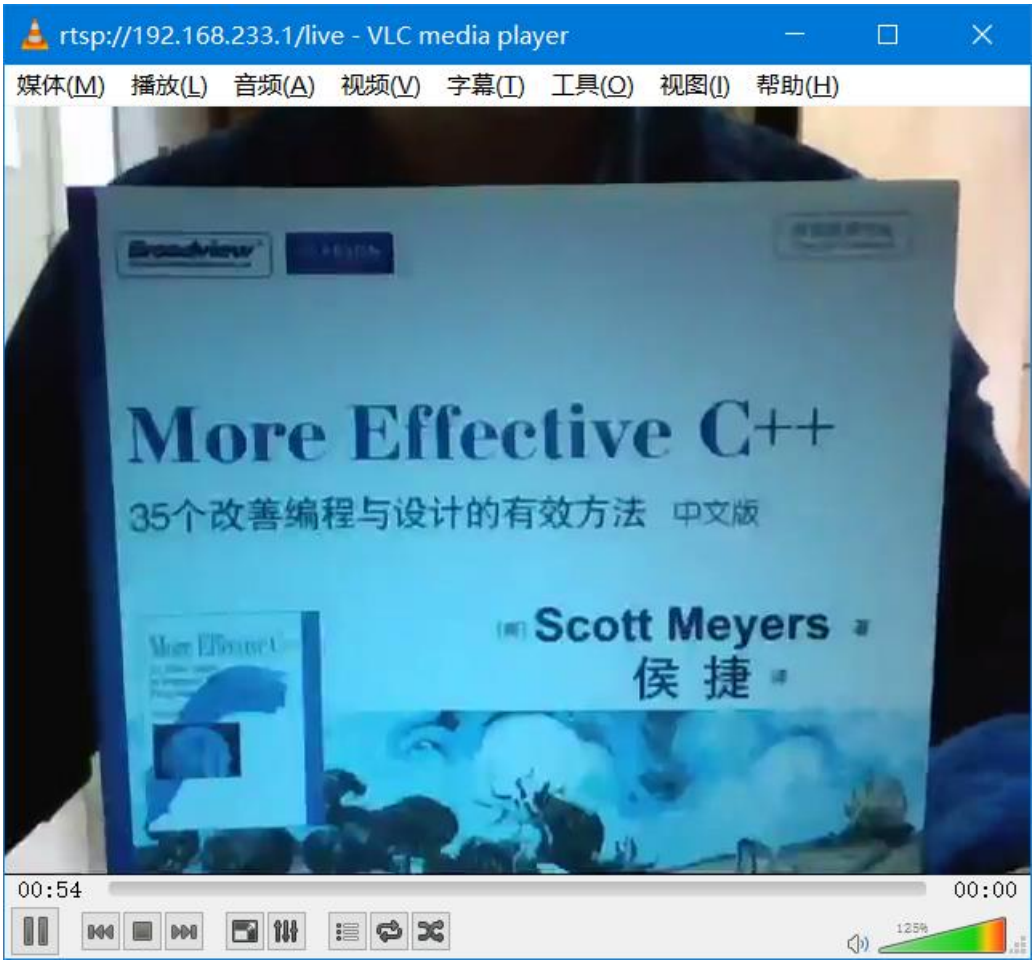

图 6-8 视频直播界面

Figure 6-8 video live interface

由上图所示，视频来源为摄像头采集的视频经过压缩编码，之后进行 TS 封装、RTP 封装经过流媒体服务器转发，最后经过 VLC 播放器播放。服务器能够完全实时转发从摄像头采集并经过压缩的数据流的功能，整个系统能够正常运行。

6.4 系统时延测试

作为视频直播系统，系统延时也是整个系统的重要特征参数。这里对服务器转发以及系统的直播延时进行估计。

6.4.1 服务器转发延时测试

本系统选用 Live555 作为系统服务器端，优点在于易于二次开发以及低延时，按照文献[48]所采用的方法对服务器转发的时延进行估计。首先，在一台终端上播放视

频同时经过服务器转发，在另一台终端上使用播放器通过实时 RTSP 地址播放这段视频。为了更加准确估计服务器转发时延，对同一段视频边播放边转发直播 10 次，依次间隔 10 秒进行一次拍照，记录当前时刻上传客户端和另一端播放器的视频播放时间，结果得出服务器转发延时在 1s 以内，能够满足要求。

6.4.2 系统直播延时

为了对系统实时转发延时进行估计，首先在视频上传端采集计时器的秒表时间，再通过播放器播放计时器的时间，两个人在两端同时记录同一时刻计时器上面显示的时间，并进行比对得出系统的延时。由于整个测试过程由两个人同时完成，多少会存在误差，因此采用多组测试来尽量减小主观判断误差。连续进行了 10 组测试，测试结果如下表 6-2 所示：

