

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：曹峻 教授

学科名称：通信与信息系统

2017年 6 月

**XXXXXXXXXXXXXXXXXXXX**（注：此处为论文题目的英文翻译，注意单词的大小写规律，如Structural Design and Rapid Development of Labyrinth Drip Irrigation Emitters）

A thesis submitted to

Xi’an Jiaotong University

in partial fulfillment of the requirements

for the degree of

Master of XXXXXXXXX（注：此处为学科门类的英文翻译，单词首字母大写，哲学：Philosophy；经济学：Economics；法学：Law；教育学：Education；文学：Literature；艺术学：Art；

理学：Natural Science；工学：Engineering；医学：Medicine；管理学：Management Science）

By

XXXXXX（注：此处为作者姓名全拼，名在前，姓在后，如Zhengying Wei）

Supervisor: Prof. XXX（注：此处为导师姓名全拼，名在前，姓在后，如Bingheng Lu）

(XXXXXXXXXXX) （注：此处为学科名称，如Mechanical Engineering）

XXX XXXX（注：此处为英文日期，月在前，年在后，如September 2003）

**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：通信与信息系统**

**学位申请人：丛延东**

**指导教师：曹峻教授**

摘 要

随着现代人们生活质量的提高，网络通信技术以及带宽技术的日益成熟化，流媒体服务器的应用也与日俱增，流媒体技术具有连续性和传输性，即流媒体技术可以利用网络实时的传递音频和视频数据，是优于其他传统多媒体下载传输的技术。流媒体技术突破了 TCP/IP 网络的限制，飞速发展，在视频点播直播、网络会议，智能家居，小区安全监测等等领域都发挥了重要的作用。

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网络用户观看。

（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）介绍了，RTMP与HLS的协议架构，并详细比较的两种协议各自的优缺点。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，我们放弃了使用RTMP和HLS作为直播源的传输协议，而是使用UDP传输TS流作为直播源。

（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。

**关 键 词**：XXX；XXX；XXX；XXX；XXX

**论文类型**：工程设计

**Title:** **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

**Descipline: XXXXXXXXXXX**

**Applicant: XXXXXXXXXXX**

**Supervisor: Prof. XXXXX**

ABSTRACT

英文摘要撰写要求如下：

（1）用词准确，符合语法；

（2）关键词按相应专业的标准术语写出，尽量从《英语主题词表》中摘选；

（3）如果论文的主体工作得到了有关基金资助，应用英文在摘要第一页的页脚处标注：本研究得到某某基金（编号： ）资助；

中文摘要和英文摘要均不要求学位申请人及其指导教师签字。

摘要正文每段开头不空格，每段之间空一行；

The key parts in drip irrigation facilities are emitters. The structural design parameters of emitters can directly affect its performance and the function of the whole drip irrigation system ……

1. Because……

2. Only ……

3. To support ……

**KEY WORDS**: XXX; XXX; XXX; XXX

每个关键词组的第一个字母大写，其余为小写，每一关键词之间用分号分开，最后一个关键词后不打标点符号。例如：Drip irrigation emitter; RP&M; Hydraulics; Labyrinth flow channel

**TYPE OF THESIS**: XXXXXXX

须与中文摘要中的论文类型一致；每个单词第一个字母大写，其余为小写。例如：Applied Research

论文类型包括： a.理论研究（Theoretical Research)；b.应用基础(Application Fundamentals)；c.应用研究(Application Research)；d.研究报告(Research Report)；e.设计报告(Design Report)；f.案例分析(Case Study)；g.调研报告(Investigation Report)；h.产品研发(Product Development)；i.工程设计(Engineering Design)；j.工程/项目管理(Engineering/Project Management)；k.其它（Others）。

# 

目 录

[1 绪论 1](#_Toc481703804)

[1.1 课题研究背景 1](#_Toc481703805)

[1.2 课题研究意义 1](#_Toc481703806)

[1.3 课题研究的主要内容与主要功能 2](#_Toc481703807)

[1.4 论文的组织结构 2](#_Toc481703808)

[2 相关理论 3](#_Toc481703809)

[2.1 流媒体技术原理 3](#_Toc481703810)

[2.2 TS流格式 3](#_Toc481703811)

[2.3 RTMP协议 5](#_Toc481703812)

[2.4 HLS协议 7](#_Toc481703813)

[2.5 FFmpeg 9](#_Toc481703814)

[3 流媒体服务器与回看录制系统的设计与实现 11](#_Toc481703815)

[3.1 系统总体设计 11](#_Toc481703816)

[3.2 直播流媒体服务器的实现 12](#_Toc481703817)

[3.2.1 开源流媒体服务器SRS 12](#_Toc481703818)

[3.2.2 SRS的部署与使用 12](#_Toc481703819)

[3.3 EPG同步服务器的实现 13](#_Toc481703820)

[3.3.1 EPG简介 13](#_Toc481703821)

[3.3.2 EPG同步服务器系统接口设计 14](#_Toc481703822)

[3.4 回看录制系统的设计与实现 16](#_Toc481703823)

[3.4.1 系统设计 16](#_Toc481703824)

[3.4.2 回看录制管理软件的设计与实现 16](#_Toc481703825)

[3.4.3 数据库设计 21](#_Toc481703826)

[3.4.4 回看录制模块的设计 22](#_Toc481703827)

[3.4.5 回看录制模块的实现 26](#_Toc481703828)

[4 客户端播放器的设计与实现 30](#_Toc481703829)

[4.1 概述 30](#_Toc481703830)

[4.2 网页播放器 30](#_Toc481703831)

[4.2.1 整体结构分析 30](#_Toc481703832)

[4.2.2 功能模块实现 30](#_Toc481703833)

[4.3 安卓手机与机顶盒播放器 32](#_Toc481703834)

[4.3.1 安卓系统组件 32](#_Toc481703835)

[4.3.2 整体结构实现 33](#_Toc481703836)

[4.3.3 功能模块实现 34](#_Toc481703837)

[5 系统测试与分析 40](#_Toc481703838)

[5.1 系统功能测试 40](#_Toc481703839)

[5.1.1 服务器系统功能测试 40](#_Toc481703840)

[5.2 服务器压力测试 41](#_Toc481703841)

[5.2.1 srs-bench 简介 41](#_Toc481703842)

[5.2.2 压力测试结果分析 41](#_Toc481703843)

[5.2.3 客户端播放器功能测试 41](#_Toc481703844)

[6 结论与展望 44](#_Toc481703845)

[6.1 工作总结 44](#_Toc481703846)

[6.2 后期展望 45](#_Toc481703847)

[致 谢 46](#_Toc481703848)

[参考文献 47](#_Toc481703849)

[攻读学位期间取得的研究成果 51](#_Toc481703850)

声明

CONTENTS

1 Preface X

1.1 Drip Irrigation Technology X

1.1.1 Drip Irrigation Systems X

2 Rapid Development of Labyrinth Drip Irrigation Emitters X

2.1 Structural Design of Labyrinth Drip Irrigation Emitters X

2.1.1 Theory X

2.6 Brief Summary X

12 Conclusions and Suggestions X

Acknowledgements X

References X

Appendices（单个附件用Appendix） X

Achievements X

Declarations

（这里的目录没办法自动生成，因为你没有相应的英文标题样式，只好你自己手工添加了，其实就是把中文目录翻译成英文就可以了）

编辑格式：“章节号＋英文标题＋Tab键1次＋页码”，编完以后，套用“CONTENTS”样式。

主要符号表

章的MathType的章标记（打印前将其字体颜色变为白色，在打印预览中看不见即可）：

# 绪论

## 课题研究背景

高清电视普及

编码压缩率

基于采集系统的完善

随着流媒体技术的高速发展

## 课题研究意义

近年来，随着互联网技术的高速发展，网络直播迅速发展成为一种新的互联网文化业态，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，越来越多的人接受这种传播形式，直播队伍逐步扩大，。

如今流媒体技术正处于高速发展的状态下，许多电视台都开展了流媒体直播业务。能够通过互联网观看电视直播节目，成为了一个越来越被人们重视的一个需求。本课题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，使用Linux系统搭建一个基于HLS协议和RTMP协议的流媒体服务器。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

