分类号_____ 密级_____

UDC_____ 编号_____

華中師範大學

硕士学位论文

基于 FFMpeg 的稳定应用层组播流媒体直播系统研究

学位申请人姓名：代文姣

申请学位学生类别：全日制硕士

申请学位学科专业：计算机技术

指导教师姓名：崔建群 教授

硕士学位论文

基于 FFMpeg 的稳定应用层组播流媒体直播系统研究

论文作者： 代文姣

指导教师： 崔建群教授

学科专业： 计算机技术

研究方向： 计算机网络

华中师范大学计算机学院

2018 年 4 月

Research on a Stable Application Layer Multicast Media Streaming System Based on FFMpeg

A Thesis

Submitted in Partial Fulfillment of the Requirement

For the M.S Degree in Computer Technology

By

Wenjiao Dai

Postgraduate Program

School of Computer

Central China Normal University

Supervisor: Jianqun Cui

Academic Title: Professor

Signature Jianqun Cui

Approved

April, 2018

华中师范大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：代文姣　　日期：2018 年 5 月 29 日

学位论文版权使用授权书

学位论文作者完全了解华中师范大学有关保留、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属华中师范大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密，在年解密后适用本授权书。

非保密论文注释：本学位论文不属于保密范围，适用本授权书。

作者签名：代文姣　　导师签名：崔建群

日期：2018 年 5 月 29 日　　日期：2018 年 5 月 29 日

本人已经认真阅读“CALIS 高校学位论文全文数据库发布章程”,同意将本人的学位论文提交“CALIS 高校学位论文全文数据库”中全文发布,并可按“章程”中的规定享受相关权益。同意论文提交后滞后：□半年；□一年；□二年发布。

作者签名：代文姣　　导师签名：崔建群

日期：2018 年 5 月 29 日　　日期：2018 年 5 月 29 日

摘 要

组播技术处理了单播和广播在带宽资源浪费等方面存在的问题，应用层组播处理了 IP 组播扩展性差、部署困难等问题。应用层组播技术采用网络终端设备实现数据转发的功能，解决了 IP 组播依赖路由器的问题，因此在组播媒体发布内容方面有良好的应用前景。终端媒体设备存在稳定性差的问题，可以随意退出组播树，在传输数据的过程中容易中断，因此由终端设备所形成的应用层组播树很不稳定。应用层组播的稳定性问题，一直以来都是研究的重点问题。本文提出基于效率稳定度的应用层组播算法（A-EBS）。首先分析了影响计算机功能的四个组件：CPU、内存、显卡、磁盘，然后阐释了本文采用 CPU 性能代表本台计算机性能的原因。全方面考虑节点的时延以及在线时间，给出节点的基于效率稳定度（Efficiency-based stability）的定义，最后对基于效率稳定度算法进行详细描述，并通过对实验结果的分析，证明该算法在降低平均时延和提高组播稳定性方面有较好的作用。

目前市场上的直播软件系统有很多，但是绝大部分软件的代码都不是开源的，而且是基于一定的框架所编写的，这样在做实验的过程中，直接使用别人编写好的直播系统就要受到它所使用的框架的约束，更改起来很困难，可扩展性极差。为了解决这个问题，本文实现了基于 FFMpeg 的流媒体直播系统。基于 FFMpeg 的流媒体直播系统从最底层的对视音频数据的采集、传输、编解码和播放做起，在以后可以根据实验者的需要随意改动而不受框架的约束，提高了系统的可扩展性。基于 FFMpeg 的流媒体直播系统在底层客户端采用 FFMpeg 视音频编码技术实现视音频数据的采集、传输、编解码和播放等功能，在上层的服务端，采用本论文提出的基于效率稳定度算法形成组播树，并对组播树进行相关操作。本论文对基于 FFMpeg 的稳定应用层组播流媒体直播系统的整体框架、服务端实现的功能和客户端功能实现的过程进行了详细的描述。在论文最后，总结整篇论文，提出该系统的下一步完善方向。

关键词： 应用层组播；FFMpeg；基于效率稳定度；流媒体直播系统

Abstract

Multicast technology solves the problem of wasting bandwidth resources in unicast and broadcast. Application layer multicast solves problems such as poor scalability and difficult deployment of IP multicast. Application layer multicast technology has a good application prospect in terms of multicast media publishing content , because it adopts network terminal equipment to implement data forwarding function and resolves the dependence of IP multicast on routers. However, the terminal media equipment has a problem of poor stability, it is possible to leave the multicast tree at any time, and it is easy to interrupt in the process of data transmission. Therefore, the application layer multicast tree formed by the terminal equipment is very unstable. So the problem of the stability of application layer multicast has always been the focus of research. This paper proposes an application layer multicast algorithm of efficiency-based stability (A-EBS). First of all, it analyzes the four components that affect the performance of the computer: CPU, graphics card, memory, and disk, and then explains why this article uses CPU performance to represent the performance of the computer. Consider the node's delay and online time comprehensively, and give a definition of the node's efficiency-based stability. Finally, Give a detailed description of the application layer multicast algorithm of Efficiency-based stability. The analysis of the experimental results proved that the algorithm has a good effect in reducing the average delay and improving the stability of multicast.

