

硕士学位论文

电视节目直播与回看录制系统的设计与实现

学位申请人：丛延东

指导教师：曹俊 副教授

学科名称：信息与通信工程

2017年 05 月

**The design and implementation of TV live broadcast and replay recording system**

A thesis submitted to

Xi’an Jiaotong University

in partial fulfillment of the requirements

for the degree of

Master of Engineering Science

By

Yandong Cong

Supervisor: Assoc. Prof. Jun Cao

(Information and Communication Engineering)

May 2017

**论文题目：电视节目直播与回看录制系统的设计与实现**

**学科名称：信息与通信工程**

**学位申请人：丛延东**

**指导教师：曹俊 副教授**

摘 要

 近年来，随着互联网的普及与流媒体技术的进步，网络电视已经成为一种新的互联网文化。能够通过互联网观看电视直播节目、回看录制的节目及点播视频文件，成为了一个越来越被人们重视的一个需求。随着高清电视节目的普及、视频编码压缩效率的提高以及网络传输速度的提升，通过互联网观看高清电视直播节目、回看录制的高清节目的要求越来越迫切。国家数据广播研究中心基于多年的相关技术积累与产品研发经验，积极参与针对南沙的“岛礁IPTV系统”项目，本文设计与实现了其中的“电视节目直播与回看录制系统”。  
  论文首先介绍了流媒体技术的RTMP协议、HLS协议、TS格式及FFmpeg开源程序，分析、比较了RTMP协议与HLS协议的优缺点。  
 为了满足局域网内用户使用网络观看电视直播节目与相应的电视回看节目的需求，作者使用Linux系统搭建了基于HLS协议及RTMP协议的流媒体直播服务器以及回看录制系统，具体步骤为：1）在Linux系统之上编译并部署了简单可靠的流媒体服务器，使之将电视节目信号变为流媒体供网络用户观看；2）使用Python语言实现了一套回看录制系统，该系统电视直播源根据EPG信息录制成视频文件，并存储在本地HTTP服务器供客户端观看。  
 对于客户端，作者实现了多平台的客户端播放器。这些平台包括Windows/Linux PC机、iOS/Android手机、各种安卓盒子。在安卓系统平台上，使用FFmpeg对视频进行软解码并播放，以保证播放器在安卓系统上的兼容性。  
 在工程中心支撑单位的实验室内，在已经搭建好的完整的“岛礁IPTV系统”光纤网络系统中（由课题组其他成员完成的），成功挂接上了作者完成的“流媒体直播服务器以及回看录制系统”。经过一个多月的测试与改进，目前该系统运行稳定，达到了预期的目标。此外，回看节目录制系统的实现与已有的录制方案相比，大大地提高了程序运行效率。

**关 键 词**：HLS；流媒体；FFmpeg；RTMP；回看录制

**论文类型**：工程设计

**Title:** **The design and implementation of TV live broadcast and replay recording system**

**Descipline: Information and Communication Engineering**

**Applicant: Yandong Cong**

**Supervisor: Assoc. Prof. Jun Cao**

ABSTRACT

In recent years, with the popularity of the Internet and the development of streaming media technology, network television has become a new Internet culture. Watching live TV programs on the Internet, and watching the recorded programs and video files become an increasingly strong. Based on years of accumulated experience in technology and product development, National data broadcasting Research Center actively participate in Nansha Reef IPTV System. This paper designs and implements the TV program live and The replay recording system in Nansha Reef IPTV System.

Firstly, this paper introduces RTMP protocol, HLS protocol, TS format and FFmpeg open source program, of the streaming media technology, and compares the advantages and disadvantages of the two protocols.

In order to meet the LAN users’ demand of using the Internet to watch live TV programs and replay the corresponding TV show, this paper uses the Linux system to build up the Streaming media live server and replay recording system, which is based on the HLS protocol and RTMP protocol. The specific steps are as follows:1) Compile and deploy a simple and reliable streaming server over the Linux system. make it possible to change the television program signal into a streaming media for network users to watch. 2)Using Python language to achieve a set of replay recording system, which is based on EPG TV live broadcast source recorded into a video file and stored in the local HTTP server for the client to watch.

For the client, this paper realized multiplatform client player. These platforms include Windows/Linux PC machine, iOS/Android phone, a variety of Android boxes. To ensure the compatibility of the player, this paper Use FFmpeg for video decoding and playback on Android platform.

TV program live and The replay recording system built in this paper is already applied in optical network system of Nansha Reef IPTV System and the laboratory of Engineering center support unit. After more than a month of testing and improvement, the system runs stably and achieves the expected goal. Furthermore, comparing with the existing recording program, the replay recording system improves the efficiency of program operation greatly.

**KEY WORDS**: HLS；Stream media；FFmpeg；RTMP；Replay video recording

**TYPE OF THESIS**: Engineering Design

目 录

[1 绪论 1](#_Toc482014760)

[1.1 课题研究背景 1](#_Toc482014761)

[1.2 课题研究意义 1](#_Toc482014762)

[1.3 课题研究的主要内容与主要功能 1](#_Toc482014763)

[1.4 论文的组织结构 2](#_Toc482014764)

[2 相关理论 3](#_Toc482014765)

[2.1 流媒体技术原理 3](#_Toc482014766)

[2.2 TS流格式 3](#_Toc482014767)

[2.3 RTMP协议 4](#_Toc482014768)

[2.4 HLS协议 6](#_Toc482014769)

[2.5 FFmpeg 8](#_Toc482014770)

[2.6 RTMP协议与HLS协议的对比 8](#_Toc482014771)

[2.7 本章小结 9](#_Toc482014772)

[3 流媒体服务器与回看录制系统的设计与实现 10](#_Toc482014773)

[3.1 系统总体设计 10](#_Toc482014774)

[3.2 直播流媒体服务器的实现 11](#_Toc482014775)

[3.2.1 开源流媒体服务器SRS 11](#_Toc482014776)

[3.2.2 SRS的部署与使用 11](#_Toc482014777)

[3.3 EPG同步服务器的实现 12](#_Toc482014778)

[3.3.1 EPG简介 12](#_Toc482014779)

[3.3.2 EPG同步服务器系统接口设计 13](#_Toc482014780)

[3.4 回看录制系统的设计与实现 14](#_Toc482014781)

[3.4.1 系统设计 14](#_Toc482014782)

[3.4.2 回看录制管理软件的设计与实现 15](#_Toc482014783)

[3.4.3 数据库设计 19](#_Toc482014784)

[3.4.4 回看录制模块的设计 20](#_Toc482014785)

[3.4.5 回看录制模块的实现 23](#_Toc482014786)

[3.5 本章小结 26](#_Toc482014787)

[4 客户端播放器的设计与实现 27](#_Toc482014788)

[4.1 概述 27](#_Toc482014789)

[4.2 网页播放器 27](#_Toc482014790)

[4.2.1 整体结构分析 27](#_Toc482014791)

[4.2.2 功能模块实现 27](#_Toc482014792)

[4.3 安卓手机与机顶盒播放器 29](#_Toc482014793)

[4.3.1 安卓系统组件 29](#_Toc482014794)

[4.3.2 整体结构实现 30](#_Toc482014795)

[4.3.3 功能模块实现 30](#_Toc482014796)

[4.4 本章小结 34](#_Toc482014797)

[5 系统测试与分析 35](#_Toc482014798)

[5.1 系统功能测试 35](#_Toc482014799)

[5.1.1 服务器系统功能测试 35](#_Toc482014800)

[5.2 服务器压力测试 35](#_Toc482014801)

[5.2.1 srs-bench 简介 35](#_Toc482014802)

[5.2.2 压力测试结果分析 36](#_Toc482014803)

[5.3 客户端播放器功能测试 36](#_Toc482014804)

[5.4 本章小结 38](#_Toc482014805)

[6 结论与展望 39](#_Toc482014806)

[6.1 工作总结 39](#_Toc482014807)

[6.2 后期展望 39](#_Toc482014808)

[致 谢 40](#_Toc482014809)

[参考文献 41](#_Toc482014810)

声明

CONTENTS

1 Preface 1

1.1 Research Background 1

1.2 Research Objectives 1

1.3 Main contents and main functions of the research 1

1.4 Framework 2

2 Relative Theory 3

2.1 Streaming media technology 3

2.2 Format of TS stream 3

2.3 RTMP protocol 4

2.4 HLS protocol 6

2.5 FFmpeg 8

2.6 Comparison of RTMP protocol and HLS protocol 8

2.7 Summary 9

3 The design and implementation of streaming media server and recording system 10

3.1 Design of system 10

3.2 The realization of live streaming media server 11

3.2.1 Simple RTMP Server 11

3.2.2 Deployment and use of SRS 11

3.3 Implementation of EPG synchronization server 12

3.3.1 EGP 12

3.3.2 The design and implementation of the EPG synchronization server 13

3.4 The design and implementation of the recording system 14

3.4.1 Design of system 14

3.4.2 Design and implementation of record management software 15

3.4.3 Design of database 19

3.4.4 Design of replay recording module 20

3.4.5 Implementation of replay recording module 23

3.5 Summary 26

4 The design and implementation of client player 27

4.1 Summary of the design and implementation of client player 27

4.2 Web client player 27

4.2.1 Structural analysis 27

4.2.2 Implementation of functional module 27

4.3 Android client player 29

4.2.1 Android system components 29

4.2.2 Design of structure 30

4.2.2 Implementation of functional module 30

4.4 Summary 34

5 System testing and analysis 35

5.1 System functional test 35

5.1.1 Test of the server 35

5.2 Server stress test 35

5.2.1 Srs-bench 35

5.2.2 Analysis of server stress test 36

5.3 Client player test 36

5.4 Summary 38

6 Conclusions and Suggestions 39

6.1 Conclusions 39

6.2 Suggestions 39

Acknowledgements 40

References 41

Declarations

# 绪论

## 课题研究背景

随着技术的不断进步，数字电视显示器所支持的分辨率不断地提高，从最初的标清分辨率，到现在的高清分辨率如720p、1080i等。高清电视相比于标清电视，长宽比由4:3变成了16:9，显示分辨率大大增加，给予的人们更好的观看体验[1]。

H.264编码标准，是由ITU-T[视频编码](http://baike.baidu.com/item/%E8%A7%86%E9%A2%91%E7%BC%96%E7%A0%81)专家组（VCEG）和ISO/IEC[动态图像专家组](http://baike.baidu.com/item/%E5%8A%A8%E6%80%81%E5%9B%BE%E5%83%8F%E4%B8%93%E5%AE%B6%E7%BB%84)（MPEG）联合组成的联合视频组（JVT，Joint Video Team）提出的一个高度压缩数字[视频编解码器](http://baike.baidu.com/item/%E8%A7%86%E9%A2%91%E7%BC%96%E8%A7%A3%E7%A0%81%E5%99%A8)标准。与市面上现有的视频编码标准相比，H.264编码标准在相同的码率下提供了更好的图像质量。通过H.264标准，在同等图象质量下视频的压缩效率比MPEG2标准提高了2倍左右[2]。

国家数字广播中心开发了一套高清电视节目采集编码并发送的解决方案。该系统使用多路统计复用的编码方法。在图像质量相近的情况下，压缩效率相比于单路编码提高了5-10倍。

近年来，随着流媒体技术的高速发展，网络直播迅速发展成为一种新的互联网文化业态[3]。网络直播作为一种新的媒介形态，随着视频直播门槛的降低和交互方式的多元化，越来越多的人接受网络直播这种传播形式，加入直播行列的队伍逐步扩大。

## 课题研究意义

近年来，随着互联网的普及与流媒体技术的进步，网络电视已经成为一种新的互联网文化[4]。能够通过互联网观看电视直播节目、回看录制的节目及点播视频文件，成为了一个越来越被人们重视的一个需求。再加上高清电视显示器的普及、视频编码压缩效率的提高、以及网络传输速度的提升，使得通过互联网观看电视直播节目成为了可能。

本课题在国家数据广播研究中心基于多年相关技术积累与产品研发经验的基础上，积极参与针对南沙的“岛礁IPTV系统”项目，为了满足岛礁上居民在局域网内观看电视直播节目与相应的电视回看节目的需求，使用Linux系统搭建一个基于HLS协议和RTMP协议的流媒体直播服务器以及回看录制系统。现有的网络直播回看系统，局端设备价格高昂，动辄亿元投资。本课题旨在针对岛礁及大型企业3~8千户用户量特点，研究一套高可靠、低成本的视频直播及回看系统，以满足用户随时随地、用任意终端观看电视的需求。

## 课题研究的主要内容与主要功能

本课题的研究主要侧重两个方面，一是基于HLS协议与RTMP协议的流媒体直播服务器与点播服务器。二是实现可以跨平台播放流媒体视频的客户端播放器。对一第一个功能我们主要完成的工作有：研究HLS协议以及RTMP协议的内容。在Linux操作系统上搭建流媒体直播服务器与点播服务器。同时使用Python语言开发一套直播回看录制系统，根据EPG信息将电视直播节目录制成视频文件供用户点播。对于第二个功能我们需要完成的功能有：实现支持多平台的网页客户端播放器。研究FFMPEG在移动端的移植，实现Android平台的客户端播放器。

## 论文的组织结构

论文安排如下：

第一章：绪论，介绍课题的应用背景与研究现状，选题意义与研究价值，以及作者的主要工作。在这些基础上提出了论文的主要研究内容。

第二章：相关理论介绍，主要介绍了流媒体技术原理、TS流格式、RTMP和HLS流协议。分析并比较了RTMP协议与HLS协议的优缺点。为论文的系统设计打下理论基础。

第三章：设计并实现流媒体服务器与回看录制系统。包括直播流媒体服务器的实现、EPG同步服务器的设计与实现以及回看录制系统的设计与实现。

第四章：介绍客户端播放器的设计与实现。详细介绍了网页播放器与安卓客户端播放的设计与实现。

第五章：系统的测试与分析。

第六章：首先对论文中所做的工作进行了一个总结。在此基础上，对系统进一步研究进行了展望。

# 相关理论

## 流媒体技术原理

流媒体是指使用视频、音频、文本、字幕等格式进行流式传输的多媒体数据。流媒体通过实时的网络传输协议以连续的媒体流的形式在网络中传送。在目的端缓存后，按照网络传输时间的先后顺序，实时地播放连续的音视流与视频流[6]。而流媒体技术则是一种使用流媒体的综合性技术，它涉及到了多媒体视音频的采集、编码、传输、解码以及存储等多个方面。