论文共分为六章：

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；

第二章 相关理论介绍，主要介绍了TS流格式，RTMP和HLS流协议。

第三章 介绍服务器程序的设计与实现

第四章 介绍客户端播放器的设计与实现

第五章 系统的测试与分析

第六章 总结与展望

# 相关理论

## 流媒体技术原理

流媒体是指采用流式传输的方式在Internet播放的媒体格式，流媒体又叫流式媒体，一个实际采集端服务器将采集的视频分组成网络包，传送到网络上，另一端即客户端通过解压设备对这些数据进行解压组装后，将数据原样显示出来。网络音视频技术和移动通信技术的发展推动了流媒体技术的提升，给流媒体技术提供了技术保障，其中涉及到流媒体技术从数据的采集到播放整个过程。

流媒体传输技术室基于FTP/TCP协议，在此基础上发展而来的，故数据具有实时性，时序性，连续性的特点，使得流媒体数据要有严格的时序关系，流媒体数据将由服务器端分组成若干个代发的数据分组，然后按照特定的实时协议将分组依次在网络上进行传输，另一端的客户将接收到的分组按照实时协议的规格和视频格式重新组装起来，从而形成最后的数据文件。

流媒体技术能够实时传送并即刻播放的特性，即使在开始会有些许等待延迟，但仍然在实时性要求高的领域得到了广泛的应用。然而若在网络速率不稳定，网络环境复杂的情况下进行流媒体传输，视频将会出现停滞、断续的现象。

编码器、流媒体服务器以及客户端播放器这三个部分就能组成一个完整的流媒体控制系统。各个模块之间通过RTP/RTSP/TCP/HTTP/RTMP等协议进行通信数据交换，编码器的作用是将采集的数据通过视频编码技术，编码成网络能够识别的流媒体格式文件，便于在网络上传输，服务器则是负责接收和转发网络上传输的流媒体数据，客户端则是将编码的数据解码以便在客户端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，具有很好的容错能力。ts流最早应用于数字电视领域，其格式非常复杂包含的配置信息表多大十几个，视频格式主要是mpeg2。苹果公司发明的http live stream流媒体是基于ts文件的，不过他大大简化了传统的ts流，只需要2个最基本的配置表PAT和PMT，再加上音视频内容就可以了，hls流媒体视频编码的主要格式为h264/mpeg4，音频为aac/mp3。

ts文件分为三层：ts层（Transport Stream）、pes层（Packet Elemental Stream）、es层（Elementary Stream）。es层就是音视频数据，pes层是在音视频数据上加了时间戳等对数据帧的说明信息，ts层是在pes层上加入了数据流识别和传输的必要信息。

ts包大小固定为188字节，ts层分为三个部分：ts header、adaptation field、payload。ts header固定4个字节；adaptation field可能存在也可能不存在，主要作用是给不足188字节的数据做填充；payload是pes数据。

TS的首部内容包括：（1）占用一个字节的同步字节，固定为0x47。（2）占用1比特的传输错误指示符，表明在ts头的adapt域后有一个无用字节，通常都为0，这个字节算在adapt域长度内。（3）占用1比特的负载单元起始标识符，一个完整的数据包开始时标记为1。（4）占用1比特的传输优先级，0为低优先级，1为高优先级，通常取0。（5）占用13比特的pid值。（6）占用2比特的传输加扰控制，00表示位加密。（6）占用2比特的自适应区控制。（7）占用4比特的递增计数器，从0到F，起始值不一定取0，但必须是连续的。

ts层的内容是通过PID值来标识的，主要内容包括：PAT表、PMT表、音频流、视频流。解析ts流要先找到PAT表，只要找到PAT就可以找到PMT，然后就可以找到音视频流了。PAT表的PID值固定为0。PAT表和PMT表需要定期插入ts流，因为用户随时可能加入ts流，这个间隔比较小，通常每隔几个视频帧就要加入PAT和PMT。PAT和PMT表是必须的，还可以加入其它表如SDT（业务描述表）等，不过hls流只要有PAT和PMT就可以播放了。PAT表的作用就是指明了PMT表的PID值。PMT表的作用就是指明了音视频流的PID值。

自适应区内容包括自适应域长度、PCR即节目始终参考、填充字节。自适应区的长度要包含传输错误指示符标识的一个字节。pcr是节目时钟参考，pcr、dts、pts都是对同一个系统时钟的采样值，pcr是递增的，因此可以将其设置为dts值，音频数据不需要pcr。如果没有字段，ipad是可以播放的，但vlc无法播放。打包ts流时PAT和PMT表是没有adaptation field的，不够的长度直接补0xff即可。视频流和音频流都需要加adaptation field，通常加在一个帧的第一个ts包和最后一个ts包里，中间的ts包不加。

PES包的包头里面的存放着两个关键参数pts和dts，这两个参数是解决音视频同步显示，防止输入缓存上溢或下溢的关键。pts是显示时间戳、dts是解码时间戳，视频数据两种时间戳都需要，音频数据的pts和dts相同，所以只需要pts。有pts和dts两种时间戳是B帧引起的，I帧和P帧的pts等于dts。如果一个视频没有B帧，则pts永远和dts相同。从文件中顺序读取视频帧，取出的帧顺序和dts顺序相同。dts算法比较简单，初始值 + 增量即可，pts计算比较复杂，需要在dts的基础上加偏移量。

音频的pes中只有pts（同dts），视频的I、P帧两种时间戳都要有，视频B帧只要pts（同dts）。打包pts和dts就需要知道视频帧类型，但是通过容器格式我们是无法判断帧类型的，必须解析h.264内容才可以获取帧类型。

点播视频dts算法：

初始值可以随便指定，但是最好不要取0，video\_frame\_rate就是帧率，比如23、30。pts和dts是以timescale为单位的，1s = 90000 timescale , 一帧就应该是90000/video\_frame\_rate 个timescale。用一帧的timescale除以采样频率就可以转换为一帧的播放时。

点播音频dts算法：

audio\_samples\_per\_frame这个值与编解码相关，aac取值1024，mp3取值1158，audio\_sample\_rate是采样率，比如24000、41000。AAC一帧解码出来是每声道1024个sample，也就是说一帧的时长为1024/sample\_rate秒。所以每一帧时间戳依次0，1024/sample\_rate，...，1024\*n/sample\_rate秒。

直播视频的dts和pts应该直接用直播数据流中的时间，不应该按公式计算。

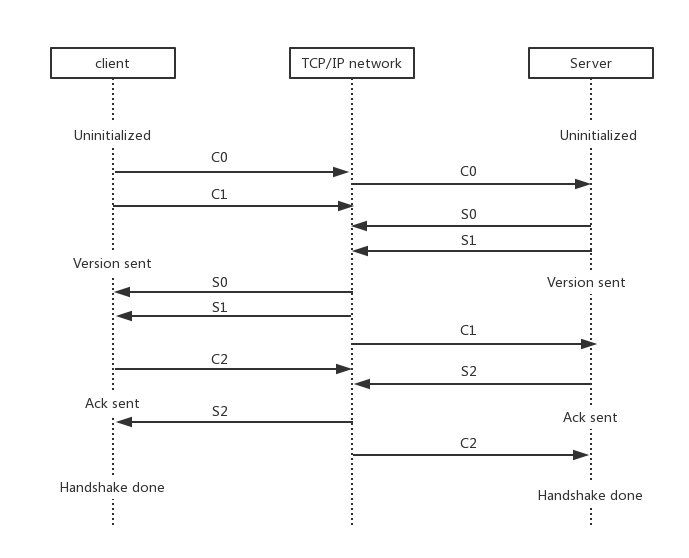
## RTMP协议

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协议，用来解决多媒体数据传输流的多路复用（Multiplexing）和分包（packetizing）的问题。随着VR技术的发展，视频直播等领域逐渐活跃起来，RTMP作为业内广泛使用的协议也重新被相关开发者重视起来。

RTMP协议是应用层协议，是要靠底层可靠的传输层协议（通常是TCP）来保证信息传输的可靠性的。在基于传输层协议的链接建立完成后，RTMP协议也要客户端和服务器通过“握手”来建立基于传输层链接之上的RTMP Connection链接，在Connection链接上会传输一些控制信息，如SetChunkSize,SetACKWindowSize。其中CreateStream命令会创建一个Stream链接，用于传输具体的音视频数据和控制这些信息传输的命令信息。RTMP协议传输时会对数据做自己的格式化，这种格式的消息我们称之为RTMP Message，而实际传输的时候为了更好地实现多路复用、分包和信息的公平性，发送端会把Message划分为带有Message ID的Chunk，每个Chunk可能是一个单独的Message，也可能是Message的一部分，在接受端会根据chunk中包含的data的长度，message id和message的长度把chunk还原成完整的Message，从而实现信息的收发。