表 6-2 延时测试

Table 6-2 Delay test

测试次数	上传客户端时间	播放客户端时间	时间延时
1	01:10.20	01:07.69	2.51s
2	02:19.53	02:16.45	3.08s
3	03:22.45	03:20.04	2.41s
4	05:09.09	05:06.85	2.24s
5	06:40.18	06:38.21	1.97s
6	07:21.91	07:19.42	2.49s
7	09:05.18	09:02.66	2.52s
8	10:32.43	10:29.63	2.8s
9	12:40.56	12:38.12	2.44s
10	14:26.42	14:24.03	2.39s
平均延时			2.485s

由以上简单的测试表明，上传客户端能够很好实现采集编码，流媒体服务器实现接收、转发媒体文件的功能，整个教育系统能够正常运转，视频画面清晰，在局域网中对系统延时测试中，平均延时为 2.485s。在文献[49]中直播系统采用 H.264 视频编码标准，并对 H.264 算法进行优化，经过测试系统平均延时为 2.16s，在比较宽裕的带

宽但硬件配置不高的个人电脑的测试环境下，2-3s 的延时是可以接受的。

由于个人测试所用电脑配置不同以及受网络情况的影响，不太具有可比性，但经过对比发现本教育系统采用 H.265 编码标准，虽然在码率上有节省，但 H.265 的编码速度还是落后于比较成熟的 H.264 编码标准，系统的编码部分有待优化。由于个人能力及时间有限无法对具体的编码时间做出具体评估，而且系统没有添加 RTCP 模块对系统的传输质量进行反馈以及对系统的拥塞控制。

6.5 本章小结

本章首先介绍了视频教育系统运行的软件环境以及硬件环境，对编码后的视频做出质量评估，之后运行系统，对视频上传客户端和流媒体服务器进行功能测试以及时延估计，并对测试结果进行分析，证明系统的可行性。

总结

本文介绍了基于 H.265 的教育系统的设计与实现过程，包括视频上传客户端的设计实现过程以及利用 Live555 二次开发实现流媒体服务器。视频上传端通过无线接入网络，将摄像头采集到的视频进行编码、打包并发送到流媒体服务器，流媒体服务器时刻监听来自上传客户端的视频流和播放器端的 RTSP 信令消息以及调度和转发工作。

在实现本系统过程中主要完成以下几方面工作：

1. 通过大量阅读流媒体技术文献和网络博客，对文中使用到的技术进行了整体的概述与归纳。其中包括流媒体传输协议，视频编码原理以及常用的流媒体开源项目。熟悉这些开源的流媒体项目的使用和编程接口，并且针对 Live555 以及 FFmpeg 的源码进行了深入分析。

2. 结合系统需求分析和结构，给出系统的整体设计框图，并分别对各个组成部分进行了简单描述。

3. 设计并实现视频上传客户端，首先，采用摄像头实现视频流的采集，并利用 FFmpeg 库对采集的视频进行 H.265 编码处理，最后实现对 H.265 码流的 TS 封装。

4. 根据服务器的整体设计框图，并选用 RTSP 协议作为通信协议，实现服务器与客户端之间的会话交互，完成播放器客户端的直播请求。根据播放器端向流媒体服务器发送的 RTSP 请求，完成媒体流的转发工作，包括流媒体服务器接收来自视频上传客户端发送的数据，并进行 RTP 打包再发送给播放器端进行播放，实现流媒体的转发。

5. 最后对视频教育系统进行了功能测试测试结果表明，整个系统能够正常运行。

本视频教育系统具有一定的实用性，但仍有许多地方需要改进，具体分析如下：

1. 对于用户量的增长，数据规模变大之后，对流媒体服务器端的负载能力没有做实际的测试。本文还只是处于初级研究阶段，没有对服务器端构架以及设计进行优化与扩展，后续还将进行对大数据高并发的服务器进行研究与完善。

2. 系统在 UDP 传输方式下工作，并没有添加 RTCP 模块对 UDP 方式下的 SR 包和 RR 包处理过程。此外，后续将实现界面的添加，以便更好的改善人机交互，提高用户体验。

参考文献

[1] 潘丽佳. MOOC 设计、学习者参与度和学习绩效的关系研究[D]. 浙江大学, 2015.

[2] 王颖, 张金磊, 张宝辉. 大规模网络开放课程(MOOC)典型项目特征分析及启示[J].远程教育杂志, 2013(4):67-75.