传统的网络下载方式有着相当大的时延，音频和视频文件一般较大，需要的存储空间也相对较大。当网络带宽较小时，下载一个文件所消耗的时间很长。对于一个大文件来说，往往需要几个小时来进行下载。这种文件下载的方式不但浪费下载时间、硬盘空间，更重要的是使用起来非常不方便，用户体验不佳。使用流媒体技术之后，人们能够实现即点即看，边下边播多媒体文件，不仅极大地缩短了启动时间，而且不需要太大的缓存空间。

一个完整的流媒体系统应包括编码器、流媒体服务器以及客户端播放器这三个部分[7]。流媒体系统中各个子模块之间通过RTP/RTSP/TCP/HTTP/RTMP等实时网络传输协议进行通信数据交换。编码器的作用是将采集下来的视频源数据通过视频编码技术，编码成网络能够识别并播放的流媒体格式文件。服务器则是负责接收和转发流媒体数据以便客户端观看。客户端则是负责将编码的数据解码并播放流媒体视频数据。

## TS流格式

TS流是MPEG-2标准中定义一种用于直播的码流结构，被广泛地用于数字电视中[8]。TS流由一个个TS包所组成的。每个TS包的大小恒定为188字节。其中，一个TS包的包头为4字节。TS包基本结构和关键的标志位如图2-1所示：

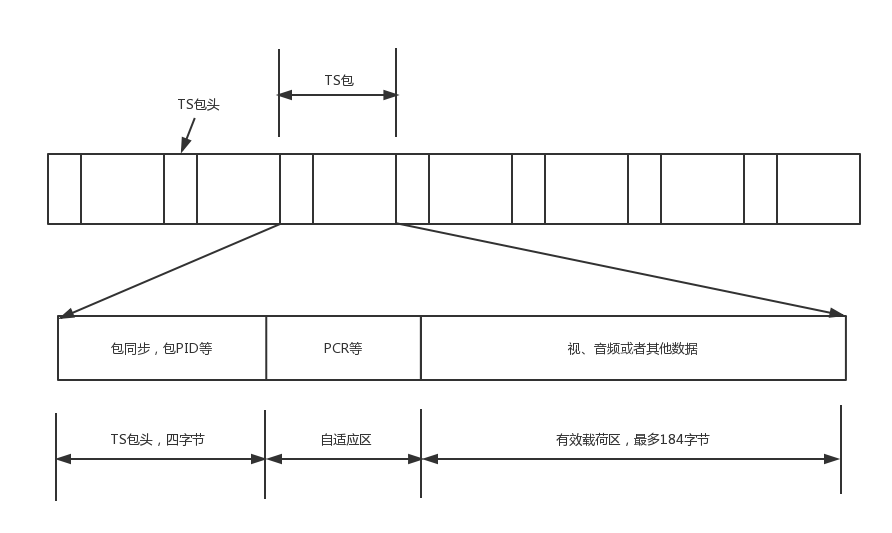


图 2‑1 TS包基本结构图

TS包语法结构如图2-2所示。可以看到一个TS包由TS包头、自适应域以及有效载荷区组成。TS包的有效负载为PES分组[9]。

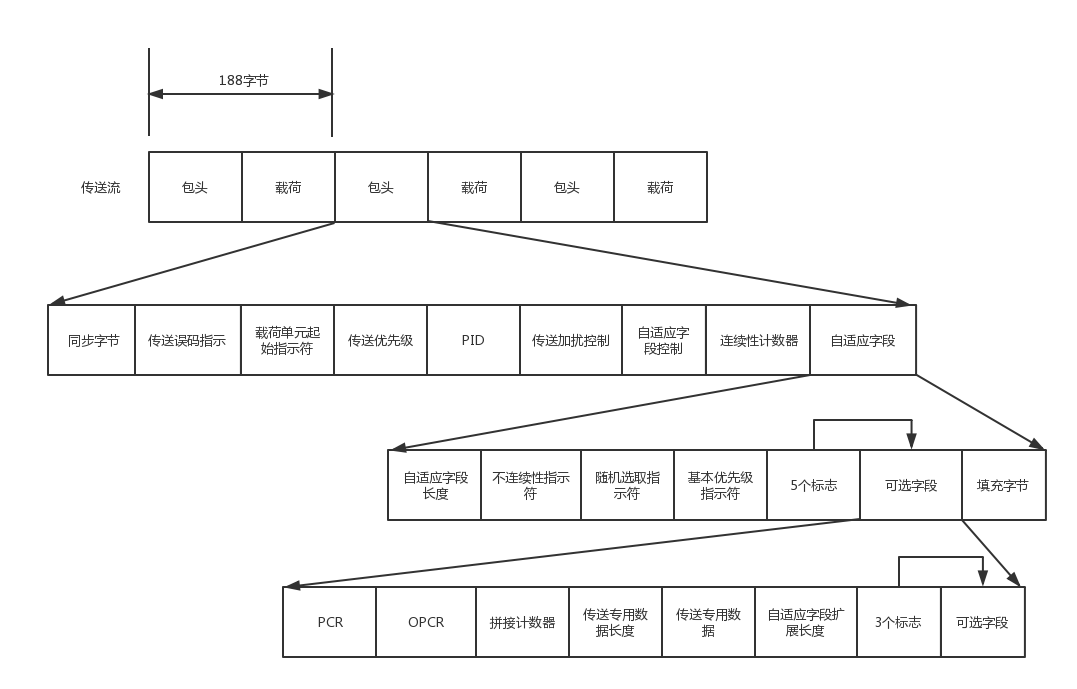


图 2‑2 TS包语法结构图

TS包头：包头中含有几个关键位的标志。包括同步字节0x47、有效载荷单元的起始符、PID以及连续计数器等标志。其中，同步字节占用1字节。有效载荷单元的起始符占用1个比特的长度，该起始符用于指示TS包中是否携带有PES数据或者PSI数据。PID是TS包的类型识别码，共占用13位。PID标志值用于标示TS包中携带的有效载荷类型，有效载荷类型分为视频、音频和辅助数据等。连续性计数器可以用于检测网络传输是否出现丢包现象。自适应字段控制指示该TS包中是否含有自适应区，共占用2位。

自适应区字段又称为调整字段，用于填充TS包，直至其达到标准的188字节长度。该字段之中有个重要标志即PCR标志，它在格式转换和TS流拼接中起到了关键的作用。当 PCR标志为l时，说明调整字段中含有PCR信息。当PCR标志为0时，表明该调整字段中不包含PCR信息。

## RTMP协议

RTMP协议是Real Time Message Protocol(实时信息传输协议)的缩写，它是由Adobe公司提出的一种应用层的实时视频传输协议。RTMP协议主要用来解决多媒体数据传输流的多路复用和分包的问题。近年来，随着VR技术的出现以及视频直播等领域的逐渐活跃，RTMP作为业内广泛使用的实时传输协议也重新被相关开发者重视起来[11]。

类似于TCP协议的三次握手，为了服务器的协议版本信息和客户端的统一，RTMP在服务器与客户端建立连接时也需要进行三次握手。RTMP的握手过程图如图2-3所示。握手的过程从客户端发送C0和C1消息块开始。客户端等待S1到达后，再继续发送C2消息会。同样，客户端必须等到S2到达之后才可以发送其他数据。服务器端必须等到C0到达之后才可以发送S0和S1数据。同样的，服务器在等到C1到达后才可以发送S2，在等到C2到达后才可以发送其他数据。C0和S0包的长度为单一的一个字节，可看做一个8比特整数域。

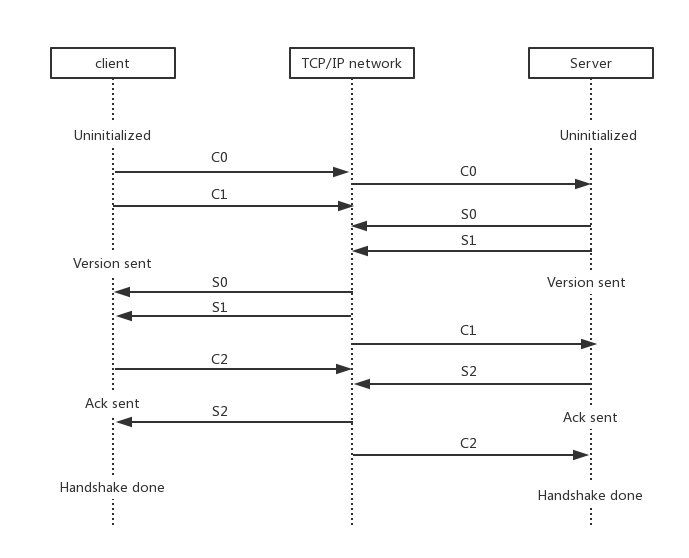


图 2‑5 RTMP握手流程图

C0消息块识别客户端需求的RTMP版本。S0消息块识别服务器端所选择的RTMP版本，当前的版本号是3。如果服务器端不能区分客户端所请求的版本，则服务器端应该返回3。此时客户端或许会选择3以下的版本，或者选择放弃握手。

C1利S1消息块的数据包各有1536个字节长。这个字段包括一个长度为4个字节的时间戳、4个字节的0字段以及最大长度为1528个字节的随机数据。时间戳字段可以作为后面的消息块从该终端发送的时间点，该字段的值可以是0或者任意值。零字段的值只能为0而且不能为其他值。随机数据字段可以包括任意值，因为每个终端必须使用自己初始化握手的值和对方初始化握手的返回值进行区分。

C2消息块和S2消息块的数据包也有1536个字节长，为C1和S1消息块的回复。这个字段由两个4字节的时间戳和随机返回数构成。第一个时间戳字段必须包含对等段发送的时间（对于C2来说是S1，对于S2来说是C1）。第二个时间戳字段必须包括先前发送的并被对端读取的包的时间戳信息。随机返同数这个字段有1528个字节，必须包含对端包含的随机数字段，每一个对等端可以利用第一个时间戳字段和第二个时间戳字段与当前的时间戳快速地估计带宽和延迟。

RTMP中的音频数据还有视频数据都是以消息（message）的形式来进行传输的。不同类型的消息通过判断消息头的数据来加以识别。为了保证视频和音频流的传递过程顺利的同时保证更大的信息传输能力，一条消息还要被切割成多个不同的块（chunk）来进行传输。这样的传输策略使得 RTMP 协议具有了更强的实时性。

RTMP协议中的各种数据是被压缩成数据包的隔阂，并打上同步的时间戳进行传输的。不同类型的消息使用不同的数据包进行传输，多个消息又可以在同一个连接上交错传输。每一个RTMP消息块都由包头和数据组成。

RTMP协议在建立连接时，首先进行三次握手操作，为服务器和客户端统一协议版本信息。然后建立连接，创建传输流，设置分块大小，发送数据。如图2-6所示，Basic header是消息快的基本头，这个字段只有一个字节，编码了消息块流的ID和消息块的类型，用于传输控制消息，比如设置分块大小，丢弃消息以及用户控制消息等。消息块的类型决定了对应消息包头的编码格式。消息块流的ID决定了消息块的长度，该字段的长度有有3个字节，可以识别单一连接中不同的流。Chunk MSG Header指消息块的消息头。该字段包含了正在传送消息的信息。Chunk MSG Header的长度由消息块中详细的消息块类型决定。Extend Time Stamp是指扩展的时间戳信息。消息分块可以将数据量大的消息分割成小的消息块，这样就不会阻塞数据量小的Message传输通道。

|  |  |  |  |
| --- | --- | --- | --- |
| Basic Header | Chunk MSG Header | Extended Time Stamp | Chunk Data |

图 2‑6 RTMP消息包结构图

RTMP协议包含多个通道，其作用是发送和接收数据包。每个通道有其特定的功能。 如处理视频流数据、处理请求和响应、处理视音频流数据以及处理控制消息等等。这些通道之间是互不干扰、相互独立的。所以，在RTMP协议的传输过程中，通道可以在任何时间同时运行。

## HLS协议

HTTP Live Streaming（HLS）是由苹果公司开发并提出的一个基于HTTP协议的流媒体网络传输协议[14]。HLS协议的工作原理是把视频流切分成一个个小的基于HTTP的TS切片文件来下载，每次只下载一部分。当客户端选择播放媒体流时，从索引文件中选择文件进行下载，同时还可以当前网络传输速率选择不同的源，即允许流媒体会话适应不同的数据速率。在开始一个流媒体会话时，客户端会下载一个包含元数据的extended M3U (m3u8)playlist文件，用于寻找可用的媒体流。

HLS协议架构图如2-7所示：

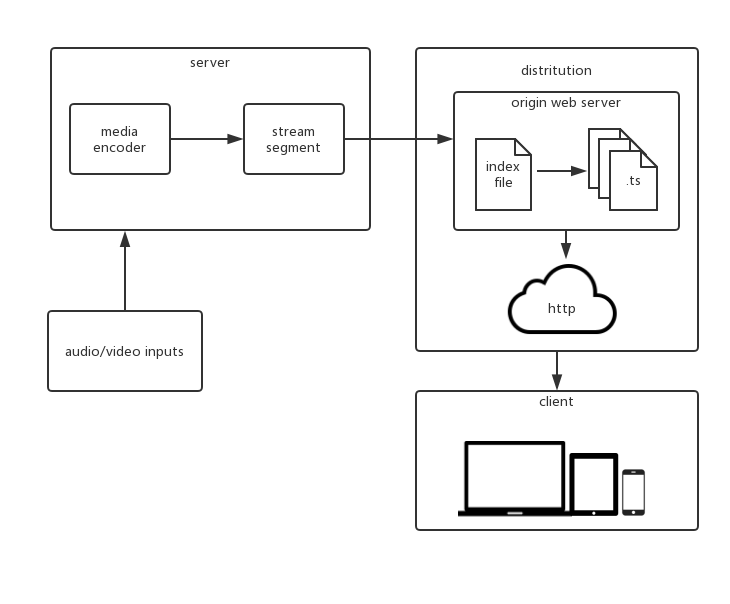


图 2‑7 HLS协议架构图

首先，视频源audio/video inputs可以是任意格式的视频源，他与server之间的通信协议也是任意的（如RTMP，HTTP），只要将视频数据以任意一种方式传输到服务器上即可。这个视频源在server服务器上被转换成HLS格式的视频（既TS和m3u8文件）文件。细拆分来看，server里面的Media encoder是一个负责将视频源中的视频数据转码到目标编码格式（H.264）的转码模块，视频源的编码格式可以是任何的视频编码格式。将视频源转码成H.264视频数据之后，在stream segmenter模块将视频切片，切片的结果就是index file（m3u8）和TS文件了。图中的Distribution其实只是一个普通的HTTP文件服务器，里面存储着切片后的TS文件和m3u8文件。客户端只需要访问一级index文件的路径就会自动播放HLS视频流[16]。