要建立一个有效的RTMP Connection链接，首先要“握手”:客户端要向服务器发送C0,C1,C2（按序）三个chunk，服务器向客户端发送S0,S1,S2（按序）三个chunk，然后才能进行有效的信息传输。RTMP协议本身并没有规定这6个Message的具体传输顺序，但RTMP协议的实现者需要保证这几点：(1)客户端要等收到S1之后才能发送C2。（2）客户端要等收到S2之后才能发送其他信息（控制信息和真实音视频等数据）。（3）服务端要等到收到C0之后发送S1。（4）服务端必须等到收到C1之后才能发送S2。（5）服务端必须等到收到C2之后才能发送其他信息（控制信息和真实音视频等数据）

如果每次发送一个握手chunk的话握手顺序如图所示：



Chunk Stream是对传输RTMP Chunk的流的逻辑上的抽象，客户端和服务器之间有关RTMP的信息都在这个流上通信。这个流上的操作也是我们关注RTMP协议的重点。

RTMP的Message是指满足该协议格式的、可以切分成Chunk发送的消息，消息包含的字段如下：Timestamp（时间戳）：消息的时间戳，4个字节。Length(长度)：是指Message Payload（消息负载）即音视频等信息的数据的长度，3个字节TypeId(类型Id)：消息的类型Id，1个字节Message Stream ID（消息的流ID）：每个消息的唯一标识，划分成Chunk和还原Chunk为Message的时候都是根据这个ID来辨识是否是同一个消息的Chunk的，4个字节，并且以小端格式存储。TMP在收发数据的时候并不是以Message为单位的，而是把Message拆分成Chunk发送，而且必须在一个Chunk发送完成之后才能开始发送下一个Chunk。每个Chunk中带有MessageID代表属于哪个Message，接受端也会按照这个id来将chunk组装成Message。

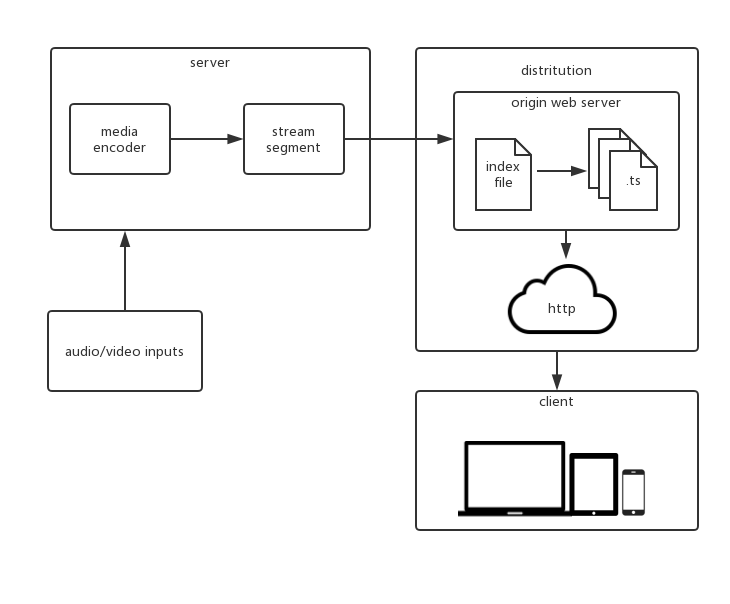
为什么RTMP要将Message拆分成不同的Chunk呢？通过拆分，数据量较大的Message可以被拆分成较小的“Message”，这样就可以避免优先级低的消息持续发送阻塞优先级高的数据，比如在视频的传输过程中，会包括视频帧，音频帧和RTMP控制信息，如果持续发送音频数据或者控制数据的话可能就会造成视频帧的阻塞，然后就会造成看视频时最烦人的卡顿现象。同时对于数据量较小的Message，可以通过对Chunk Header的字段来压缩信息，从而减少信息的传输量。（具体的压缩方式会在后面介绍）

Chunk的默认大小是128字节，在传输过程中，通过一个叫做Set Chunk Size的控制信息可以设置Chunk数据量的最大值，在发送端和接受端会各自维护一个Chunk Size，可以分别设置这个值来改变自己这一方发送的Chunk的最大大小。大一点的Chunk减少了计算每个chunk的时间从而减少了CPU的占用率，但是它会占用更多的时间在发送上，尤其是在低带宽的网络情况下，很可能会阻塞后面更重要信息的传输。小一点的Chunk可以减少这种阻塞问题，但小的Chunk会引入过多额外的信息（Chunk中的Header），少量多次的传输也可能会造成发送的间断导致不能充分利用高带宽的优势，因此并不适合在高比特率的流中传输。在实际发送时应对要发送的数据用不同的Chunk Size去尝试，通过抓包分析等手段得出合适的Chunk大小，并且在传输过程中可以根据当前的带宽信息和实际信息的大小动态调整Chunk的大小，从而尽量提高CPU的利用率并减少信息的阻塞机率。

## HLS协议

HTTP Live Streaming（缩写是HLS）是由苹果公司提出的一个基于HTTP协议的流媒体网络传输协议。是苹果公司QuickTime和iPhone软件系统的一部分。它的工作原理是把整个视频流切分成一个个小的基于HTTP的ts切片文件来下载，每次只下载一些。当客户端播放媒体流是，可以选择从索引文件中指定的不同的备用源中以不同的速率下载同样的资源，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协议架构图如图所示：



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（H264）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H264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视频流了。

使用HLS有诸多优点。首先，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，与实时传输协议（RTP)不同，HLS可以穿过任何允许HTTP数据通过的防火墙或者代理服务器。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。比如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。使用 HTTP 协议网络兼容性好, HTTP 数据包也可以方便地通过防火墙或者代理服务器，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。其实HLS最大的优势是他的跨平台兼容性。苹果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。这样一来，HLS成为了移动端最为兼容的视频播放协议。FLV，RTMP这些视频方案要想在移动设备上播放需要额外下点功夫。当然flash对移动设备造成很大的性能压力确实也是自身的问题。

## FFmpeg

FFmpeg 是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。它包括了目前领先的音/视频编码库 libavcodec。 FFmpeg 是在 Linux 下开发出来的，但它可以在包括 Windows 在内的大多数操作系统中编译。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、H263、H264等格式。FFmpeg可以轻易地实现多种视频格式之间的相互转换，例如可以将摄录下的视频 avi 等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：

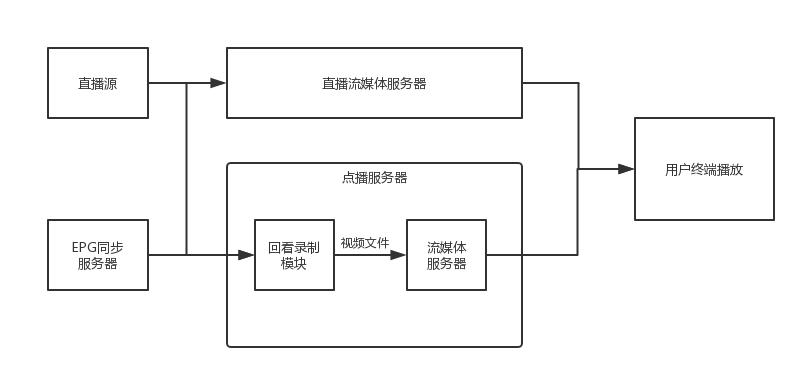
libavformat：用于各种音视频封装格式的生成和解析，包括获取解码所需信息以生成解码上下文结构和读取音视频帧等功能；音视频的格式解析协议，为 libavcodec 分析码流提供独立的音频或视频码流源。  
 libavcodec：用于各种类型声音/图像编解码；该库是音视频编解码核心，实现了市面上可见的绝大部分解码器的功能， libavcodec 库被其他各大解码器 ffdshow， Mplayer 等所包含或应用。  
 libavdevice：硬件采集、加速、显示。 操 作 计 算 机 中 常 用 的 音视频捕获或输出设备：ALSA,AUDIO\_BEOS,JACK,OSS,1394，VFW。  
 libavfilter:音视频滤波器的开发。  
 libavutil：包含一些公共的工具函数的使用库，包括算数运算 字符操作；  
 libavresample：音视频封转编解码格式预设等。  
 libswscale：（原始视频格式转换） 用于视频场景比例缩放、 色彩映射转换； 图像颜色空间或格式转换，如rgb565、rgb888等与yuv420等之间转换。  
 libswresample：原始音频格式转码  
 libpostproc：同步、时间计算的简单算法，用于后期效果处理；音视频应用的后处理，如图像的去块效应。  
 ffmpeg：该项目提供的一个工具，可用于格式转换、解码或电视卡即时编码等；  
 ffsever：一个 HTTP 多媒体即时广播串流服务器；  
 ffplay：是一个简单的播放器，使用ffmpeg库解码，通过SDL显示。