[3] 曾明星,周清平,蔡国民等.基于 MOOC 的翻转课堂教学模式研究[J].中国电化教育, 2015(4):102-108.

[4] 阳永清,黄立新.基于校园网的便携式视频直播系统的实现[J].现代教育技术,2011,12:100-103.

[5] 周祥.华东师范大学:设计一个智能化视频直播系统[J].中国教育网络,2015,04:72-73.

[6] 王志献,于兆民,夏亚东,薛宗珑. 基于校园网的视频直播技术及其应用[J].中国现代教育装备,2008,09:11-12.

[7] 何志强,罗肖辉.基于校园网的视频直播系统[J].电脑知识与技术,2009,18:4791-4792+4798.

[8] 韩俊伟,王少锋.基于 P2P 的流媒体直播系统研究与设计[J].计算机应用研究,2006,06:227-229.

[9] He L, Ma X, Liu W, et al. A Peer-to-Peer Internet Video Broadcast System Utilizing the Locality Properties[C].IEEE International Conference on High PERFORMANCE Computing and Communications. IEEE, 2008:404-411.

[10] Hiroshi Yamamoto,Katsuyuki Yamazaki. Decentralized Live Video Broadcasting System using Yamamoto H, Yamazaki K. Decentralized live video broadcasting system using location-aware P2P network technology[C].Consumer Communications and NETWORKING Conference. IEEE, 2011:262-266.

[11] 彭宏,吴海巍,叶敏展,陈娴.基于流媒体的移动视频直播系统的设计与实现[J].电子技术应用,2014,09:111-113+117.

[12] 丁杰,潘晨光,田源.基于 crtmpserver 的手机直播系统[J].计算机工程与设

计,2014,09:3173-3178.

[13]Majumder M, Biswas D. Real-time mobile learning using mobile live video streaming[C].Information and Communication Technologies. 2012:469-474.

[14]Ullrich C, Shen R, Tong R, et al. A Mobile Live Video Learning System for Large-Scale Learning—System Design and Evaluation[J]. Learning Technologies IEEE Transactions on, 2010, 3(1):6-17.

[15]Chen T Y, Chen T H, Lin Y T, et al. H.264 Video Authentication Based on Semi-fragile Watermarking[C].International Conference on Intelligent Information Hiding and Multimedia Signal Processing. DBLP, 2008:659-662.

[16]Jamil F H B M, Lee R M A, Jerome T D, et al. Performance of H.264 in Real Time Optimization[M]. IEEE, 2009.

[17]Chu D, Jiang C H, Hao Z B, et al. The Design and Implementation of Video Surveillance System Based on H.264, SIP, RTP/RTCP and RTSP[C].Sixth International Symposium on Computational Intelligence and Design. IEEE, 2013:39-43.

[18]Bross B, Schwarz H, Marpe D. The New High-Efficiency Video Coding Standard[J]. Smpte Motion Imaging Journal, 2013, 122(4):25-35.

[19]Ohm J, Sullivan G J. High efficiency video coding: the next frontier in video compression [Standards in a Nutshell][J]. Signal Processing Magazine IEEE, 2013, 30(1):152-158.

[20]丁宗豪,曾国良.多媒体应用的图像压缩及其标准 MPEG—1(上)[J]. 电视技术, 1994(8):2-5.

[21]杨秀华. MPEG─2 标准及其编、解码器的新进展[J].电视技术, 1996(1):8-10.

[22]毕厚杰,王健.新一代视频压缩编码标准[M].人民邮电出版社, 2009.

[23]鲁业频,李凤亭,陈兆龙等.离散余弦变换编码的现状与发展研究[J].通信学报, 2004, 25(2):106-118.

[24]Sze V, Budagavi M, Sullivan G J. High Efficiency Video Coding (HEVC)[J]. Integrated Circuits & Systems, 2014.

[25]Sullivan G J, Ohm J, Han W J, et al. Overview of the High Efficiency Video

Coding (HEVC) Standard[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2012, 22(12):1649-1668.

[26]Han W J, Min J, Kim I K, et al. Improved Video Compression Efficiency Through Flexible Unit Representation and Corresponding Extension of Coding Tools[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2010, 20(12):1709-1720.

[27]Sze V, Budagavi M. A comparison of CABAC throughput for HEVC/H.265 VS. AVC/H.264[J]. 2013:165-170.

[28]赵耀,黄晗,林春雨等.新一代视频编码标准 HEVC 的关键技术[J].数据采集与处理, 2014, 29(1):1-10.

[29]安然,王浩全,张秀林等.下一代视频编码标准 H.265 的核心技术研究[J].计算机技术与发展, 2014(4):210-213.