## FFmpeg

FFmpeg 是一套开源的音视频处理软件。它可以用来进行格式转换以及视音频编解码等操作[19]。FFmpeg有着跨平台的特性，其代码可以在市面上所有流行的操作系统中进行编译。FFmpeg在源码中包含了格式众多的编解码器，可以涵盖众多的音视频格式，如FLV、MPEG2、H.263、H.264等格式。使用FFmpeg可以轻松地实现多种视频格式之间的转换。例如，可以将AVI格式的视频转成现在视频网站所采用的FLV格式。

FFmpeg拥有的众多强大的视频处理功能，包括视频采集功能、视频格式转换、视频抓图、给视频加水印等。FFmpeg源码文件中包含以下模块：

libavformat：该模块用于生成和解析各种封装格式的视音频。主要包含获取解码所需信息和生成解码上下文结构以及读取音视频帧等功能。该模块为 libavcodec模块分析码流提供独立的音频和视频的码流源数据。  
 libavcodec：该模块负责视音频的编码以及解码任务。libavcodec模块是FFmpge音视频编解码的核心。市面上流行的解码播放软件如ffdshow以及Mplayer等都使用了该库提供的解码功能。  
 libavdevice：该模块负责硬件的采集、加速以及显示。  
 ffmpeg：该软件提供的一个基于命令行的工具。可进行视频格式转换、视频编解码等操作。  
 ffsever：该软件体统一个基于HTTP协议的流媒体服务器。  
 ffplay：是一个简单的视频播放器。ffplay使用FFmpeg库进行解码，通过SDL进行显示播放。

本工程主要将FFMpeg移植到Android系统之中，对H.264视频格式的视频进行解码与播放。

## RTMP协议与HLS协议的对比

HLS协议在兼容性与扩展性方面相比RTMP协议有着天然的优势。首先，HLS的客户端支持简单，只需要支持 HTTP 请求即可。HLS只请求基本的HTTP报文。它也很容易使用内容分发网络来传输媒体流。HTTP 协议无状态, 只需要按顺序下载媒体片段即可。相比之下RTMP协议不使用标准的HTTP接口传输数据，所以在一些特殊的网络环境下可能被防火墙屏蔽掉。但是HLS由于使用的HTTP协议传输数据，不会遇到被防火墙屏蔽的情况。由于使用HTTP协议，所以HLS的CDN 支持良好。相比来说，由于RTMP是一种有状态协议，很难对视频服务器进行平滑扩展，因为需要为每一个播放视频流的客户端维护状态。而HLS基于无状态协议（HTTP），客户端只是按照顺序使用下载存储在服务器的普通TS文件，做负责均衡如同普通的HTTP文件服务器的负载均衡一样简单。另外HLS协议本身实现了码率自适应，不同带宽的设备可以自动切换到最适合自己码率的视频播放。

HLS协议另外一个优势是他的跨平台兼容性。苹果公司在自家的平台上只提供了对HLS的原生支持，并且放弃了FLASH。安卓系统也原生支持了HLS。而在客户端上播放RTMP格式的视频，需要额外安装FLASH播放器。庞大的FLASH也会给系统带来更重的负担。

由于HLS协议是以ts切片文件的形式进行传输，所以HLS会有一个较高的延迟。延迟的最大时间为单个ts视频文件的视频长度（直播一般为10秒）。而RTMP由于是专有协议，可以做到很小的延迟时间。

综上所述，HLS协议由于有着更好的平台兼容性以及扩展性，所以广泛地用于移动端的视频直播与点播。而RTMP协议由于其延时小，所以一般用于PC端的视频直播之中。

## 本章小结

本章介绍了流媒体相关技术，包括视频传输协议TS流协议、RTMP协议、以及HLS协议。分析并比较了RTMP协议与HLS协议的优缺点。HLS协议有着更好的平台兼容性以及扩展性，被广泛地用于移动端的视频直播与点播。而RTMP协议由于其延时小，被用于PC端播放直播视频。本章的研究为后文直播点播流媒体服务器的设计以及传输协议的选择提供了有力的技术支撑与理论依据。

# 流媒体服务器与回看录制系统的设计与实现

## 系统总体设计

流媒体服务器与回看录制系统的整体系统框图如图3-1所示：

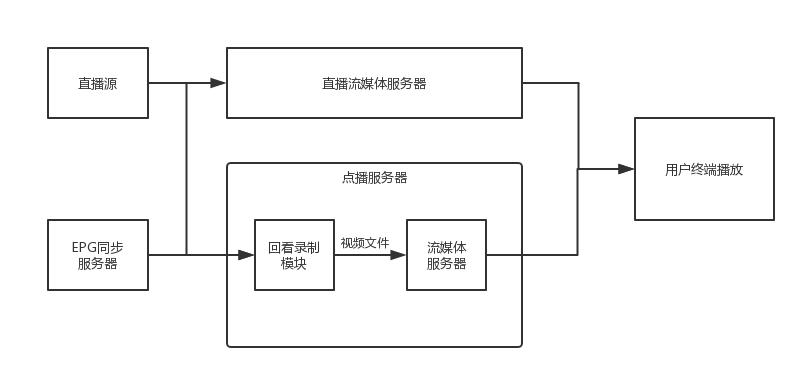


图 3‑1 整体系统框图

图中的直播源为指定格式的视频源。本课题使用国家数字电视广播中心开发的卫星直转系统接收来自卫星的电视节目及自办节目，通过转码设备重新编码为各个码率的H.264直播流，以RTMP协议的传输格式将视频流推流到直播流媒体服务器上。同时，直播源还以UDP协议的传输格式将视频流推流到点播服务器上。点播服务器中的回看录制模块根据EPG同步服务器提供的EPG信息，将视频流切分成视频文件存储在本地流媒体服务器中。直播流媒体服务器和点播流媒体服务器均支持多用户的流媒体直播服务。最后，终端用户可以使用本课题开发的客户端软件，即可观看电视直播节目以及最近七天的电视回看节目。

由图可见，该流媒体服务器与回看录制系统应该包括以下几个模块：

流媒体服务器模块：服务器的主要功能是以流式协议将视频文件传输到客户端，以供多用户在线观看。直播流媒体服务器需支持接收RTMP推流，输出RTMP与HLS协议的直播流。

EPG同步服务器模块：向回看录制模块提供相应的电视频道信息（EPG）。每条EPG信息包括电视频道的名称，节目的起始时间与结束时间，节目的名字等。

回看录制模块：回看录制模块为整个回看录制系统的核心。该模块根据EPG同步服务器模块提供的EPG信息，将直播源传入的直播流切分成对应的视频文件存入本地的流媒体服务器中，同时，将对应的视频信息存入设计好的数据库中以供客户端播放的时候读取。

流媒体服务器采用PC作为硬件支持，搭建在Ubuntu操作系统上，EPG同步服务器使用Java语言实现，回看录制模块使用Python语言实现。

## 直播流媒体服务器的实现

流媒体服务器的主要功能就是接受任意格式的视频流，当终端用户请求视频时，将视频流以指定格式传送给终端用户。在本课题中，直播流媒体服务器的输入源传输格式为RTMP，输出传输格式为RTMP和HLS，视频源的编码格式为H.264。

出于对系统稳定性以及易维护性的考虑，本课题选用开源流媒体服务器SRS（Simple RTMP Server）作为直播流媒体服务器。

### 开源流媒体服务器SRS

SRS定位是运营级的互联网直播服务器集群，追求更好的概念完整性和最简单实现的代码。SRS提供了丰富的接入方案将RTMP流接入SRS，包括推送RTMP到SRS、拉取流到SRS。SRS还支持将接入的RTMP流进行各种变换，譬如直播流转码、转发给其他服务器、转封装成HLS、录制成FLV。SRS包含支大规模集群如CDN业务的关键特性，譬如RTMP多级集群、VHOST虚拟服务器、无中断服务Reload。此外，SRS还提供丰富的应用接口，包括HTTP回调、HTTP API接口、RTMP测速。SRS在源站和CDN集群中都得到了广泛的应用。

与支持rtmp的http服务器nginx-rtmp的相比，SRS(Simple Rtmp Server)单进程能支持9000并发，nginx-rtmp单进程最多支持3000个，单进程的性能SRS(Simple Rtmp Server)是nginx-rtmp的三倍。

### SRS的部署与使用

SRS主要运行在Linux系统上，譬如Centos和Ubuntu，包括x86、x86-64、ARM和MIPS。MacOS支持代码编辑和编译。

SRS的系统架构如图3-2所示：

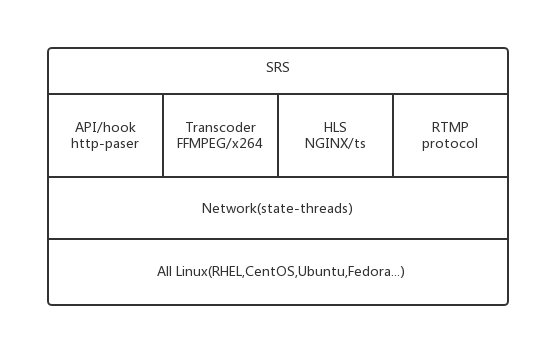


图 3‑2 SRS系统架构图

SRS服务器架构在所有Linux操作系统之上，网络服务器架构使用state-threads架构，state-thread是一个C的网络程序开发库，提供了编写高性能、高并发、高可读性的网络程序的开发库，支持UNIX-like平台。它结合了多线程编写并行成的简单性，一个进程支持多个并发，支持基于事件的状态机架构的高性能和高并发能力。在此之上，SRS提供了http API回调接口、视频转码以及HLS协议与RTMP协议的支持。

在下载源码、确定用什么编译选项后，编译SRS其实很简单。只需要RTMP和HLS：

./configure && make

指定配置文件，即可启动SRS：

./objs/srs -c conf/srs.conf

SRS启动之后，只需要将直播流以RTMP的形式推送到服务器上，便可以实现视频的直播功能。

## EPG同步服务器的实现

### EPG简介

EPG 是Electronic Program Guide的英文缩写，全称为电子节目单。EPG一般为IPTV提供各种电视节目信息。EPG以“频道-时间”的方式提供一段时间内所有电视节目的信息。相关的应用根据EPG提供的数据实现其业务逻辑。EPG可选的功能相当的丰富。其中，可以给电视节目信息增加附加信息，如情节介绍、节目类别等。VOD节目也可以按照节目类别进行分类。比如电影类VOD节目可以分类为喜剧片、动作片、爱情片、恐怖片、卡通片以及卡通片等。在EPG节目单的基础上，还可实现节目预约功能、业务搜索功能以及业务导航功能。节目预订功能指预约一段时间之后将要播放电视的节目。时间到了便自动播放。业务搜索功能即提供多种方式的基于EPG的业务搜索功能。业务导航功能即根据EPG信息提供业务更新、业务排行、最热视频等导航功能。

在“岛礁IPTV系统”之中，只需要EPG提供最基本的时间节目信息即可。其中包括：电视直播频道的信息，以及每个频道的节目信息。其中，每个频道的节目信息应包括节目的名称、起始时间以及结束时间。直播服务器根据所提供的的EPG频道信息可以提供相应的直播服务。回看服务器可以根据所提供的EPG节目信息录制相应的回看节目并提供回看服务。

### EPG同步服务器系统接口设计

EPG同步服务器主要实时同步100套电视直播节目的EPG信息，并对外提供相应的数据接口。

首先，服务器每隔一段时间，便从第三方数据源处获取实时的EPG更新数据。EPG同步服务器提供的数据接口如下：

授权频道接口：

http://tv.com/EPG/channel?secret=XXXX

节目单接口（两周以内全量节目单）：

http://tv.com/EPG/schedule?secret=XXXX&id=XXXX

其中secret参数为唯一认证秘钥，客户调用所有接口都需要传入此参数来标识客户的身份。id参数为需要获得频道信息的频道id。客户首先需调用接口得到所有EPG授权给该客户的频道，然后将获得的频道id传入接口。接着得到该频道两周内（不足两周的有多少给多少）的节目单：节目单中若related\_teams节点不为空（1-2个id），则取其id传入接口。

授权频道的XML结构图如图3-3所示：

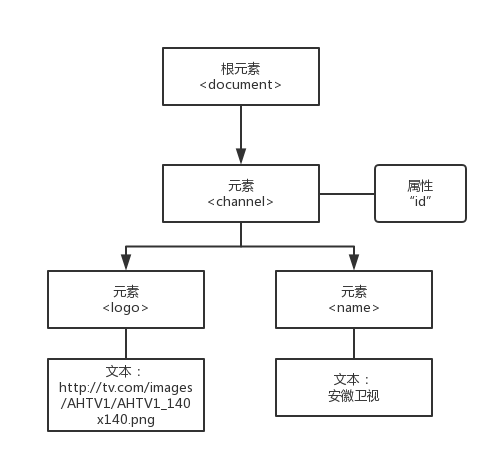


图 3‑3 授权频道XML结构图

根节点document下包含若干channel元素，每个channel元素使用id属性唯一标识。channel元素下的logo标签存储频道台标图片URL，channel元素下的name标签存储频道名称。