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。

# 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

整体系统框图如图所示。



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H264。

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

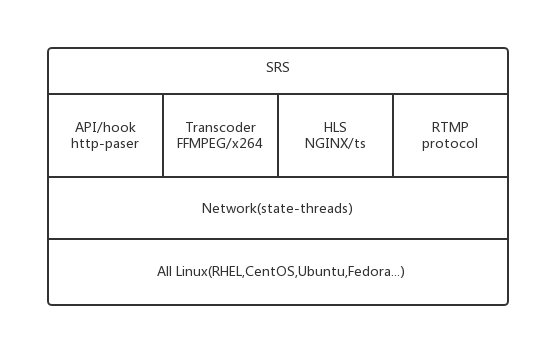
SRS定位是运营级的互联网直播服务器集群，追求更好的概念完整性和最简单实现的代码。SRS提供了丰富的接入方案将RTMP流接入SRS，包括推送RTMP到SRS、拉取流到SRS。SRS还支持将接入的RTMP流进行各种变换，譬如直播流转码、转发给其他服务器、转封装成HLS、录制成FLV。SRS包含支大规模集群如CDN业务的关键特性，譬如RTMP多级集群、VHOST虚拟服务器、无中断服务Reload。此外，SRS还提供丰富的应用接口，包括HTTP回调、HTTP API接口、RTMP测速。SRS在源站和CDN集群中都得到了广泛的应用。

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的系统架构如图所示：



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，意思是电子节目指南。 IPTV所提供的各种业务的索引及导航都是通过 EPG系统来完成的。 IPTV EPG实际上就是 IPTV的一个门户系统。 EPG系统的界面与 Web页面类似，在 EPG界面上一般都提供各类菜单、按钮、链接等可供用户选择节目时直接点击的组件； EPG的界面上也可以包含各类供用户浏览的动态或静态的多媒体内容。

EPG的主要作用就是用户利用EPG提供的菜单，可以选择自己喜欢的组播频道；点播自己喜欢的视频节目；在线演唱自己喜欢的歌曲；查找IPTV提供的各种信息，包括生活信息、娱乐信息、教育信息、体育信息等等；用户也可使用EPG提供的菜单来订购自己喜欢的节目；甚至可以通过EPG提供的菜单支付水费、电费、进行电子商务的交易等。用户还可以利用EPG菜单查看节目的附加信息，例如对节目内容介绍、演员及导演的介绍等。同时通过EPG菜单中提供的家长控制功能，家长可以对某些节目加以限制，不给孩子开放所有的观看权限。

EPG为IPTV提供的基本业务(如VOD点播/KTV/歌曲)及各种增值业务的使用提供了简单方便的操作平台，为IPTV用户收看电视节目、享受多媒体节目点播以及开展信息服务提供了一个良好的导航机制。使用EPG系统可使用户能够方便快捷地找到自己关心的节目。使用EPG系统，用户通过电视机这个终端和IP机顶盒就可以登陆Internet。更重要的是使用EPG系统用户就可以和电视进行互动，这样用户就不再被动地接收信息，用户可以及时、主动地发表自己的意见和看法，并将这些意见和看法及时反馈给内容制作商。因此EPG在IPTV系统中起着十分重要的作用。

### 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。EPG同步服务器的系统框图如图所示：

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：

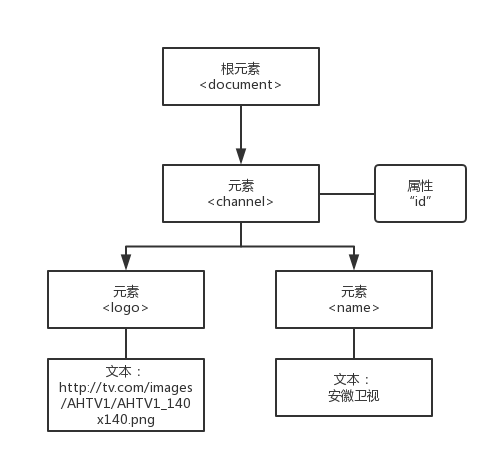
http://tv.com/EPG/channel?secret=XXXX

节目单接口（两周以内全量节目单）：

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

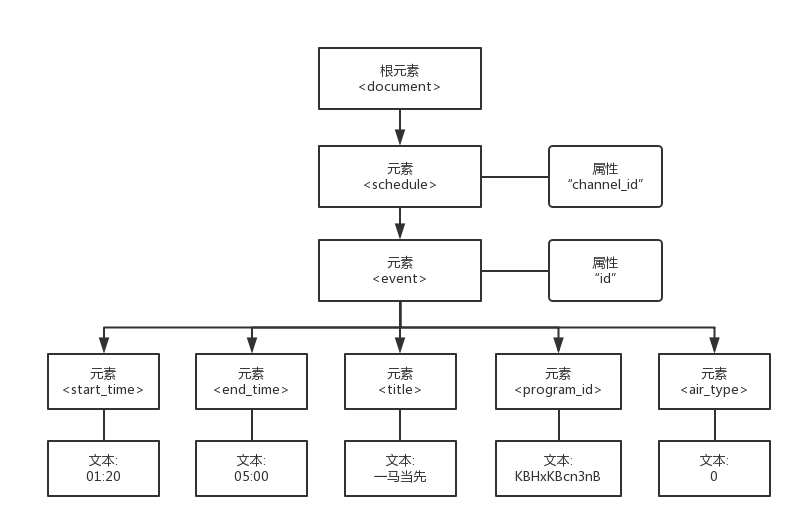
其中secret参数为唯一认证秘钥，客户调用所有接口都需要传入此参数来标识客户的身份。id参数为需要获得频道信息的频道id。客户首先需调用接口得到所有EPG授权给该客户的频道，然后将获得的频道id传入接口。接着得到该频道两周内（不足两周的有多少给多少）的节目单：节目单中若related\_teams节点不为空（1-2个id），则取其id传入接口。

授权频道的xml结构图如图所示：



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结构图如图所示：

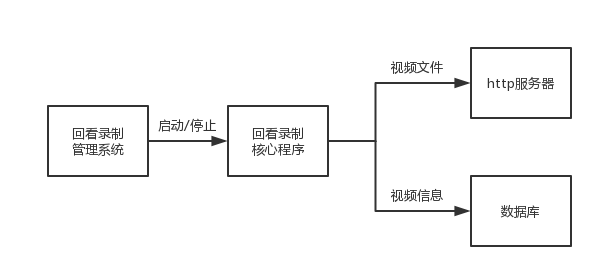


根节点document下包含一个schedule元素，属性channel\_id表示该schedule元素的频道id。schedule元素下包含若干event元素，每个event元素使用属性id唯一标示，标示一个电视节目的信息，数据库中节目的增删均根据该id进行。event元素下的start\_time标签和end\_time标签分别存储该节目的起始时间与结束时间，title标签存储该节目的节目名称，program\_id为标示该节目的唯一id，air\_type为该节目的播放类型（仅支持综艺栏目），其中序号0代表位置，序号1代表直播，序号2代表首播，序号3代表重播。

## 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。系统流程图如图所示：



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）系统管理员可以添家直播频道的信息。（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。

系统用例图如图所示：



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。从这些对象翅中筛选，如果有与该对象相关的身份和行为，那么久应该为之创建类。

（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应该具有下列私有属性：

//TODO

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。界面如图所示：



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。界面如图所示：



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所示：



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所示：



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下图所示。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。



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。<br>

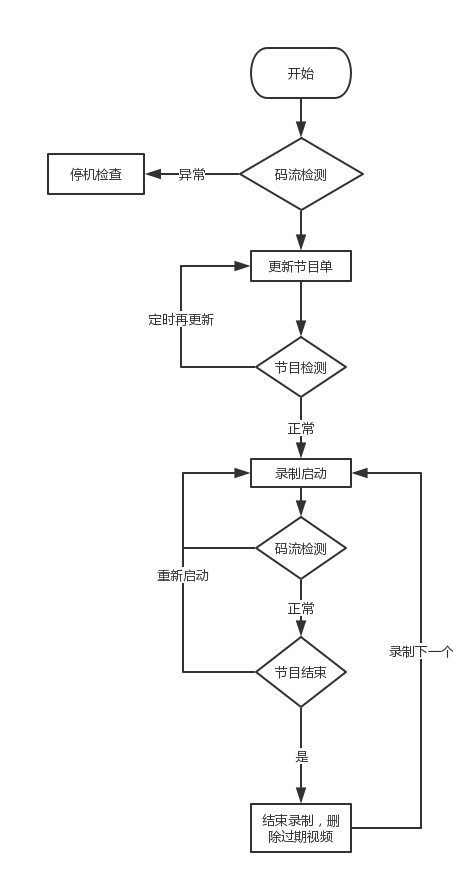
channel表与program表为一对多的关系。channel表与program表之间的一对多关系通过在表program中插入外检“channel\_id”以匹配channel表中的“channel\_id”来模拟。