[30]张萍,刘晓玲. H.265 关键技术研究[J].广东通信技术, 2016, 36(10):43-45.

[31]何圆圆,何凯.基于 FFmpeg 的 H.264 视频解码器的研究与实现[J].电脑知识与技术,2012,35:8519-8521+8535.

[32]蒋志峰. ffmpeg 的快速音视频开发方法[J].单片机与嵌入式系统应用,2008,01:69-71.

[33]谢志华. MPEG-2 TS 流处理及其网络传输技术的研究[D]. 江西师范大学, 2004.

[34]高建水,陈耀武,李岚岚.基于 RTSP 协议的视频点播系统设计[J].电子器件, 2006, 29(4):1143-1146.

[35]茅炎菲,黄忠东.基于 RTSP 协议网络监控系统的研究与实现[J].计算机工程与设计, 2011, 32(7):2523-2526.

[36]李校林,刘海波,张杰等.RTP/RTCP,RTSP 在无线视频监控系统的设计与实现[J].电视技术, 2011, 35(19):89-92.

[37]李燕灵,马瑞芳,左力.基于 RTP/RTCP 的实时视频数据传输模型及实现[J].微电子学与计算机, 2005, 22(8):138-140.

[38]佟雨兵,胡薇薇,杨东凯等.视频质量评价方法综述[J].计算机辅助设计与图形学学报, 2006, 18(5):735-741.

[39]李永强,沈庆国,朱江等.数字视频质量评价方法综述[J].电视技术,2006, 2006(6):74-77.

[40]韩含.视频质量 PSNR 的无参考评估方法研究[D].西安电子科技大学, 2009.

[41]Wang Z, Bovik A C, Lu L. Why is image quality assessment so difficult?[C].IEEE International Conference on Acoustics, Speech, and Signal Processing. IEEE, 2002:IV-3313-IV-3316.

[42]林翔宇.无参考视频质量评价方法研究[D].浙江大学, 2012.

[43]何应锦.基于高校平台的视频直播系统的研究与构建[D].华南理工大学, 2013.

[44]吕少君,周渊平.基于 Live555 的实时流媒体传输系统[J].计算机系统应用, 2015, 24(1):56-59.

[45]刘畅棂,包杰,王宁国.基于 Live555 的网络视频监控系统设计与实现[J].现代电信科技, 2012(12):38-42.

[46]许华滨,谢维波,黄奕.基于 Live555 的流媒体服务器设计与实现[J].微型机与应用, 2014(18):48-50.

[47]章民融,徐亚锋.基于 RTSP 的流媒体视频服务器的设计与实现[J].计算机应用与软件, 2006, 23(7):93-95.

[48]李罗涛.基于 RTSP 的 H.264 实时流媒体传输方案的研究与实现[D].华南理工大学, 2014.

[49]吴海巍.基于流媒体技术的移动视频直播系统的设计与实现[D].浙江工业大学, 2015.

独创性声明

本人郑重声明：所呈交的学位论文是我个人在导师的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，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明，并表示了谢意。本人依法享有和承担由此论文所产生的权利和责任。

论文作者签名：袁勋　　日期：2017.6.1

学位论文版权使用授权声明

本学位论文作者完全了解学校有关保存、使用学位论文的规定，同意授权广东工业大学保留并向国家有关部门或机构送交该论文的印刷本和电子版本，允许该论文被查阅和借阅。同意授权广东工业大学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、扫描或数字化等其他复制手段保存和汇编本学位论文。保密论文在解密后遵守此规定。

论文作者签名：袁勋　　日期：2017.6.1

指导教师签名：[illegible]　　日期：2017.6.2

致谢

人生有太多的不确定，原本打算两年的研究生生涯变成了广东工业大学三年的研究生生活。时间转瞬即逝，三年也即将过去，期间有过疑惑，有过彷徨，有过欢笑，也有过忧伤。不过一切即将过，人还是不知道明天要发生的事。

在老师的指导以及学长的帮助下，本研究得以顺利进行，在这里向所有在论文选题、研究、实验过程中给予帮助的老师、同学表示由衷的感谢。

首先，要感谢我的导师原玲老师，在学习上以及论文选题上给我指导，同时为我提供良好的环境来完成自己的学业。在生活上，老师就像长辈一样，为我们排忧解难。在这里向原老师表达我最诚挚的敬意和感谢！

其次，要感谢实验室同学对我在学习上和项目上的帮助，大家在生活上和睦相处，在学习上相互帮助，相互学习。

感谢我的家人一直在背后默默地支持我，是家人的关爱和信任，让我更加充满前进的动力。

最后，要感谢百忙之中抽出时间来评阅论文和参加答辩的各位专家、教授！