节目单信息的XML结构图如图3-4所示：

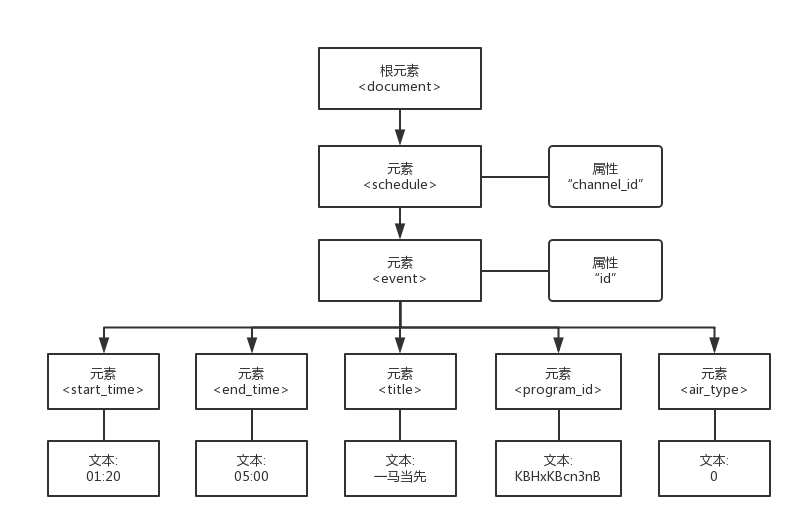


图 3-4 节目单信息XML结构图

根节点document下包含一个schedule元素，属性channel\_id表示该schedule元素的频道id。schedule元素下包含若干event元素，每个event元素使用属性id唯一标示，标示一个电视节目的信息，数据库中节目的增删均根据该id进行。event元素下的start\_time标签和end\_time标签分别存储该节目的起始时间与结束时间，title标签存储该节目的节目名称，program\_id为标示该节目的唯一id，air\_type为该节目的播放类型（仅支持综艺栏目），其中序号0代表位置，序号1代表直播，序号2代表首播，序号3代表重播。

## 回看录制系统的设计与实现

### 系统设计

回看录制系统根据EPG同步服务器模块提供的EPG信息，将直播源传入的直播流切分成对应的视频文件存入本地的流媒体服务器中，同时，将对应的视频信息存入设计好的数据库中以供客户端播放的时候读取。除此之外，还需设计一个管理系统，控制节目录制的开始于停止，以及录制节目频道的添加与删除。系统流程图如图3-5所示：

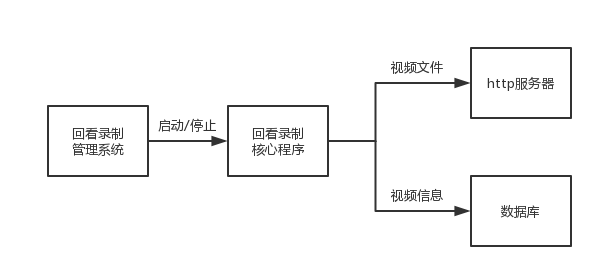


图 3-5 回看录制系统流程图

由图可知，回看录制系统发送命令给回看录制核心程序，回看录制程序按照EPG信息将直播源切分成视频文件存入http服务器中。同时将视频信息存入数据库中以供客户端播放的时候读取对应的视频文件。http服务器充当支持HLS协议的流媒体服务器。

### 回看录制管理软件的设计与实现

对系统进行基本的需求分析之后，可知该管理软件至少应该具有如下功能：（1）系统管理员可以添加直播频道的信息。（2）系统管理员可以删除直播频道的信息。（3）系统管理员可以编辑直播频道的信息。（4） 系统管理员可以启动回看节目的录制。（5）系统管理员可以停止回看节目的录制。（6）系统管理员可以监控回看节目的录制情况。

通过对系统需求的分析，可以确定该系统的参与者即为系统管理员。可以确定系统中有如下用例存在：（1）Add Channel Info（添加直播频道信息）：本用例提供了添加直播频道信息的功能。（2）Delete Channel Info（删除直播频道信息）：本用例提供了删除直播频道信息的功能。（3）Edit Channel Info （编辑直播频道信息）：本用例提供了编辑直播频道信息的功能。（4）Start Replay（启动回看节目录制）：本用例提供了启动回看节目录制的功能。（5）Stop Replay（停止回看节目录制）：本用例提供了停止回看节目录制的功能。

系统用例图如图3-6所示：



图 3-6 回看录制管理软件系统用例图

定义完系统需求，就可以根据系统需求来识别系统中存在的对象了。系统对象的识别可以通过寻找系统域描述和需求描述中的名词来进行。从这些对象中筛选，如果有与该对象相关的身份和行为，那么就应该为之创建类。（1）类Channel：类Channel代表直播频道的信息。直播频道的信息包括频道的唯一id、名称、直播流的地址。类Channel应该具有下列私有属性：channel\_id代表直播频道的唯一id。channel\_name代表直播频道的名称。url代表直播流的地址。active代表该对象的激活状态。start代表回看节目的录制状态。Channel的公共操作包括如下：query方法查询是否存在对应的直播频道信息。update方法更新直播频道信息的内容，如改变频道名称，直播流地址以及激活状态和录制状态。（2）类Program ：类Program代表回看节目的信息。 类Program的公共操作包括如下：add方法增加一条回看节目信息。delete方法删除指定的回看节目。

系统管理员与系统需要交互，一个用户友好的系统通常都采用直观的图形化界面，因此需要定义系统的用户界面类。

（1）类TSTable：TSTable是系统的主界面，为一个显示直播频道信息的列表。每一行显示的是直播频道信息，其中包括直播频道的id、名称、直播流的地址，回看录制机ip等信息。每条信息都提供编辑删除按钮。点击按钮上便可以编辑或删除该条信息。同时，每条直播频道信息都有一个录制按钮，当按下录制按钮时，启动回看录制程序。再次按下停止。界面如图3-7所示：



图 3‑7 回看录制管理软件系统主界面原型图

公共方法如下：newTSTable方法创建系统主界面。edit方法编辑对应的直播频道信息，当按下“编辑”按钮时，该方法被调用。delete方法删除对应的直播频道信息，当按下“删除”按钮时，该方法被调用。addChannel方法添加一条新的直播频道信息，当按下“添加”按钮时，该方法被调用。start方法启动录制，当按下“录制”按钮时，该方法被调用。stop方法停止录制，当处于录制状态按下“录制”按钮时，该方法被调用。

（2）类AddDialog：界面类AddDialog是用来添加一条新的直播频道信息。当按下TSTable中的“添加”按钮时，对话框AddDialog弹出，管理员填写相关的直播频道信息，点击“提交”按钮，便可将该条信息添加到系统数据库中并显示出来。界面如图3-8所示：



图 3-8 添加对话框界面原型图本

类AddDialog具有如下方法：newDialog方法负责创建添加窗口。add方法用于添加录制频道信息。当添加窗口被提交时，该方法被调用。

（3）类EditDialog：界面类EditDialog的功能与AddDialog较为相似，用来重新编辑一条直播频道信息。当按下TSTable中的“编辑”按钮时，弹出EditDialog对话框。管理员修改相关的信息，点击“提交”按钮，便可以修改该条直播频道信息。

类EditDialog具有如下方法newDialog方法用于创建编辑窗口。edit方法用于编辑录制频道信息，当编辑窗口被提交时，该方法被调用。

添加直播频道信息的时序图如图3-9所示：



图 3-9 添加直播频道信息时序图

当要添加直播频道信息时。管理员发送add消息给类TSTable，类TSTable又发送newDialog消息给类AddDialog，即类AddDialog的方法被调用，创建用于添加直播频道信息的对话框。管理员填写必要的信息之后提交信息，类AddDialog的方法add被调用，发送消息给Channel类，首先调用Channel类中的query方法，确认数据库中是否存在该条信息，如若存在（若不存在，则显示提示信息），则调用add方法，更新并激活该条信息。

编辑与删除直播频道信息的流程与添加直播频道信息的流程类似。不同的是TSTable所调用的方法分别为edit与delete方法。

启动与停止录制回看视频的时序图如图3-10所示：



图 3-10 启动与停止录制回看视频的时序图

当要启动或停止录制回看视频时，管理员发送send消息给类TSTable，类TSTable通过当前显示状态判断是进行启动还是停止操作，接着，类TSTable发送start或者stop命令给管理服务器，最后管理服务器发送start或者stop命令给远程的回看录制服务器。最后回看录制服务器返回程序执行的结果，并显示在用户界面上。

除了管理回看节目的管理与启动之外，本软件还要负责提供一套供客户端播放器进行播放的数据接口。接口定义如下：

function get\_channels()

function get\_videos(string channel\_id,int day)

其中，函数get\_channels返回当前所有可用的频道信息。函数get\_videos则返回指定天数的回看视频信息。客户端获取数据时使用该数据接口调用顺序如下：首先，调用get\_channels函数，该函数将返回所有数据库中active项等于1的直播信息，接着，客户端遍历所有的直播频道，在每个直播频道中使用get\_video函数获取最近7天的回看节目信息，返回的数据中，如果字段finished为1，则显示在界面上待用户点击观看。

### 数据库设计

本系统共有两个实体类，即类Channel与类Program。类Channel与类Program为一对多的关联关系。

根据已有需求，建立数据库模型。数据库的逻辑模型如图3-11所示。本系统使用关系型数据库存储和管理数据。关系表的UML符号用衍型为<<retional table>>的类符号表示，带有衍型<<pk>>的属性代表主键，带有<<fk>>的属性代表外键。



图 3-11 数据库逻辑模型图

本系统数据库共有两个表，channel中表包含了所有频道的数据。其中“channel\_id”为描述频道的唯一id，“channel\_name”为频道名字，“rtmp\_url”为直播流的url地址，“active”为激活标志，“start”为是否启动录制回看节目的标志。program存储所有回看节目的数据，其中“channel\_id”为描述频道的唯一id，“start\_time”与“end\_time”为回看节目的起始时间与结束时间，“url”为录制节目的url地址，“finished”为录制成功的标志位，”title”为回看节目的名称。

channel表与program表为一对多的关系。channel表与program表之间的一对多关系通过在表program中插入外键“channel\_id”以匹配channel表中的“channel\_id”来模拟。

### 回看录制模块的设计

在图3-7中，我们看到，系统管理员在回看管理软件中，点击“启动”按钮时，管理服务器会发送一条命令给回看服务器，此时，回看服务器便把指定的直播源根据EPG信息录制成回看视频文件并存放在回看服务器中。由此我们知道，回看服务器的主要功能便是从直播流中录制可供用户观看的电视节目。首先，回看服务器先从EPG同步服务器中将对应的EPG信息存储在本地数据库中，接着，回看服务器从指定的流中录制视频，并根据实时的EPG信息生成回看视频文件，最后将生成的回看视频信息写入数据库中以便用户进行观看。

从图中我们可以看到，实现回看服务器的关键便是从指定流中录制视频这一过程，既要考虑程序运行的稳定性、又要考虑生成视频格式的兼容性等等问题。在开发过程中，一共实现过以下三个方案:

第一种方案：将rtmp直播流录制成flv文件存储在本地http服务器。流程图如图3-12所示：

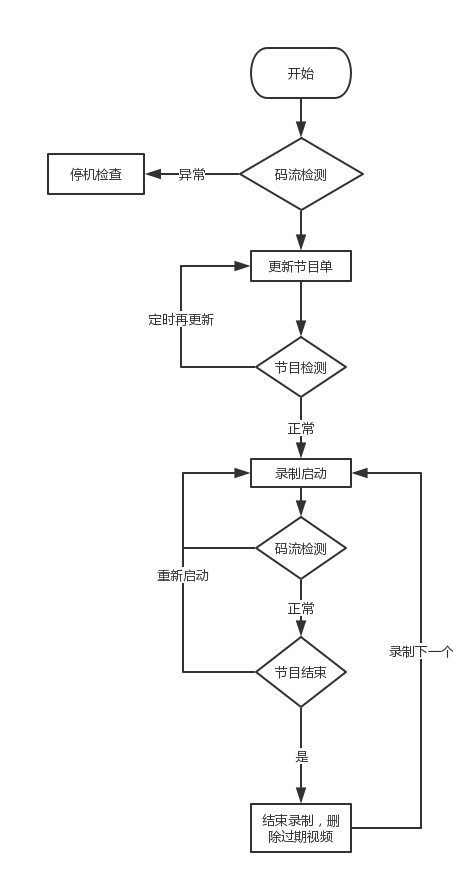


图 3-12 回看录制程序流程图1

初次启动录制时，程序以channel\_id作为运行参数，以达到录制相应频道节目的目的。创建文件目录和程序运行日志，目录存在则跳过次步骤，如果目录不存在则创建相应目录。根据channel\_id从channel表中获取RTMP码流地址并测试码流是否获取正常。如果测试失败则将数据库中运行状态和主进程号写0并终止程序，如果测试通过则将主进程号和运行状态1写入数据库并将进行下一步操作，从节目单接口获取跟channel\_id相对应的XML格式节目数据，然后将其中有用的部分进行解析和处理后保存在数据库中。然后根据channel\_id和当前系统时间，从刚刚保存在数据库的program表中找到符合当前时间的节目并准备开始录制。如果当前时间刚好处在两段节目中间的空档期时，则进入等待状态直到节目播出时间后再准备开始录制。

到达节目录制时间后，程序为当前节目文件创建临时文件名dump.flv和随机文件名变量，这是因为直接抓取的直播流数据文件没有节目的时长信息，在页面播放器中播放节目时会造成无法拖动的问题，所以每个节目录制时先以临时文件名dump.flv保存，待录制结束后注入元数据并生成最终的节目文件，然后删除临时文件。

然后再次以当前channel\_id及日期创建文件夹和子文件夹，目的是解决节目循环录制过程中遇到日期变化的问题，如果文件夹已存在则跳过该步骤，创建当前节目文件的保存路径变量，创建码流录制命令变量，创建元信息注入命令变量，创建节目录制子进程变量，创建进程组号变量，完成上述操作后，开始录制节目文件。