### 回看录制模块的设计

在图xxx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。流程图如图所示，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，最后讲生成的回看视频信息写入数据库中一遍用户进行观看。

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，一共实现过过以下三个方案:

1. 将rtmp直播流录制成flv文件存储在本地http服务器。流程图如图所示：



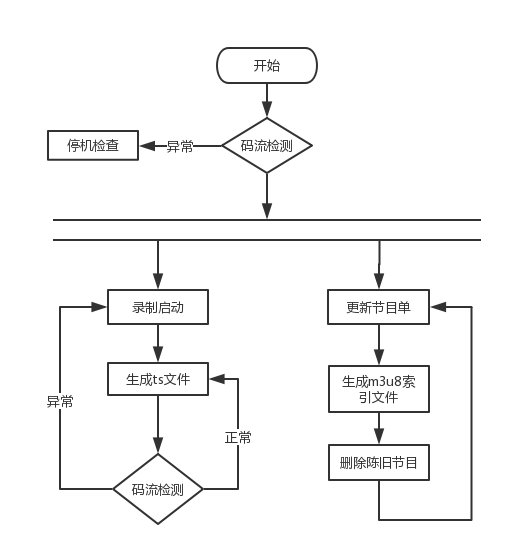
初次启动录制时在命令行中输入脚本名后加入channel\_id作为运行参数，以达到录制相应频道节目的目的。创建文件目录和程序运行日志，目录存在则跳过，日志如果存在则从最后一行开始记录。根据channel\_id从channel表中获取RTMP码流地址并测试码流是否获取正常。如果测试失败则将数据库中运行状态和主进程号写0并终止程序，如果测试通过则将主进程号和运行状态1写入数据库并将进行下一步操作，从节目单接口获取跟channel\_id相对应的XML格式节目数据，然后将其中有用的部分进行解析和处理后保存在数据库中。然后根据channel\_id和当前系统时间，从刚刚保存在数据库的program表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起变化的问题，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72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：录制脚本rtmpdump.py、数据库管理脚本db\_manager.py及关闭脚本kill.py。软件整体框架

2. 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，流程图如图所示。



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。（2）删除本地文件系统中陈旧的视频节目。

3. 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1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。而且flv文件无法再苹果公司的ios平台浏览其中进行播放，影响了系统的兼容性。方法2将视频格式由flv编程了ts格式，解决了兼容性问题，但是由于其录制时会产生累计误差，导致录制时间与系统时间偏移较大。因此最终选用方案3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，statu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

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)

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：

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xxx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()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他切片成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。

# 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.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：

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();

}

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。

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”，是指一种创建交互式网页应用的网页开发技术。通过在后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。这意味着可以在不加载整个网页的情况下对网页的某分进行更新。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。

## 安卓手机与机顶盒播放器

### 安卓系统组件

Android 是基于 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。底层 Linux 内核仅提供基本功能，其他系统中的应用软件则开发者或厂商自行开发，其中部分代码以 Java 编写。如图 2.1 所示，Android 系统的架构由底层到应用层主要由 4 个层次 5 个部分组成，分别为 Linux 内核层(Linux Kernel)、系统类库(Libraries)和运行库层(Android Runtime)、应用程序框架层(Application Frameworks)

和应用程序层(Applications)。Android四大基本组件分别是Activity，Service服务,Content Provider内容提供者，BroadcastReceiver广播接收器。

1. Android 系统中最常用、使用最频繁的基本组件就是 Activity 组件，在一个 Android应用中，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。不同的Activity 被系统视为不同的类，这些不同类的父类都继承自 Activity 这个基类。这个Activity 类就会显现出由多个 Views 控件组成的用户自定义的接口，并且针对该接口的事件作出对应的响应。因为一个程序对应多个用户定义的 Activity，所以多数的应用程序都会包含多于一个的 Activity。例如，联系人列表是一个 Activity，查看联系人详细信息是里另一个 Activity，给其中一个联系人发短信息又是另外一个 Activity。尽管一个整体的用户界面是由多个 Activity 组成，但是每个 Activity 都是独立于其他的 Activity 而存在的。

2. Service通常情况下 Service 是一个没有可视化用户界面但拥有长生命周期的应用程序，这个程序不需要与用户进行交互，它是无时间限制运行在系统的后台任务，与 Unix 进行相类似。每个服务都继承于 Service 基类，通常是执行一些需要持续运行的情况。它负责触发通知和某些可视的 Activity 和数据源。它可以绑定其他正在持续运行的服务，若该服务没有运行，则启动该服务。连接后，它可以通过那个服务的接口与服务进行通讯。Service 服务在应用程序中的主线程内运行，它和 Activity 组件等运行情况相同，所以Service 会派生新的线程来进行耗时任务，这些任务不会对其它应用界面和组件产生任何阻塞或干扰。

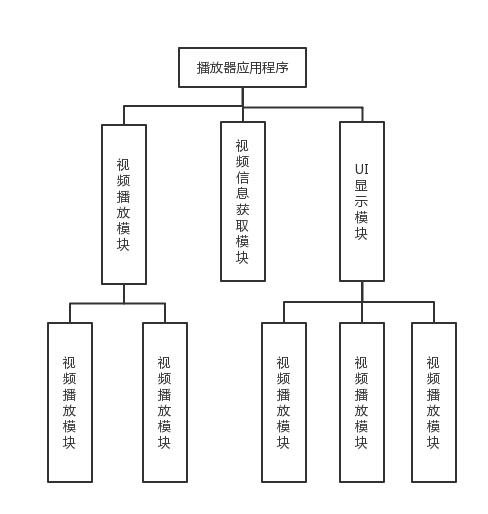
3. Broadcast Receiver 和 Service 一样，不会显示有 UI 元素的图形界面，它是为了实现广播而提供的一种组件，广播来自系统或应用程序，它负责接收广播并做出相应的动作亦接收请求并处理 Intent 。通常 Service 和 Broadcast Receiver 都会在AndroidManifest.xml 文件中注册。

4. Content Provider 组件主要用于 Android 应用程序的数据存储管理，为了能够在应用程序间共享数据。一般情况它起到扮演服务器的角色，通过它可以对数据存储进行访问。通过 Content Provider 中的读写功能来传递对数据进行存储和更新等操作，但是应用程序是调用一个 Content Resolver 对象的方法来实现访问数据的。Content Resolver 与Content Provider 相对应，每一个 Content Resolver 能够与任何一个 Content Provider 通信

### 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，安卓手机客户端播放器效果图如图所示：

该应用程序整体机构设计如图所示如图所示：

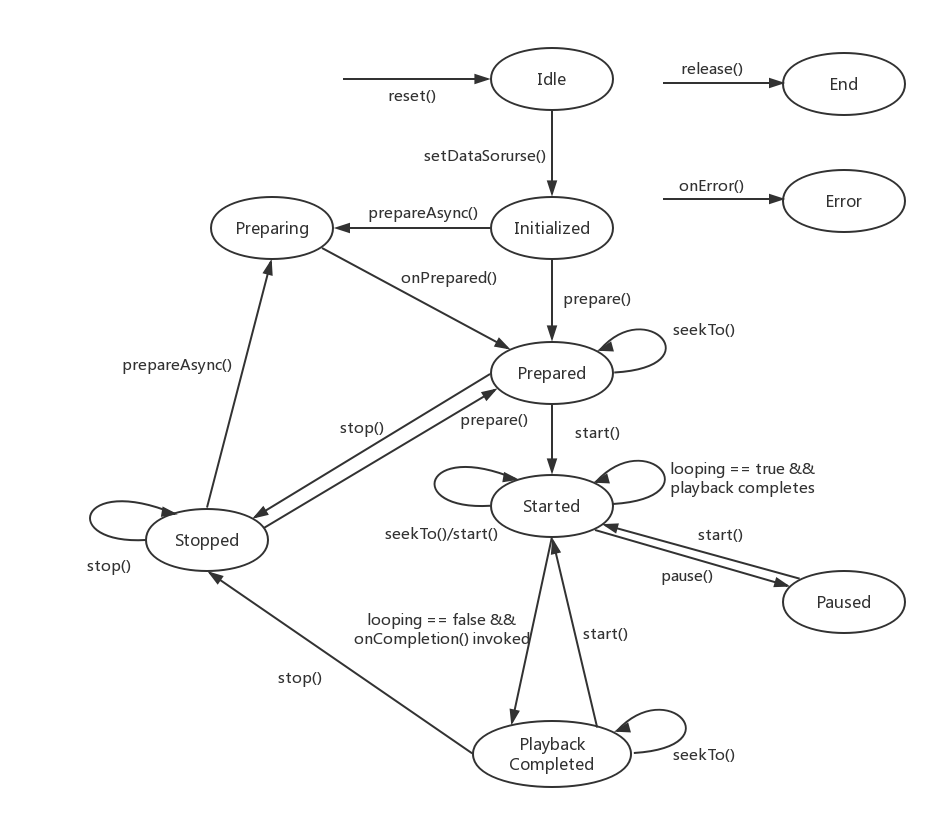


共分为三大模块，分别为视频播放模块、视频信息获取模块以及ui界面模块。其中，视频播放模块包括播放显示模块和播放控制器模块，播放显示模块负责将视频从网络上获取下来进行解码并显示在屏幕上。播放控制模块则负责显示当前视频节目的播放进度并控制其跳转与快进快退。视频信息获取模块负责从管理服务器提供的数据接口中提取相应的视频信息，传送给ui界面模块。ui界面模块负责显示节目信息以及控制视频节目的播放。

### 功能模块实现

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的状态转换图如图所示。



这张状态转换图清晰的描述了MediaPlayer的各个状态，也列举了主要的方法的调用时序，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：当使用new()方法创建一个MediaPlayer对象或者调用了其reset()方法时，该MediaPlayer对象处于idle状态。这两种方法的一个重要差别就是：如果在这个状态下调用了getDuration()等方法（相当于调用时机不正确），通过reset()方法进入idle状态的话会触发OnErrorListener.onError()，并且MediaPlayer会进入Error状态；如果是新创建的MediaPlayer对象，则并不会触发onError(),也不会进入Error状态。

End 状态：通过release()方法可以进入End状态，只要MediaPlayer对象不再被使用，就应当尽快将其通过release()方法释放掉，以释放相关的软硬件组件资源，这其中有些资源是只有一份的（相当于临界资源）。如果MediaPlayer对象进入了End状态，则不会在进入任何其他状态了。

Initialized 状态：这个状态比较简单，MediaPlayer调用setDataSource()方法就进入Initialized状态，表示此时要播放的文件已经设置好了。

Prepared 状态：初始化完成之后还需要通过调用prepare()或prepareAsync()方法，这两个方法一个是同步的一个是异步的，只有进入Prepared状态，才表明MediaPlayer到目前为止都没有错误，可以进行文件播放。

Preparing 状态：这个状态比较好理解，主要是和prepareAsync()配合，如果异步准备完成，会触发OnPreparedListener.onPrepared()，进而进入Prepared状态。

Started 状态：显然，MediaPlayer一旦准备好，就可以调用start()方法，这样MediaPlayer就处于Started状态，这表明MediaPlayer正在播放文件过程中。可以使用isPlaying()测试MediaPlayer是否处于了Started状态。如果播放完毕，而又设置了循环播放，则MediaPlayer仍然会处于Started状态，类似的，如果在该状态下MediaPlayer调用了seekTo()或者start()方法均可以让MediaPlayer停留在Started状态。

Paused 状态：Started状态下MediaPlayer调用pause()方法可以暂停MediaPlayer，从而进入Paused状态，MediaPlayer暂停后再次调用start()则可以继续MediaPlayer的播放，转到Started状态，暂停状态时可以调用seekTo()方法，这是不会改变状态的。

Stop 状态：Started或者Paused状态下均可调用stop()停止MediaPlayer，而处于Stop状态的MediaPlayer要想重新播放，需要通过prepareAsync()和prepare()回到先前的Prepared状态重新开始才可以。

PlaybackCompleted状态：文件正常播放完毕，而又没有设置循环播放的话就进入该状态，并会触发OnCompletionListener的onCompletion()方法。此时可以调用start()方法重新从头播放文件，也可以stop()停止MediaPlayer，或者也可以seekTo()来重新定位播放位置。

Error状态如果由于某种原因MediaPlayer出现了错误，会触发OnErrorListener.onError()事件，此时MediaPlayer即进入Error状态，及时捕捉并妥善处理这些错误是很重要的，可以帮助我们及时释放相关的软硬件资源，也可以改善用户体验。通过setOnErrorListener(android.media.MediaPlayer.OnErrorListener)可以设置该监听器。如果MediaPlayer进入了Error状态，可以通过调用reset()来恢复，使得MediaPlayer重新返回到Idle状态。

MediaPlayer底层调用开源视频编解码软件ffmpeg进行视频解码。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。首先，创建一个数据读取线程。 函数如下所示：

static VideoState \*stream\_open(FFPlayer \*ffp, const char \*filename, AVInputFormat \*iformat)

{

frame\_queue\_init(&is->pictq, &is->videoq, ffp->pictq\_size, 1)

frame\_queue\_init(&is->sampq, &is->audioq, SAMPLE\_QUEUE\_SIZE, 1)

packet\_queue\_init(&is->videoq);

packet\_queue\_init(&is->audioq);

SDL\_CreateThreadEx(&is->\_video\_refresh\_tid, video\_refresh\_thread, ffp, "ff\_vout") //视频显示线程创建

SDL\_CreateThreadEx(&is->\_read\_tid, read\_thread, ffp, "ff\_read")

}

前面几句都是初始化队列的操作，分为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。

后面创建了两个线程，video\_refresh\_thread为视音频同步显示线程，read\_thread则为数据读取线程。读取线程部分代码如下：

static int read\_thread(void \*arg)

{

err = avformat\_open\_input(&ic, is->filename, is->iformat, &ffp->format\_opts);

err = avformat\_find\_stream\_info(ic, opts);

/\* open the streams \*/

if (st\_index[AVMEDIA\_TYPE\_AUDIO] >= 0) {

stream\_component\_open(ffp, st\_index[AVMEDIA\_TYPE\_AUDIO]);

}

ret = -1;

if (st\_index[AVMEDIA\_TYPE\_VIDEO] >= 0) {

ret = stream\_component\_open(ffp, st\_index[AVMEDIA\_TYPE\_VIDEO]);

}

for (;;) {

if (is->seek\_req) {

avformat\_seek\_file();

}

ret = av\_read\_frame(ic, pkt);

packet\_queue\_put(&is->audioq, pkt);

//如果是视频的话

//packet\_queue\_put(&is->videoq, pkt);

}

}

首先如果让我们来实现一个读取线程，我们是不是要先判断视频源的格式？？没错，avformat\_open\_input（）/Utils.c这句话内部就是探测数据源的格式。这里会跳转到ffmpeg里面了，不在本文讲解的范围内，有兴趣的童鞋可以自行阅读下源码，其实在avformat\_open\_input（）/Utils.c内读取网络数据包头信息时调用id3v2\_parse()，然后获取到头信息后会执行ff\_id3v2\_parse\_apic()/id3v2.c。最后 会更改ic这个AVFormatContext型的参数。

avformat\_find\_stream\_info()解析流并找到相应解码器。当然解码器在我们上一文初始化的时候注册过了，在哪里呢？

在JNI\_Load()函数里面有一个ijkmp\_global\_init()，然后它会调用avcodec\_register\_all();这个函数其实就是注册解码器等。

然后对于音频或者视频都会调用stream\_component\_open()函数来进行音视频读取和解码。在stream\_component\_open函数中，分别创建视频和音频的解码线程video\_thread与audio\_thread。在video\_thread与audio\_thread，分别调用decoder\_decode\_frame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。

2. 视频信息获取模块

该模块使用异步任务AsyncTask实现。当一个程序第一次启动时，Android会同时启动一个对应的主线程（Main Thread），主线程主要负责处理与UI相关的事件，如：用户的按键事件，用户接触屏幕的事件以及屏幕绘图事件，并把相关的事件分发到对应的组件进行处理。所以主线程通常又被叫做UI线程。在开发Android应用时必须遵守单线程模型的原则： Android UI操作并不是线程安全的并且这些操作必须在UI线程中执行。如果在非UI线程中直接操作UI线程，会抛出异常。这与普通的Java程序不同。由于UI线程负责事件的监听和绘图，因此，必须保证UI线程能够随时响应用户的需求，UI线程里的操作应该向中断事件那样短小，费时的操作（如网络连接）需要另开线程，否则，如果UI线程超过5s没有响应用户请求，会弹出对话框提醒用户终止应用程序。