录制节目进行中，每隔60秒进行节目剩余时间判断及码流判断，如果判断出现码流不正常，则跳出该录制子进程并重新开始录制，直到码流恢复正常为止。如果码流正常则继续录制，节目结束前60秒停止判断。结束后进行元信息注入并保存节目文件，删除临时录制文件，删除724小时前的节目单信息和节目文件，将录制好的节目地址及完成状态写入数据库中，结束录制子进程。完成上述步骤后重新回到读取节目单信息的步骤，从节目单接口更新节目单数据，然后读取一条合适的节目信息并开始录制循环。如果节目单出现更新问题，无法获取新的节目单，则每隔30秒重试1次，共5次。5次获取节目单信息失败后程序等待30分钟再次重试5次，直到节目单接口能够正常获取数据为止。为了优化程序结构，便于管理维护，将录制部分的程序分为三个模块，分别为录制脚本模块rtmpdump、数据库管理模块db\_manager以及关闭脚本模块kill。

第二种方案：将hls直播流中的ts文件重组，以m3u8索引文件的格式存储在本地http服务器中。由于hls直播流由一个个长度为10秒的ts文件组成。对ts文件进行简单拼接，存储在本地。再生成对应的m3u8文件，即可完成对直播视频的录制任务，流程图如图3-13所示：

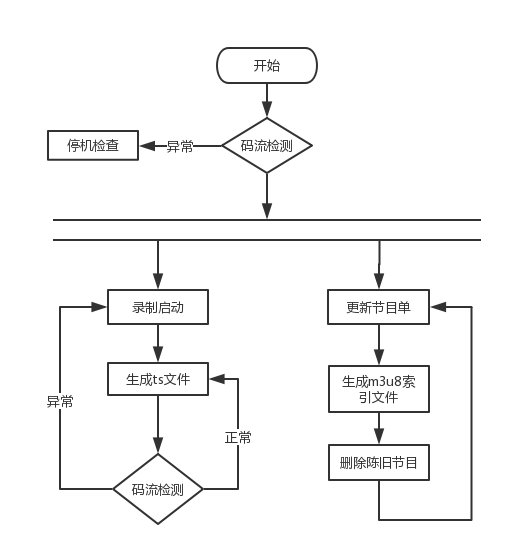


图 3-13 回看录制程序流程图2

初次启动录制时以程序输入的channel\_id作为参数，以达到录制相应频道节目的目的。创建文件目录和程序运行日志，目录存在则跳过，日志如果存在则从最后一行开始记录。根据channel\_id从channel表中获取RTMP码流地址，将RTMP码流地址映射为对应的hls码流地址并测试码流是否获取正常，如果测试失败则修改数据库中该频道的录制状态并终止程序。如果测试通过则进行下一步操作。程序创建一个新的进程，将原来长度为10秒的ts文件拼接，生成全新的长度为60秒的ts文件并存储在本地。与此同时，程序主进程每隔三十分钟对服务器中所有进行录制的频道进行如下操作：（1）从EPG同步服务器中更新最新的EPG信息到服务器中。（2）根据数据库中的EPG信息与本地存储的ts文件，生成对应的m3u8文件以供用户观看。（3）删除本地文件系统中陈旧的视频节目。

第三种方案：将UDP流文件录制成ts切片，以m3u8索引文件的形式存储在本地http服务器中。与方案二不同的是，视频录制的源的网络传输协议从HLS变成了UDP。

在经过大量的测试得知，对于节目时长过长的节目来说，方法一中使用RTMP录制下来的FLV文件往往过大，这使得在网络传输中造成了不必要的带宽浪费。FLV文件无法在苹果公司的iOS平台浏览其中进行播放，影响了系统的兼容性。且方法一中录制程序繁琐，给服务器造成很大的负担。方法二将视频格式由FLV变成了ts格式，解决了兼容性问题。将录制程序放到一个单独的线程中运行，使程序更加稳定。但是由于其录制过程会产生累计误差，导致录制时间与系统时间偏移较大。因此最终选用方案三中的方式进行录制。

### 回看录制模块的实现

回来看服务器中的录制程序使用Python3.5进行编写，服务器选用flask框架用于与管理服务器进行通信。程序模块图如图所示：

录制程序共分为HTTP服务器模块，数据库模块，EPG更新模块，日志模块以及主录制模块

（1）服务器模块

服务器模块主要负责创建HTTP服务器与管理服务器通信。接收管理服务器发送来的录制信息并返回录制程序的状态。

服务器模块的主要方法如下。

def status()

def start()

def kill()

当管理服务器请求录制程序状态时，status方法返回指定节目的录制状态，如“运行中”、“已停止”、“发生错误”等等。

当管理服务器发来开始录制请求时，该方法根据所请求的参数分别调用start和kill方法启动和停止节目的录制。

（2）数据库模块

数据库模块主要负责数据库相关操作。服务器模块包含的主要方法如下：

def get\_udp\_port(channel\_id)

def set\_start(channel\_id,active)

def get\_available\_program(channel\_id,START\_TIME)

def delete\_program(channel\_id):

def insert\_program(event\_id, channel\_id, st, et, title)

def delete\_expire\_program(channel\_id,expire=8)

函数get\_live\_url主要负责根据输入的channel\_id参数获取udp流端口信息信息。如果获取成功，则可以监听本机的该udp端口获取到待录制的视频流。函数set\_start函数负责设置指定channel\_id的录制状态，其中0代表未运行，1代表运行中，2代表已运行。函数get\_available\_program函数负责获取当前可用的epg信息用于生成相应的m3u8文件。函数delete\_program负责删除指定频道未录制的epg信息以便更新最新的epg信息。函数insert\_program负责添加指定channel\_id的epg信息。函数delete\_expire\_program用来删除陈旧的epg信息，默认的过期时间为8天。

（3）EPG更新模块

EPG更新模块主要负责定时更新EPG信息。因为电视节目是处于不断变化之中的，所以存储在数据库之中的EPG信息也需要不断的更新。EPG模块的主要函数update伪代码如下所示:

def update(channel\_id):

try:

tree = et.parse(ur.urlopen(EPG\_URL + channel\_id))

except:

return

if tree == None:

return

dbutil.delete\_program(channel\_id) #删除未录制的视频

# 解析epg信息

root = tree.getroot()

now = datetime.now()

for schedule in root.iter('schedule'):

parseEPG()

# 如果该epg信息的结束时间大于系统当前时间，则将该条epg信息添加到数据库。

if end\_dt > now :

dbutil.insert\_program(event\_id,channel\_id,start\_dt,end\_dt,title)

首先，函数update根据输入的channel\_id参数像EPG服务器中请求对应的epg信息。如果获取失败则返回。接着删除数据库中未录制的EPG信息。最后解析获取到的新的EPG信息，遍历所有EPG信息，如果该EPG信息的结束时间大于系统当前时间，则将该条epg信息添加到数据库。

（4）日志模块

日志模块提供了通用的日志系统。底层调用python的logging模块，该模块提供不同的日志级别，还可以方便的调整日志文件输出的内容。日志模块进一步的对logging模块进行封装。根据程序的需求建立不同的日志文件，方便随时观察程序运行情况，观察可能出现的问题。日志模块包含的getLogger函数伪代码如下所示：

def getLogger(filename,name=None):

if name == None:

logger = logging.getLogger()

else:

logger = logging.getLogger(name)

formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')

fh = logging.FileHandler(filename)

fh.setLevel(logging.DEBUG) #设置日志显示级别

fh.setFormatter(formatter) #设置日志显示格式

logger.addHandler(fh) #设置日志文件

return logger

该函数根据输入的日志文件名初始化一个日志对象。在初始化的过程中分别设置好日志的级别以及日志的显示格式并返回该日志对象。在程序中只需要调用logger的debug函数即可将调试信息打印到日志文件之中。调试过程中将日志设为DEBUG级别，logging模块将输出所有步骤的运行信息，以便开发过程中判断出现问题的步骤。当程序调试完成后，将日志设为INFO级，则DEBUG日志内容不再保存，只留下日常所需的日志信息。

（5）主录制模块

主录制模块负责程序的最核心部分：将UDP流进行切片，生成一个个ts文件，并根据数据库中的EPG信息生成可供回看的m3u8文件。主录制模块核心函数main函数伪代码如下所示：

def Dump2(channel\_id)：

port = dbutil.get\_udp\_port(channel\_id)

if port == None:

error\_map[channel\_id] = "wrong stream"

dbutil.set\_start(channel\_id, False)

return

channel\_path = html\_path + channel\_id

if not os.path.exists(channel\_path + '/'):

os.makedirs(channel\_path +'/')

try:

while True：

res = dump(port, channel\_path.encode(), 60)

if res != 0:

error\_map[channel\_id] = "no stream"

time.sleep(60 \* 5)

finally:

dbutil.set\_start(channel\_id, False)

首先，函数根据传入进来的channel\_id参数，从数据库中获取对应的client\_ip字段，从client\_ip字段中解析出对应的UDP端口，接着，程序进入无限循环中，调用dump()函数将udp流录制为长度为一分钟的ts文件。当出现错误（流断开或者网络异常）时，dump函数返回，更改当前录制状态，延时五分钟之后，继续执行dump函数。由于python的性能限制，并不能用来处理节目的录制，所以dump函数使用速度较快的c语言完成，dump函数的声明如下所示：

int dump(int iPort, char\* channel\_path, int duration);

该函数被调用时，启动一个线程监听iPort端口，接受iPort端口传送来的数据并把其切片成ts文件。生成的文件存放在程序创建的channel\_path文件夹下当前日期的子文件夹下，如果文件夹不存在则创建一个新的文件夹。duration为ts文件的视频时间。当iPort端口没有数据超过一分钟，该函数返回-1。

## 本章小结

本章详细分析了流媒体服务器与回看录制系统的系统结构。介绍了直播流媒体服务器的实现与EPG同步服务器的实现。详细设计了回看录制系统，其中包括回看录制软件的设计、数据库设计以及回看录制模块的设计，并给出具体实现。

# 客户端播放器的设计与实现

## 概述

为了用户能够观看流媒体服务器上的流媒体视频节目，需要设计实现跨平台的视频播放器客户端，使用户能够在不同的终端平台上（如PC、手机，电视机顶盒）都能够流畅的观看电视直播与回看节目。系统需求如下：

（1）支持跨平台播放，用户能够在PC、手机浏览器、手机客户端、电视机顶盒等客户端之中自由选择。

（2）支持选择与流畅播放指定直播节目。

（3）支持显示与流畅播放指定节目最近7天的回看节目。

（4）支持回看节目的快进与快退。

客户端播放器共分为pc网页播放器、手机浏览器播放器，安卓手机客户端播放器以及安卓机顶盒客户端播放器。

## 网页播放器

### 整体结构分析

为了使用户在PC上和手机上都能有良好的播放体验。网页播放器分别设计两款UI界面供用户观看。

网页播放器使用HTML+CSS+JAVASCRIPT语言实现。视频播放框架使用开源的videojs框架以及扩展的videojs-hls扩展插件。videojs是一个几乎兼容市面上所有浏览器的HTML5播放器插件。可以使用CSS轻松定制皮肤。videojs在不支持HTML5的浏览器上自动切换成flash进行播放。支持H.264、FLV等主流视频格式，极大地提高了浏览器对视频播放的兼容性。

### 功能模块实现

（1）视频播放模块

视频播放模块负责控制视频的播放，获取网络流信息到本地，解码后进行播放。videojs的初始化与播放函数如下所示：

在系统初始化加载过程中，调用InitPlyaer函数完成对指定控件的初始化操作。在初始化的过程中制定好播放控件的宽和高。在需要播放视频的时候调用Play函数进行播放即可。

function InitPlayer()

{

playerInstance = videojs('myElement',{ "height": video\_height, "width": video\_width });

}

function Play(url)

{

playerInstance.src({

src: rtmp2hls(url),

type: 'application/x-mpegURL',

withCredentials: false

});

playerInstance.load();

}

（2）节目信息获取模块

节目信息的获取使用JavaScript开源库zeptojs。Zepto是专门为现代智能手机浏览器退出的 Javascript 框架, 拥有和jQuery相似的语法,大小却比jQuery小很多。使用zeptojs相关的AJAX函数，可以在不跳转页面的条件下轻松的完成数据获取操作。AJAX即“Asynchronous Javascript And XML”，是指一种新型的异步网页开发技术。通过在后台与服务器通过对应的数据协议进行少量的数据交换，AJAX可以使网页实现异步更新。这意味着可以在不刷新整个页面的情况下对网页的某个部分进行刷新，从而改变网页的内容。传统的网页如果需要更新内容，必须重载整个网页页面。使用zepto的getJSON函数获取直播信息过程如下：

$.getJSON(CHANNEL\_URL, function(data){

$("#channel-list").remove();

$.each(data.category, function(index, item){

$("#channel-list").append(channel);

channel.on('click',function(){

UpdateChannel(item.channel\_id,item.channel\_name,item.rtmp\_url);

});

});

}

首先，程序访问管理服务器的直播频道信息接口。将获取下来的频道信息依次添加到界面上的channel-list列表上，然后在每一条频道信息上添加点击监听。当用户点击每个频道是，调用UpdateChannel函数，程序会访问管理服务器的回看节目信息接口，将最近7天的回看信息下载下来并存储在对应的列表之中。当用户选择日期时，程序将相应回看频道列表的信息添加到回看节目频道列表上供用户点击观看。

## 安卓手机与机顶盒播放器

### 安卓系统组件

安卓系统是Google公司开发的基于 Linux 的开源手机操作系统。其系统采用了软件堆层(software stack，又名软件叠层)的架构[32]。Android系统[架构](http://lib.csdn.net/base/architecture)为四层结构，从上层到下层分别是应用程序层、应用程序框架层、系统运行库层以及Linux内核层。Linux内核层仅提供基本功能其核心系统服务如安全性、内存管理、进程管理、网路协议以及驱动模型都依赖于Linux内核。系统运行库是应用程序框架的支撑，是连接应用程序框架层与Linux内核层的重要纽带。应用程序框架层是从事Android开发的基础，很多核心应用程序也是通过这一层来实现其核心功能的，该层简化了组件的重用，开发人员可以直接使用其提供的组件来进行快速的应用程序开发，也可以通过继承而实现个性化的拓展。应用程序层则是构建在最上面的应用程序。