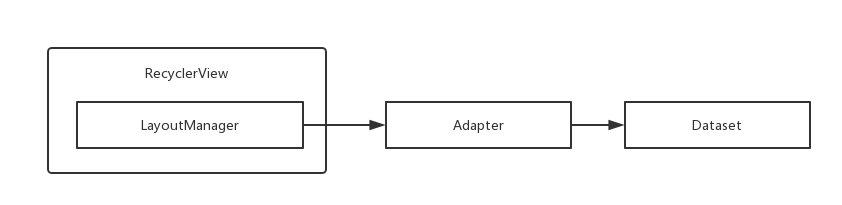
为了不再主线程中执行耗时的信息获取操作，我们使用安卓提供的AsyncTask异步任务异步地获取信息，程序结束之后再讲结果返回给主线程。使用异步任务获取到直播与点播的视频信息之后，需要把信息存入对应的数据结构之中。

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。ListView的基本结构徐图所示:



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。

# 系统测试与分析

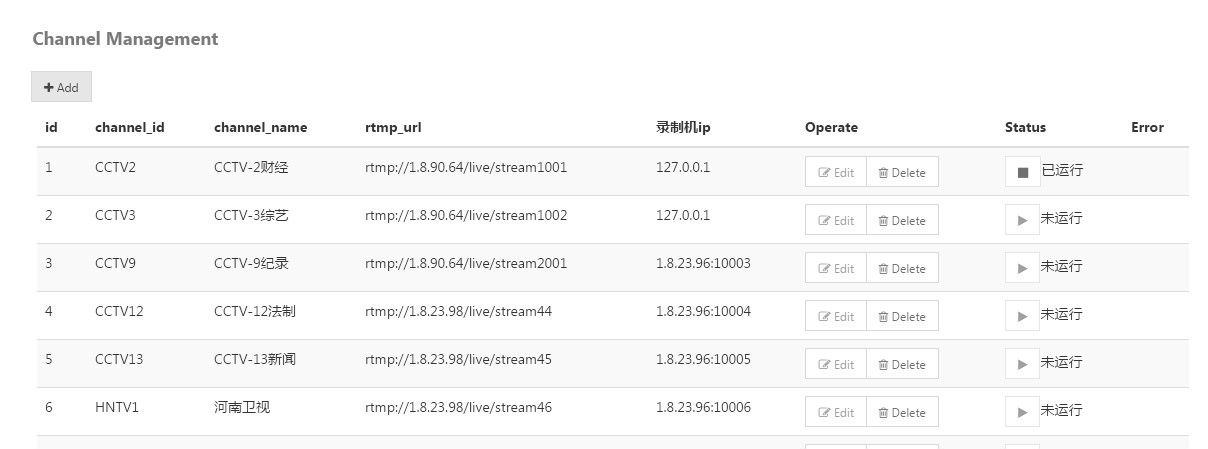
## 系统功能测试

系统测试的目标是以最少的时间和人力找出软件中潜在的各种错误和缺陷。如对系统实施了严格的规范测试,就能够发现其中大部分的错误。系统测试能够确认系统实现的功 能和性能与需求说明的一致性。系统测试还能收集到足够的测试结果为系统可靠性提供依据。 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。本系统我们进行了长期的测试,邀请多人次、多时段对本系统进行测试,虽然在其中 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所示：



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。效果如图所示：

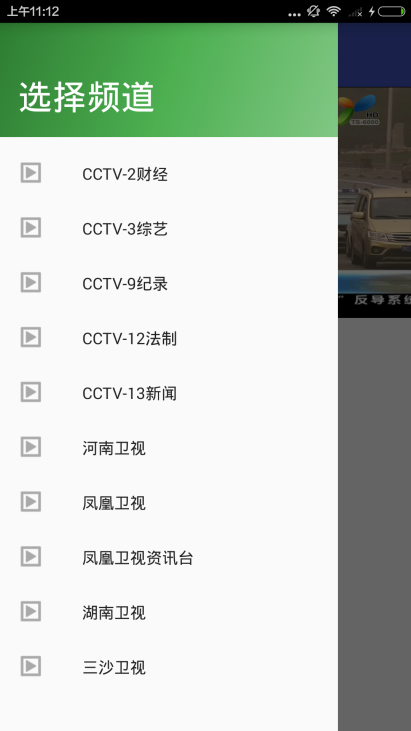
  

图（a）为播放主界面，点击直播标签切换至直播页面，点击回看标签则切换至回看页面。图（b）为回看节目列表，列出当前日期的所有回看节目信息。图（c）为频道列表，列出当前所有可选的电视频道，用户点击可以切换不同的频道进行观看。

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

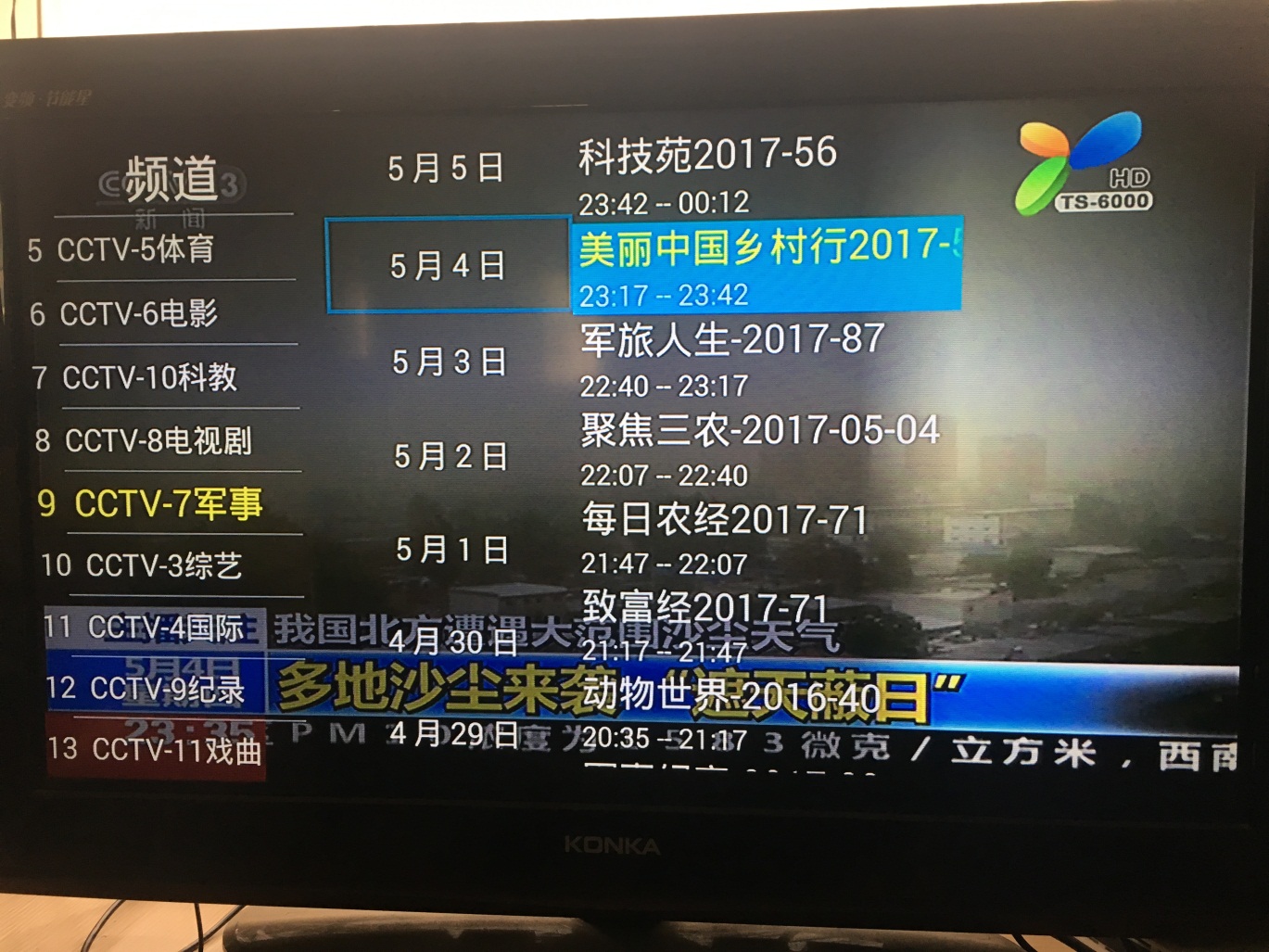


使用安卓手机安装本课题开发的apk应用程序，指定好管理服务器的ip地址。应用程序界面如图所示：

图（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（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（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。应用程序界面如图所示：



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。

# 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。从广义上讲，网络直播吸取和延续了互联网的优势，利用视讯方式进行网上现场直播，可以将产品展示、相关会议、背景介绍、方案评测、网上调查、对话访谈、在线培训等内容现场发布到互联网上，利用互联网的直观、快速、表现形式好、内容丰富、交互性强、地域不受限制、受众可划分等特点，加强活动现场的推广效果。现场直播完成后，还可以为读者继续提供重播、点播，有效延长了直播的时间和空间，发挥直播内容的最大价值。

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网络用户观看。

（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）介绍了，RTMP与HLS的协议架构，并详细比较的两种协议各自的优缺点。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，我们放弃了使用RTMP和HLS作为直播源的传输协议，而是使用UDP传输TS流作为直播源。

（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。忘读者批评指正。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，是你们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

（此上两空行不能删除，是为EndNote的参考文献列表所预留）

文后著录的参考文献务必实事求是。论文中引用过的文献必须著录，未引用的文献不得出现。应遵循学术道德规范，避免涉嫌抄袭、剽窃等学术不端行为。

参考文献一般应是作者亲自考察过的对学位论文有参考价值的文献，除特殊情况外，一般不应间接引用。

参考文献应有权威性，要注意引用最新的文献。

参考文献的数量：

硕士学位论文，一般应在30篇以上，其中，期刊文献不少于20篇，国外文献不少于10篇，均以近5年的文献为主。

对于申请专业学位的学位论文，参考文献的数量可参照执行。

参考文献的著录格式应符合国家标准GB/T 7714-2005《文后参考文献著录规则》。参考文献中每条项目应齐全。

文献中的作者不超过三位时全部列出，超过三位时，一般只列前三位，中文的后面加 “等”字，英文的后面加 “et al”，作者姓名之间用逗号分开。

外国人名一般采用姓在前，名在后的著录法，姓全写且第一个字母大写，名简写成单个大写字母且不加标点，姓和名之间空1格，如：“Metcalf SW”。也可采用名在前，姓在后的著录法，姓全写且第一个字母大写，名简写成单个大写字母且不加标点，名和姓之间空1格，如：“SW Metcalf”。

中文人名的英文表达方式：

简写时，采用姓在前，名在后的著录法，姓全写且第一个字母大写，名简写成单个大写字母且不加标点，如，“钱学森”，简写为“Qian XS ”。

全拼时，名在前，姓在后的著录法，名的第一个字母大写，名连写，名后空1格写姓，姓的第一个字母大写。如，“钱学森”，写为“Xuesen Qian”。

文后参考文献著录格式范例样板，采用五号。

具体要求如下：

A 专著（包括普通图书［M］、论文集和会议录［C］、科技报告［R］、学位论文［D］、标准［S］）

主要责任者．文献题名［文献类型标志］．其他责任者．版本项(第１版不标注) ．出版地：出版者，出版年：引文页码．获取和访问路径．

B 专著中的析出文献

析出文献主要责任者．析出文献题名[文献类型标志]．析出文献其他责任者//专著主要责任者．专著题名：其他题名信息. 版本项(第１版不标注) ．出版地：出版者，出版年：析出文献的起止页码．获取和访问路径．

C连续出版物

主要责任者．题名:其他题名信息［文献类型标志］．年，卷（期）－年，卷（期）.出版地：出版者，出版年．获取和访问路径．

D连续出版物中的析出文献（包括期刊中析出的文献[J]、报纸中析出的文献[N].）

析出文献主要责任者．析出文献题名［文献类型标志］．连续出版物题名：其他题名信息，年，卷（期）：页码．获取和访问路径．

E专利文献

专利发明者/专利申请者或所有者．专利题名: 专利国别,专利号［文献类型标志］.公告日期或公开日期. 获取和访问路径．

F电子文献（包括专著或连续出版物中析出的电子文献）

主要责任者．题名：其他题名信息[文献类型标志/载体类型标志]．出版地：出版者，出版年（更新或修改日期）．获取和访问路径．

表2-2 文献类型和标志代码

|  |  |  |  |
| --- | --- | --- | --- |
| 文献类型 | 标志代码 | 文献类型 | 标志代码 |
| 普通图书 | M | 会议录 | C |
| 汇编 | G | 报纸 | N |
| 期刊 | J | 学位论文 | D |
| 报告 | R | 标准 | S |
| 专利 | P | 数据库 | DB |
| 计算机程序 | CP | 电子公告 | EB |

表2-3 电子文献载体和标志代码

|  |  |  |  |
| --- | --- | --- | --- |
| 载体类型 | 标志代码 | 载体类型 | 标志代码 |
| 磁带（magnetic tape） | MT | 磁盘（disk） | DK |
| 光盘（CD-ROM） | CD | 联机网络（online） | OL |

样例：

1. 刘国钧,郑如斯．中国书的故事［M］．北京：中国青年出版社，1979：110-115．
2. 昂温 G． 外国出版史[M].陈生铮译．北京：中国书籍出版社，1988．
3. 辛希孟．信息技术与信息服务国际研讨会论文集：A集［C］．北京:中国社会科学出版社,1979．
4. 冯西桥．核反应堆压力容器的LBB分析［R］．北京：核能技术设计研究院，1997．
5. 张和生．地质力学系统理论［D］．太原：太原理工大学，1998．
6. 全国文献工作标准化技术委员会第七分委员会．GB/T 5795-1986．中国标准书号［S］．北京:中国标准出版社，1986．
7. 罗云．安全科学理论体系的发展及趋势探讨[M]//白春华，何学秋，吴宗之．21世纪安全科学与技术的发展趋势．北京：科学出版社，2000：1-5．
8. 钟文发．非线性规划在可燃毒物配置中的应用[C]//赵玮．运筹学的理论与应用：中国运筹学会第五届大会论文集．西安：西安电子科技大学出版社，1996：468－471．
9. 高义民，张凤华，邢建东等．颗粒增强不锈钢基复合材料冲蚀磨损性能研究[J]． 西安交通大学学报，2001，35(7)：727-730．
10. Papworth A, Fox P, Zeng GT, et al. Ability of aluminum alloy to wet alumina fibres by addition of bismuth[J]. Mater Sci & Technol,1999,15(4):419-428.
11. 丁文祥．数字革命与竞争国际化[N]．中国青年报，2000－11－20(15)．
12. 姜锡洲．3一种温热外敷药制备方案：中国，881056078［P］．1989-07-26．
13. Koseki A,Momose H,Kawahito M,et al Complier:US,828402［P/OL］2002-05-25 [2002-05-28].http://FF&p.
14. Online Computer Library Center, Inc. History of OCLC[EB/OL].[2000-01-08]. http://www. clc.org/ about/history/default.htm.
15. 江向东．互联网环境下的信息处理与图书管理系统解决方案[J/OL]．情报学报,1999,18(2):4[2000-01-18].http://www.chinainfo.gov.cn/periodical/qbxb．
16. Scitor C. Project scheduler[CP/DK].Sunnyvale,Calif.:Scitor Corp, 1983.
17. Metcalf SW. The Tort Hall air emission study[C/OL]//The International Congress on Hazardous Waste, Marquis Hotel, Atlanta,Georgia,June 5-8,1995: impact on human and ecological health[1998-09-22]. <http://atsdrl>.atsdr.cdc.gov:8080/cong95. html.

参考文献里面标点符号：英文文献用半角,中文文献用全角。

# 攻读学位期间取得的研究成果

1）已发表或已录用的学术论文、已出版的专著/译著、已获授权的专利按参考文献格式列出。

2）科研获奖，列出格式为：

获奖人（排名情况）．项目名称．奖项名称及等级，发奖机构，获奖时间．

3）与学位论文相关的其它成果参照参考文献格式列出。

4）全部研究成果连续编号编排。

样例：

1. Wei ZY, Tang YP, Zhao WH, et al. Rapid development technique for drip irrigation emitters[J]. RP Journal,UK., 2003, 9(2):104~110 (SCI: 672CZ; EI: 03187452127).
2. 魏正英,唐一平,卢秉恒．滴灌管内嵌管状滴头的快速制造方法研究[J]．农业工程学报, 2001,17(2):55~58 (EI:01226526279,01416684777)．

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)