Android四大基本组件分别是Activity、Service服务、Content Provider内容提供者与BroadcastReceiver广播接收器。

（1）Android：Activity组件是系统中最常用、使用最频繁的基本组件。在一个 Android应用程序之中，一个Activity 表示一个可视化的界面，也就相当于是一个单独的屏幕。它是 Android 应用程序的基本功能单元。因为一个应用程序通常包含多个界面，所以大多数的应用程序都会包含多个 Activity类。不同的Activity之间使用Intent进行通信。Intent负责对操作的动作、动作所涉及到的数据以及附加的数据进行描述。系统则根据此Intent的描述，找到相对应的组件，并将 Intent传递给应该调用的组件，并完成组件的调用。因此，Intent在这里起着一个媒介的功能，在组件之中传递有用的信息。

（2）Service组件是一个没有用户界面的在后台运行执行耗时操作的应用组件。Service组件不需要与用户进行交互，它是一个没有时间限制的运行在系统后台的服务。其他应用组件能够启动Service，并且当用户切换到另外的应用场景时，Service将持续在后台运行。Service可以在后台处理一系列耗时操作如网络操作、播放音乐、操作文件I/O或者与内容提供者（Content Provider）交互等。

（3）Broadcast Receiver是为了实现广播接收而提供的一种组件。Android系统在运行的过程中，会产生很多事件，比如开机、电量改变、收发短信、拨打电话、屏幕解锁等。系统在产生某个事件时发送广播，应用程序使用广播接收者接收这个广播，就知道系统产生了什么事件。

（4）Content Provider组件主要用于在Android 应用程序间共享数据。ContentProvider把应用中的数据共享给其他应用访问，其他应用可以通过ContentProvider对指定应用中的数据进行操作。整体结构实现

为了使终端用户可以随时随地的观看视频节目，使用Android Studio制作安卓手机与安卓机顶盒的客户端应用程序。

该应用程序整体机构设计如图所示如图4-1所示：

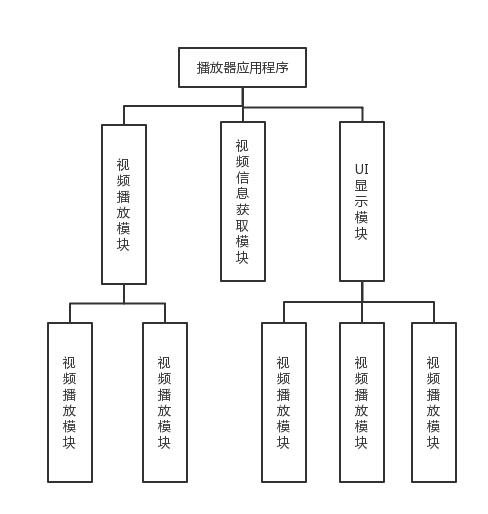


图 4-1 安卓客户端应用程序模块图

共分为三大模块，分别为视频播放模块、视频信息获取模块以及UI界面模块。其中，视频播放模块包括播放显示模块和播放控制器模块，播放显示模块负责将视频从网络上获取下来进行解码并显示在屏幕上。播放控制模块则负责显示当前视频节目的播放进度并控制其跳转与快进快退。视频信息获取模块负责从管理服务器提供的数据接口中提取相应的视频信息，传送给UI界面模块。UI界面模块负责显示节目信息以及控制视频节目的播放。

### 功能模块实现

安卓客户端的整体架构如图4-2所示。MainActivity为应用程序入口。MainActivity内包含类VideoView、类SidebarFragment、类ReplayController、类InfoFragment以及类ErrorFragment所生成的对象。视频播放模块由类VideoView实现，内部包含类MediaPlayer和类MediaController。MediaPlayer负责视频的解码与显示，MediaController负责视频的播放控制（如快进、快退）。视频信息获取模块由类FetchInfoTask实现，类FetchInfoTask将服务器上提供的直播与回看视频的信息获取下来，存入相应的数据结构之中，以便显示模块的显示和播放模块的播放。UI显示模块由SidebarFragment、ReplayController、Infofragment以及ErrorFragment实现。其中，SidebarFragment用于显示视频节目信息，ReplayController用于显示回看视频进度条，InfoFragment用来显示直播频道信息，ErrorFragment用来显示播放错误界面。

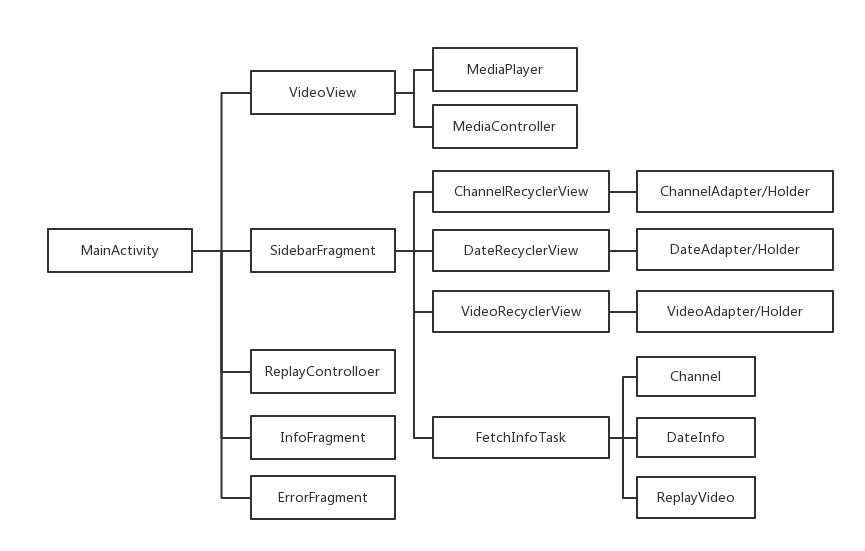


图 4-2 安卓客户端应用程序架构图

程序启动时，在MainActivity类中的onCreate方法中初始化各个模块组件。使用MainActivity中的FragmentManager将创建好的Fragment添加进MainActivity的界面中，并将各个Fragment隐藏。接着初始化VideoView组件用来进行视频的播放。初始化VideoView的步骤如下：（1）加载视频解码so库。（2）注册视频播放完成与视频播放失败的监听函数。（3）播放默认的视频节目。

此时，应用程序播放默认的电视直播节目。当使用遥控器按下上方向键或者下方向键时，程序执行切换直播频道的操作同时显示InfoFragment显示当前直播频道名称并在两秒后隐藏。当按下确定键或者右方向键时，SidebarFragment显示出来并获得焦点。此时可以在SidebarFragment中进行频道选择的或者回看视频选择的操作。在播放回看视频的状态下，使用遥控器按下左方向键或者右方向键时，程序将ReplayController界面显示出来并执行快进快退的操作。

在MainActivity中，实现了SidebarFragment的PlayVideoCallBack接口，用于SidebarFragment与MainActivity之间的通信。其中包括播放视频、视频跳转以及按键监听等操作。当程序退出即MainActivity销毁时，MainActivity中的onStop函数将会被调用，在onStop函数中，释放与VideoView组件相关的资源，以防发生内存泄漏。

视频播放模块、数据获取模块以及UI显示模块的实现如下：

（1）视频播放模块

安卓客户端中的视频播放使用VideoView实现，内部封装MediaPlayer进行视频的播放的控制。视频的解码由开源解码器FFmpeg实现。MediaPlayer的状态转换图如图4-3所示：

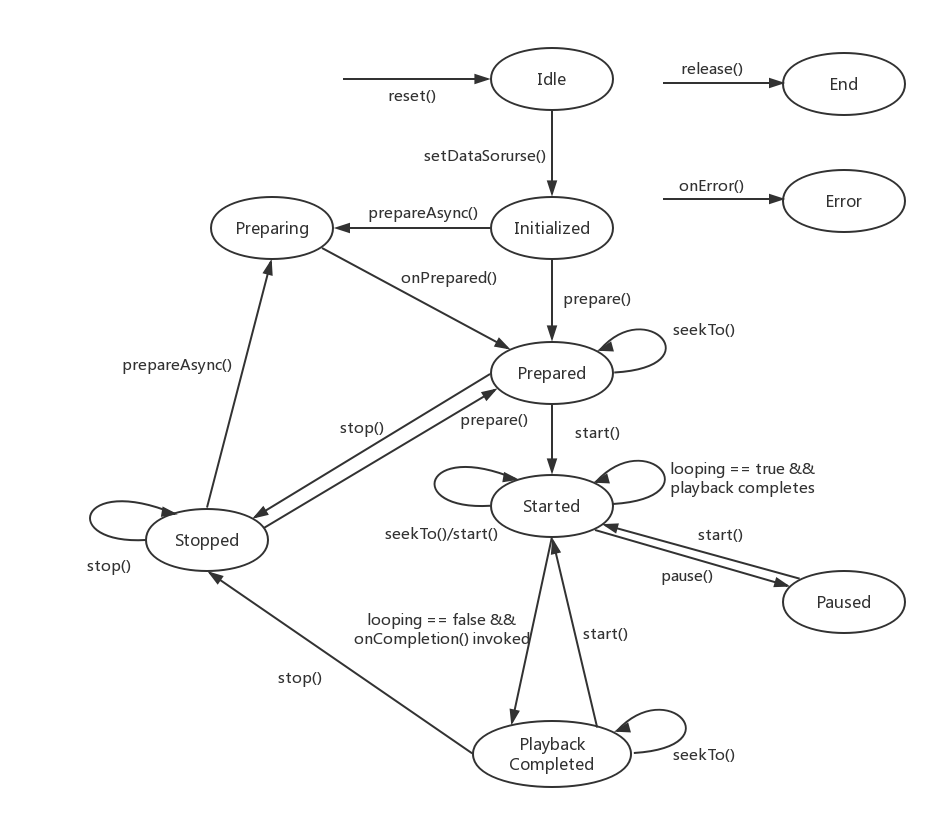


图 4-3 MediaPlayer状态转换图

这张状态转换图清晰的描述了MediaPlayer的各个状态，也列举了各个状态之间的转换关系。每种方法只能在一些特定的状态下使用，如果使用时MediaPlayer的状态不正确则会引发IllegalStateException异常。

Idle状态：当程序使用new方法新建了一个MediaPlayer对象或者调用了其reset方法时，该MediaPlayer对象处于idle状态。new方法和reset方法的一个重要差别就是：如果在idle状态下程序调用了seek等方法（等于出现了错误的函数调用时机），通过reset方法进入idle状态的话，OnErrorListener.onError方法将会被调用，并且MediaPlayer会进入Error状态。如果是通过new方法新建的MediaPlayer对象，程序则并不会触发onError错误，也不会进入Error状态。

End 状态：程序通过调用release方法可以进入End状态。只要MediaPlayer对象不再被调用，就应当通过release方法，释放掉相关的程序资源。如果MediaPlayer对象通过release方法进入了End状态，则不会再进入MediaPlayer所包含的其他状态了。

Initialized 状态：该状态表明要播放的文件已经被设置好。MediaPlayer通过调用setDataSource方法就进入此状态。

Prepared 状态：MediaPlayer初始化完成之后，程序还需要通过调用prepare方法或或者prepareAsync方法。其中prepare方法是同步的而prepareAsync方法是异步的。进入Prepared状态之后，说明MediaPlayer没有发生错误，可以进行后续的播放操作。

Preparing 状态：Prepare状态表示MediaPlayer正在准备准备中，通过prepareAsync方法进入该状态，如果prepareAsync方法调用完成之后，程序将会触发onPrepared回调函数，从而进入Prepared状态。

Started 状态：MediaPlayer如果一切准备就绪，便可以调用start方法进行播放。程序通过调用start方法进入Started状态。程序可以使用isPlaying方法测试MediaPlayer是否处于Started状态之中。如果程序设置了循环播放，那么在播放结束时，MediaPlayer仍然会处于Started状态并且继续播放视频。同样的的，如果在Started状态下程序调用了seekTo方法或start方法，都可以让MediaPlayer继续停在Started状态。

Paused 状态：在Started状态下，程序调用pause方法可以暂停播放器的播放，从而使MediaPlayer进入Paused状态。播放暂停后，再一次调用start方法便可以继续M播放器的播放，进而转到Started状态。暂停状态时，程序可以调用seekTo方法，这时MediaPlayer依然处于Paused状态。

Stop 状态：当程序调用stop方法时，MediaPlayer便从Started状态或者Paused状态转到Stop状态。而处于Stop状态的MediaPlayer如果想要重新进行播放，需要通过调用prepareAsync方法或者prepare方法回到Prepared状态重新开始，才可以重新进行播放。

PlaybackCompleted状态：如果程序没有设置循环播放，且视频正常播放完毕，则MediaPlayer进入PlaybackCompleted状态。OnCompletionListener中的的onCompletion方法也会被触发。在该状态下，可以调用start方法继续重新从头播放视频文件。也可以使用stop方法停止MediaPlayer。或者使用seekTo方法来重新指定播放的时间位置。

Error状态：如果由于某些原因，MediaPlayer发生了一些错误，程序将会触发OnErrorListener中的onError事件。此时MediaPlayer便进入了Error状态。在Error状态中，程序可以及时捕捉并妥善处理这些错误。在Error状态中，调用reset方法可以使MediaPlayer重新回到Idle状态，进而重新播放视频文件。

MediaPlayer底层调用开源视频编解码软件FFmpeg进行视频解码。

FFmpeg解码线程创建图如图4-4所示：

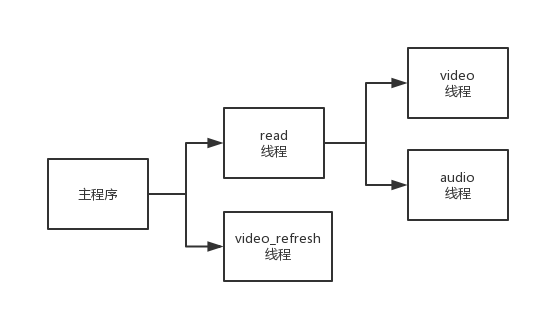


图 4-4 FFmpeg解码线程创建图

FFmpeg视频解码过程分为网络数据读取与视音频解码与播放两个步骤。在解码开始时，程序首先创建了两个线程，分别为video\_refresh线程与read线程。video\_refresh线程为视音频同步显示线程，read则为数据读取线程。在read线程中，程序又创建了video线程与audio线程分别用于视频信息与音频信息的解码。

在网络数据读取线程中，程序首先分别初始化一个音频队列和一个视频队列，读取线程将获取到的音频帧和视频帧分别压倒队列中以便进行解码与播放。接着读取线程调用FFmpeg中的avformat\_open\_input函数判断视频源格式，调用avformat\_find\_stream\_info函数解析流并找到相应解码器，调用stream\_component\_open创建音频和视频的解码线程。最后程序循环调用av\_read\_frame函数，将视音频信息从视频流中读取出来。

在video线程和audio线程中，程序从视音频队列中取出帧信息，调用decoder\_decode\_frame函数进行视音频的解码。解码后放入相应的队列中，由jni调用上层java函数进行播放。

类MediaController负责进行播放控制，如快进、快退、全屏等操作。类MediaController对象初始化时，首先调用initFloatingWindowLayout方法和initFloatingWindow方法初始化整个窗口的布局。接着调用setMediaPlayer方法指定播放器，调用setAnchorView方法指定播放视图（VideoView）。在setAnchorView方法中，使用makeControllerView方法初始化播放控制视图。播放控制视图中一共包括三个控件：第一，控制播放开始和暂停的开始/暂停按钮。第二，控制播放进度的进度条控件。第三，切换全屏显示的全屏按钮。在手机播放器客户端中，当用户点击播放界面，会弹出播放控制条。用户可以通过开始/暂停按钮控制视频的播放，通过播放进度条改变视频的播放进度。当用户按下全屏按钮是，可将视频变换为全屏播放。

（2）视频信息获取模块

该模块使用异步任务AsyncTask实现。当一个应用程序第一次启动时，安卓程序会创建一个的主线程（Main Thread），该主线程主要负责处理和UI相关的事件，比如用户的按键事件和接触屏幕的事件以及屏幕的绘图事件等等，并把一系列相关的事件分发到对应的组件中进行处理。主线程通常又被叫做UI线程。在开发安卓应用时，程序必须遵守单线程模型的原则。由于Android中的UI操作并不是线程安全，所以必须在主线程中执行相应的UI操作。如果在非UI线程中直接进行UI操作，程序则会抛出异常。由于UI线程主要负责事件的监听和绘图，为了达到良好的用户体验，程序必须保证UI线程能够随时响应用户的需求，因此，UI线程里的操作都应该简单且耗时短。一些费时的操作（如网络连接、文件读取等）需要在新建的线程完成，否则，如果UI线程长时间没有响应用户的请求，应用程序便会弹出对话框，提醒用户终止该应用程序。

为了不再主线程中执行耗时的信息获取操作，我们使用安卓提供的AsyncTask异步任务异步地获取信息，程序结束之后再讲结果返回给主线程。使用异步任务获取到直播与点播的视频信息之后，需要把信息存入对应的数据结构之中。

程序中的模型类Channel代表直播频道的信息。类DateInfo代表会看日期对象，类ReplayVideo代表会看节目的视频信息。每个Channel对象维护一个长度为7的DateInfo列表，分别代表最近7天的回看节目容器，列表中的每个DateInfo对象中维护着一个ReplayVideo的列表，列表中存储着当天的回看节目信息。

在AsyncTask中的doInBackground方法中，执行信息获取的操作，首先，执行FetchChannels方法得到所有直播频道的信息，存储在全局的Channel列表中。如果执行成功，则执行FetchVideos方法，遍历所有频道，获取每个频道最近7天的视频回看信息，并存储在各个Channel对象中对应日期的ReplayVideo列表中。

（3）UI显示模块

UI显示模块主要负责显示直播频道与回看节目的列表信息。程序中使用RecyclerView显示相应的列表信息。RecyclerView是Android L版本中新添加的一个用来取代ListView的SDK,它的灵活性与可替代性比ListView更好。RecyclerView的基本结构如图4-5所示:

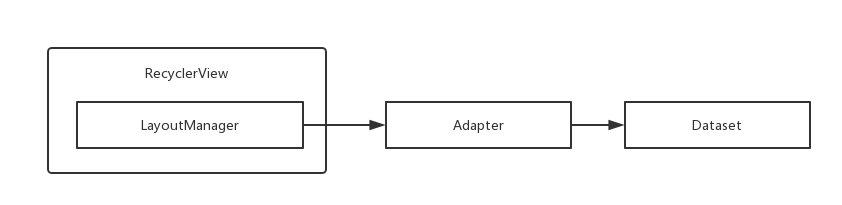


图 4-5 RecyclerView基本结构图

对于ListView来说，通过创建ViewHolder来提升性能并不是必须的。因为ListView并没有严格的ViewHolder设计模式。但是在使用RecyclerView的时候，Adapter必须实现至少一个ViewHolder，必须遵循ViewHolder设计模式。ListView只能实现垂直线性排列的列表视图，与之不同的是，RecyclerView可以通过设置RecyclerView.LayoutManager来定制不同风格的视图，比如水平滚动列表或者不规则的瀑布流列表。在ListView中针对不同数据封装了各种类型的Adapter，比如用来处理数组的ArrayAdapter和用来展示Database结果的CursorAdapter。在RecyclerView中必须自定义实现RecyclerView.Adapter并为其提供数据集合。

以初始化ChannelRecyclerView为例，首先，创建数据封装类ChannelHolder，类ChannelHolder中包含每个列表项应该显示的视图。接着创建适配器类ChannelAdapter，类ChannelAdapter将数据模型与显示界面绑定在一起。在类ChannelAdapter中，需实现onCreateViewHolder与onBindViewHolder两个方法。方法onCreateViewHolder负责在程序初始化的时候，创建ViewHolder并加入内存中。方法onBindViewHolder负责在显示的时候，将数据与ViewHolder绑定。最后，初始化ChannelRecyclerView对象，将ChannelAdapter与ChannelRecyclerView绑定，对应的直播频道节目信息就会显示在应用程序界面上。

## 本章小结

本章详细介绍了各个客户端播放器的设计与实现。分别详细设计了适合PC与手机浏览器播放的网页播放器和适合安卓手机和机顶盒播放的安卓播放器应用程序。在网页播放器中使用videojs作为播放库，保证了视频播放在各个浏览器上的兼容性。在安卓播放器应用程序中，使用FFmpeg软解码，保证了视频播放在各个安卓系统之中的兼容性。

# 系统测试与分析

## 系统功能测试

系统测试是将计算机硬件、软件、外设、网络等其他元素结合在一起，进行信息系统的各种组装测试和确认测试。通过与制定的系统需求相比较，系统测试发现所开发的系统与用户需求不符或矛盾的地方，从而提出更加可靠的方案并且加以改进。

系统测试的任务是尽可能彻底地排查出程序中的错误，提高软件系统的可靠性。目前测试仍然是保证系统质量的关键步骤。它是对系统需求规格、设计和编码最后的验证、复审。系统测试集中反映了人们心理上、技术上、经济上对系统的认识，这种认识在很大程度上又影响了系统的设计。 实验室邀请多人次长期对本系统进行功能测试。在测试的过程之中发现了一些系统漏洞，但均通过重新修改源程序将其弥补。近一步的系统测试将在今后的用户使用阶段来完成。

### 服务器系统功能测试

测试使用硬件平台如下：CPU使用Intel i7处理器，硬盘使用1T SCSI硬盘。网卡使用1000m网卡，与IEEE 802.1p标准兼容。服务器软件平台如下：操作系统使用Ubuntu 16.04，流媒体服务器使用开源流媒体服务器srs，http服务器使用开源HTTP服务器nginx。

管理服务器页面如下图5-1所示：

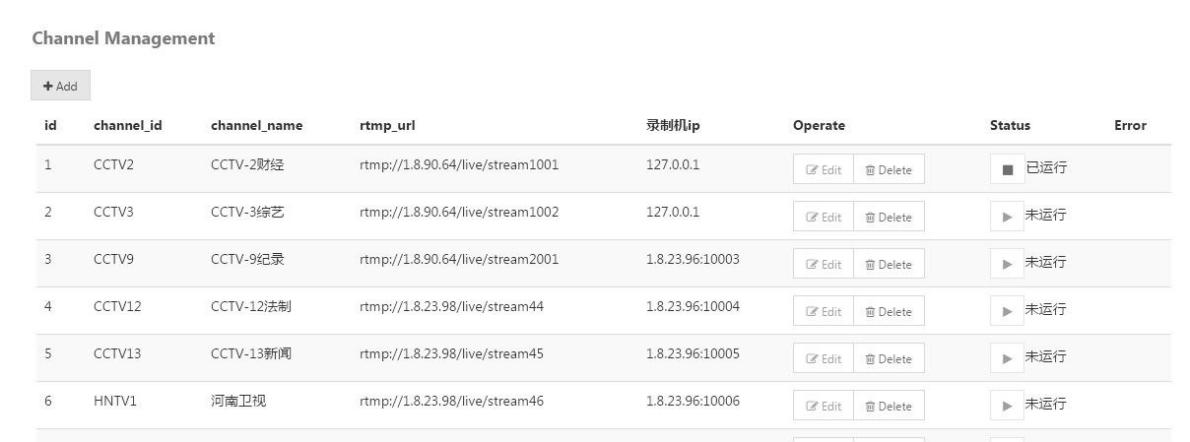


图 5-1 管理服务器页面效果图

分别对管理软件各项功能模块进行详细的反复测试。实践证明管理系统可以达到添加、编辑、删除直播频道信息以及控制回看节目录制等功能。性能稳定，出错后能够及时反馈。提供详细的系统日志以便于寻找潜在的问题。

## 服务器压力测试

### srs-bench 简介

srs-bench是一个流媒体服务器负载测试工具，模拟huge并发：2G内存就可以开300k连接。基于states-threads的协程。支持HLS解析和测试，下载ts切片后等待一个切片长度，模拟客户端。支持HLS点播和直播。支持HTTP负载测试，所有并发重复下载一个http文件。可将80Gbps带宽测试的72Gbps。支持RTMP流播放测试，一个进程支持5k并发。支持RTMP流推流测试，一个进程支持500个并发。

### 压力测试结果分析

使用srs-bench进行服务器的压力测试，分别进行100路、500路、1000路并发测试。实验结果显示，流媒体服务器满足性能要求。

对点播服务器进行16路录制测试，实验结果显示，CPU占用率只有3.1%。而使用RTMP作为直播员的录制框架CPU占用率则高达35.3%。可见使用UDP协议作为直播源进行录制的软件架构，性能相比使用RTMP作为直播源的软件架构，性能提升了1000%。

## 客户端播放器功能测试

将客户端测试机与直播点播流媒体服务器构建在同一局域网内，确保可以访问到流媒体服务器。

使用手机浏览器登录URL地址为http://tv.com/replay/replay/index.html的网页。效果如图5-2所示：



图 5-2 手机客户端页面效果图

图5-2（a）为播放主界面，点击直播标签切换至直播页面，点击回看标签则切换至回看页面。图5-2（b）为回看节目列表，列出当前日期的所有回看节目信息。图5-2（c）为频道列表，列出当前所有可选的电视频道，用户点击可以切换不同的频道进行观看。

使用PC浏览器登录URL地址为http://tv.com/replay/pc/index.html的网页，效果如图5-3所示：



图 5-3 PC客户端页面效果图

使用安卓手机安装本课题开发的apk应用程序，指定好管理服务器的ip地址。应用程序界面如图5-4所示：

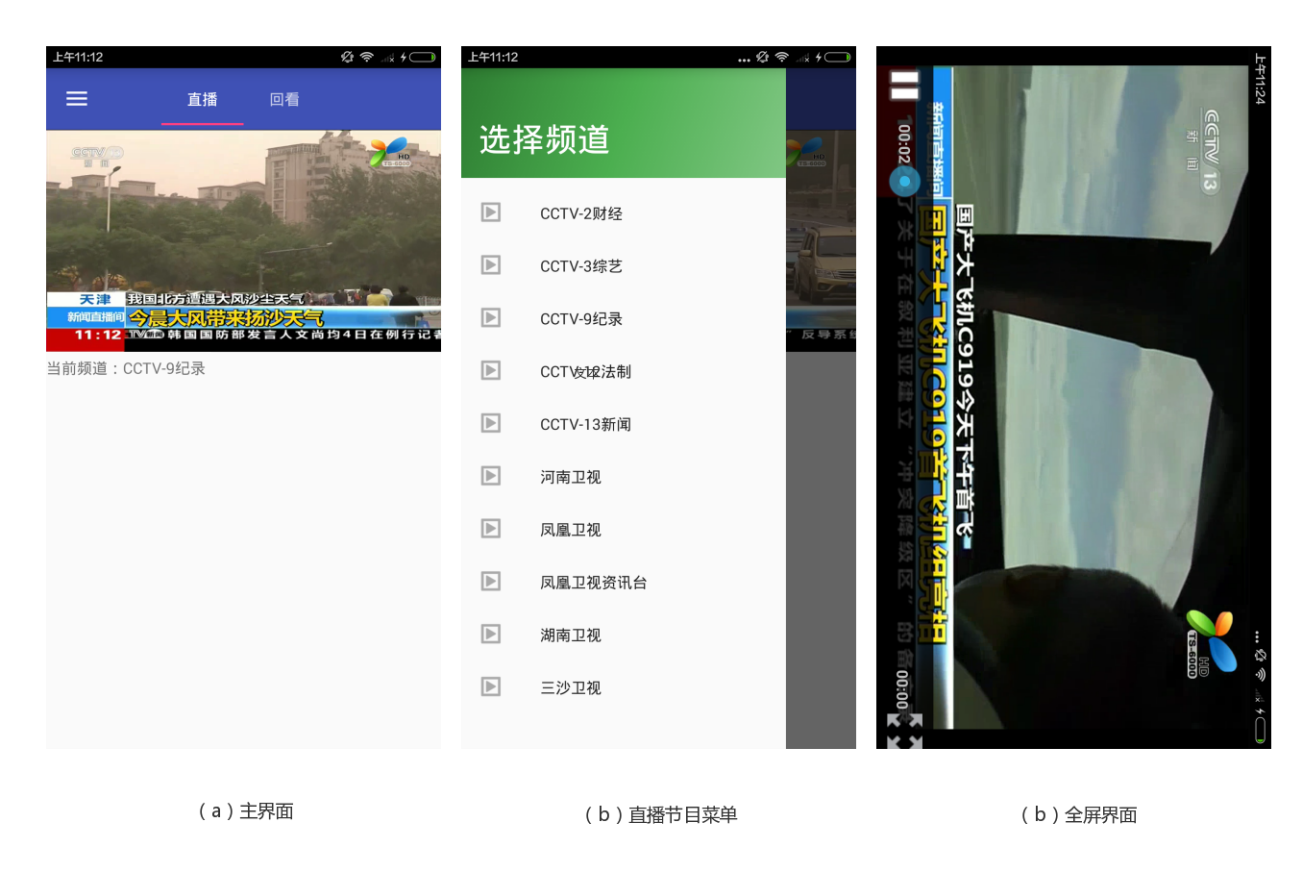


图 5-4 安卓客户端界面效果图

图5-4（a）为播放主界面，滑动屏幕可以在直播和回看两个选项卡中来回切换。图5-4（b）为回看界面，底部的列表列出当前日期的回看节目列表。图5-4（c）为全屏播放界面，当点击全屏按钮或者屏幕横置时，可进入全屏页面。

使用安卓机顶盒安装本课题开发的APK应用程序，指定好管理服务器的ip地址。应用程序界面如图5-5所示：



图 5-5 安卓机顶盒客户端界面效果图

初次进入程序，客户端播放器会自动播放直播节目，使用遥控器按下确定键或者右方向键，进入频道选择菜单。选择对应的直播频道，继续按下右方向键可以进入回看节目菜单。选择对应的选项即可观看回看节目。在观看回看节目时，按下左右方向键可进行快进快退选择。在观看直播节目时，按下上下方向键可进行直播频道的选择。

## 本章小结

本章对系统进行了一系列的功能测试以及压力测试。对回看录制管理软件进行详细的系统测试，保证功能的完善与稳定。对流媒体服务器进行压力测试，确保流媒体服务器运行稳定。最后对客户端播放器进行功能测试，保证播放器在不同的系统下均可以流畅播放视频节目。

# 结论与展望

## 工作总结

本文首先介绍了如今市场上流媒体现今的发展现状以及发展趋势。随着上高清电视显示器的普及、视频编码压缩效率的提高、以及网络传输速度的提升，通过互联网观看电视直播节目成为了可能。为了满足普通用户通过互联网观看电视直播节目的需求，本课题实现了将电视信号转化为网络流的流媒体服务器，并在此基础上进一步实现了回看录制系统。本文对RTMP以及HLS协议、流媒体服务器、以及客户端软件开发进行了深入的研究，并做了如下工作：

（1）在Linux系统之上编译并部署了简单可靠的流媒体服务器，使之将电视节目信号变为流媒体供客户端用户观看。

（2）实现了一套回看录制系统，该系统使用Python语言实现，将电视直播源根据EPG信息录制成视频文件，并存储在本地HTTP服务器供客户端观看。

（3）详细比较的RTMP与HLS协议各自的优缺点：RTMP协议的延时较小，但是播放时必须借助FLASH插件，消耗系统资源较多且兼容性较差。HLS协议兼容性较好，但是延时较大。相比之下，我们在PC端选择RTMP协议进行播放而在移动端我们则选择HLS协议进行播放。在回看录制系统的方案选择过程中，出于对系统性能的考虑，使用UDP传输TS流作为直播源进行回看节目的录制。

（4）实现了多平台的客户端播放器。保证了普通用户可以在任何终端之下观看视频节目。其中分别实现了PC和手机端的网页播放器客户端，以及安卓手机和机顶盒的播放器客户端。在安卓系统平台上，我们使用FFmpeg进行对视频进行软解码并播放，以保证播放器在安卓系统上的兼容性。

## 后期展望

由于网络直播近年来的爆炸式增长，基于流媒体服务器的网络直播需求必定会越来越多。本文所开发的电视节目直播与回看录制系统满足了普通用户通过互联网观看电视直播节目以及回看节目的需求。但是现阶段的系统设计只是针对单个流媒体服务器而言的，一台服务器最多只能支持三千到五千个用户同时观看。

在下一个阶段，为了支持更多的用户同时在线观看，在本课题的基础上，进一步架构一个服务器集群，其中直播流媒体服务器使用Forward方式搭建小型集群，即主服务器把接收到的所有流都发送给从服务器，从而使从服务器也拥有了直播流。最后使用DNS服务器进行负载均衡，使不同来源的IP地址响应不同节点，从而实现流媒体服务器的负载均衡。

由于个人技术有限，该系统架构设计可能有些纰漏，服务器以及可用户端的功能还有一些不完善。望读者批评指正。上述不足，我均会在日后的研究中，努力学习，加以改进。

# 致 谢

三年的研究生生活就快要接近尾声了，这三年的历练，让我收获良多，不仅仅收获了知识，同时也收获了关心自己的朋友，三年间，充实的每一天让我感觉不到时光的流逝，不知不觉中就走完了三年，提到这三年中认识了许多帮助过我的朋友和老师，我心中就会充满暖意和感激，因此，在这里，怀揣着感恩的心，对那些帮助过我的人们郑重的说声谢谢你们。

这三年时光里，我首先需要感谢的是我的导师曹峻老师，您以严谨的管理，认真的面对每一个细节，对知识永远充满着激情，这些态度都深深的感染着我，让我看清了自己的许多不足，意识到自已之前不以未然的坏毛病，在这里我郑重的感谢导师这三年对我的监督催促和悉心教导，并且真心的向导师说一声对不起，这三年给您添了很多麻烦了，虽然是这样，可你依然至始至终都对我们充满着耐心，永远让我们感受到您的关怀，并且把您的知识都毫无保留的交给我们，这些年我们一起做项目，一起探讨， 一起学习，一起进步，虽然我们即将毕业，即将离开这个学校，但是我永远会记住您，记住您的处事态度，记住您对知识的执着，努力的向前走。

同时我还要感谢实验室里的同学李嘉杰的帮助，是你回答了我一遍又一遍的问题，对于我的问题，无论是简单还是复杂，你们都会耐心的帮助我，并且会在繁忙之中抽出时间给我详细分析讲解我的疑惑，有了你们的帮助才有了这个项目的完成，才有了我的进步。研究生的三年有你们，无论是在研究室的研究工作还是日常的生活工作，都充满了欢声笑语，开心的度过了每一天。这里不仅要感谢这些同学和老师， 更好感谢我的父母，是你们从小对我的教育和培养，一直到我现在，每时每刻都让我感受到了你们对我的关怀。你们的鼓舞给了我极大的动力，你们的安慰是我面对困难的信心。正是因为这些人的存在，这些人对我的帮助，我会一直告诫自己不要懈怠，要不断努力，不断进步，绝不辜负你们对我的期望，最后还要感谢评审此论文的老师们，感谢你们在百忙之中评阅我的论文并对我的论文提出了您宝贵的意见。

# 参考文献

1. 张印.基于RTMP协议的流媒体系统的设计实现[D].电子科技大学,2015.
2. 胡国栋.基于网络存储的流媒体服务器系统[D].浙江大学,2012.
3. 李志昂.HLS流媒体服务器的设计与实现[D].北京邮电大学,2014.
4. 杨戈,廖建新,朱晓民等.流媒体分发系统关键技术综述[J].电子学报, 2009, 37(1) :137 -145.DOI:10.3321/j.issn: 0372-2112.2009.01.025.
5. 秦臻,周帆,李乐民等.DNS对CDN流媒体服务质量的影响[J].电子科技大学学报,2013,(4):577-580.DOI:10.3969/j.issn.1001-0548.2013.04.009.
6. 赵辉,郑庆华,张未展等.多版本视频点播流媒体服务器集群资源分配方法[J].西安交通大学学报,2016,50(6):30-35.DOI:10.7652/xjtuxb201606005.
7. 张璇.基于智能手机的流媒体播放及编解码研究[D].南京邮电大学,2011.DOI:10.7666/d.d177964.
8. 杨光平.基于Android的移动流媒体实时传输系统设计与实现[D].西安电子科技大学,2012.DOI:10.7666/d.d216626.
9. 黄诗文.基于ffmpeg的高性能高清流媒体播放器软件设计[D].浙江大学,2012.
10. 许金.基于Android的流媒体播放器研究与设计[D].广东工业大学,2012.DOI: 10.7666/d.y2097522.
11. 曲柳莺.流媒体传输协议的研究[D].电子科技大学,2005.
12. 秦璐.基于iPhone OS系统的流媒体播放器应用软件设计[D].天津大学,2011.
13. 杨蓓.流媒体系统中音视频同步机制的设计与实现[D].华中科技大学,2008.DOI: 10.7666/d.d072124.
14. 宫小娜.Linux系统下实现浏览器中嵌入播放流媒体的技术研究[D].重庆大学,2013.DOI:10.7666/d. D356363.
15. 杨建锋.移动视频监控系统中流媒体传输控制的设计与实现[D].浙江工业大学,2012.DOI:10.7666/d.y2143003.
16. 申志伟.流媒体客户端的分析与研究[D].北京邮电大学,2007.
17. 黄艺.泛娱乐化时代网络直播平台热潮下的冷思考[J].新闻研究导刊,2016,7(2):106.
18. 金侠飞.我国网络直播平台可持续发展方式探究[J].科技传播,2016,(6):89-89,101.
19. 李芳芳,苏凯雄.基于FFmpeg的H.264格式转换器的设计与实现[J].电视技术,2016,40(7): 32-35.DOI: 10.16280/j. videoe.2016.07.008.
20. 刘红,卢程.FFmpeg开源项目移植到Android的研究[J].价值工程,2016,35(4):166-169.
21. 罗淑贞,耿恒山,徐祥男等.基于HLS协议的流媒体直播系统的研究和改进[J].郑州大学学报（工学版）,2014,35(5):36-39.DOI:10.3969/j.issn.1671-6833.2014.05.009.
22. Lei, Xiaohua,Jiang, Xiuhua,Wang, Caihong et al.Design and Implementation of a Real-Time Video Stream Analysis System Based on FFMPEG[C].//2013 Fourth World Congress on Software Engineering: 2013 4th World Congress on Software Engineering (WCSE 2013), 3-4 December 2013, Hong Kong.2013:212-216.
23. Hao Zeng,Yuan Fang.Implementation of video transcoding client based on Ffmpeg[C].//Information technology applications in industry, computer engineering and materials science: selected, peer reviewed papers from the 2013 3rd International Conference on Materials Science and Information Technology (MSIT 2013), September 14-15, 2013, Nanjing, Jiangsu, China.2013:1740-1744.
24. Huang J, Wu D, Liu X. Implementation of the RTMP server based on embedded system[C].//Computer Science and Information Processing (CSIP), 2012 International Conference on.IEEE, 2012:160 - 162.
25. M．Zhang，Y．Tang，L．Zhao．A Multi·sender based Peer—to·peer Muiticast System for Video Streaming．In Proceedings of IEEE International Conference on Multimedia&Expo 2005．MIT Press，2005，41(12)：1330-1339
26. Richardson I．H．264 and MPEG一4 Video Compression·Video Coding for Next·Generation Multimedia．Chichester：John Wiley and Sons，2003，284-29
27. Q．Wu．Design and implementation for embedded real-time video encoder H．264．Journal of Jiangxi University of Technology，2009，1 5：205-244．
28. Y．Hao．Design and Deployment of a Hybrid CDN—P2P System for Live Video Streaming:Experiences with LiveSky．Beijing，China,,2009:20-30．
29. 曾金．嵌入式流媒体服务器的设计和实现[D]．南京：南京邮电大学，2011
30. Jerry Whitaker. 曹晨译. 高清晰度数字视频原理与应用[M]. 北京： 电子工业出 版社，2002.
31. 吕敏．TS流实时低延迟无缝拼接的算法研究及验证[D]．广州：华南师范大学，2007
32. 谢文焘，董黎刚.基于 Android 的校园网移动客户端设计与实现[J].杭州电子科技大学学报， 2012.5

学位论文独创性声明（1）

本人声明：所呈交的学位论文系在导师指导下本人独立完成的研究成果。文中依法引用他人的成果，均已做出明确标注或得到许可。论文内容未包含法律意义上已属于他人的任何形式的研究成果，也不包含本人已用于其他学位申请的论文或成果。

本人如违反上述声明，愿意承担以下责任和后果：

1．交回学校授予的学位证书；

2．学校可在相关媒体上对作者本人的行为进行通报；

3．本人按照学校规定的方式，对因不当取得学位给学校造成的名誉损害，进行公开道歉。

4．本人负责因论文成果不实产生的法律纠纷。

论文作者（签名）： 日期： 年 月 日

学位论文独创性声明（2）

本人声明：研究生 所提交的本篇学位论文已经本人审阅，确系在本人指导下由该生独立完成的研究成果。

本人如违反上述声明，愿意承担以下责任和后果：

1．学校可在相关媒体上对本人的失察行为进行通报；

2．本人按照学校规定的方式，对因失察给学校造成的名誉损害，进行公开道歉。

3．本人接受学校按照有关规定做出的任何处理。

指导教师（签名）： 日期： 年 月 日

学位论文知识产权权属声明

我们声明，我们提交的学位论文及相关的职务作品，知识产权归属学校。学校享有以任何方式发表、复制、公开阅览、借阅以及申请专利等权利。学位论文作者离校后，或学位论文导师因故离校后，发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为西安交通大学。

论文作者（签名）： 日期： 年 月 日

指导教师（签名）： 日期： 年 月 日

(本声明的版权归西安交通大学所有，未经许可，任何单位及任何个人不得擅自使用)