There are many live broadcast software systems on the market, but the vast majority of software's code is not open source, and they are all written based on a certain framework. Directly using other people's live broadcast system is subject to the constraints of the framework it uses in the process of doing experiments. It is difficult to change and has poor scalability. In order to solve this problem, a live streaming system based on FFMpeg was been written. The FFMpeg-based live

streaming system starts with the collection, transmission, encoding and decoding and playback of video and audio data at the lowest level. it can be arbitrarily changed according to the experimenter's needs without being constrained by the framework and improving the system's scalability. In the underlying, the system uses FFMpeg video and audio coding technology to achieve the functions of data acquisition, transmission, encoding and decoding and playback functions. In the upper server, the multicast tree is formed based on the efficiency-based stability algorithm proposed in this paper . This paper describes the overall framework of the system, the functions implemented by the server and the process of implementing the client functions in detail. At the end, I summarized and looked forward to the work done in this paper, and proposed the further improvement direction of the FFMpeg-based live streaming system.

Key words: Application Layer Multicast; FFMpeg; Efficiency-based stability; live streaming system

目 录

摘 要 I

Abstract II

第一章 绪论 1

1.1 研究背景 1

1.1.1 IP 组播 2

1.1.2 应用层组播 3

1.2 研究意义 4

1.3 研究现状 7

1.3.1 终端系统组播 8

1.3.2 可扩展应用层组播 8

1.3.3 应用层组播体系结构（ALMI） 9

1.4 研究内容 9

1.5 本文的组织结构 10

第二章 关键技术介绍 11

2.1 FFMpeg 11

2.2 SDL（Simple DirectMedia Layer） 12

2.3 RTP/RTCP 协议 13

2.4 JRTPLIB 14

2.5 MySQL 15

2.6 编码技术 16

2.6.1 IPB 帧 17

2.6.2 PTS 与 DTS 17

2.6.3 GOP 的概念 18

2.7 FFMpeg 相关 API 18

2.7.1 通用 API 18

2.7.2 解码 API 19

2.7.3 编码 API 20

2.8 本章小结.....20
第三章 应用层组播稳定性研究.....21
3.1 应用层组播的稳定性问题.....21
3.1.1 问题描述.....21
3.1.2 相关研究.....22
3.2 基于效率稳定度的应用层组播算法.....24
3.2.1 稳定性指标.....24
3.2.2 效率稳定度的定义.....25
3.2.3 算法描述.....26
3.3 仿真实验.....28
3.3.1 仿真环境.....28
3.3.2 参数设置.....29
3.3.3 性能指标.....29
3.3.4 实验结果分析.....30
3.4 本章小结.....33
第四章 基于 FFMpeg 的流媒体直播系统设计与实现.....34
4.1 基于 FFMpeg 的流媒体直播系统整体设计.....34
4.2系统服务端的设计与实现.....34
4.2.1 用户管理模块.....35
4.2.2 组播树构造.....36
4.2.3 组播树维护.....36
4.2.4 组播树显示.....38
4.3 系统客户端的设计与实现.....39
4.3.1 编码.....40
4.3.2 解码.....41
4.3.3 播放.....42
4.4 本章小结.....43
第五章 总结与展望.....45

5.1 总结.....45
5.2 展望.....45
参考文献.....47
攻读硕士期间参与的科研项目.....51
致　谢.....52

第一章 绪论

1.1 研究背景

传统的IP通讯方法有两种：一种是单播（Unicast），即在源主机和目的主机之间，点到点的通讯。此时只在两个节点之间发生信息的传递和接收。通过与web服务器连接，我们才可以查看网页；通过与邮件服务器的连接，我们才可以与别人通过邮件沟通，这时候的数据是单播传送的；另一种是广播（Broadcast），是一种点到多点的通讯，多点代表的是本网段内的所有主机。不管客户需要还是不需要，网络对每一台主机发出的信号，都无条件复制和转发，所有主机，都能够接收到所有消息。广播型网络的一个最典型代表是有线电视网络，所有电视频道的信号都被接收，只不过电视只将其中的一个播放出来。这种方式不需要很多成本，因为不需要选择路径，但是因为要接收所有的数据，存在形成广播风暴的危险。近年来随着视频会议，直播和点播软件的出现，我们需要将信息从一台主机发送给多个主机，而不是一个网段里的所有主机，这些主机也可能不处于同一个网段。如果广播发送，处在不同网段内的主机无法同时接收信息，同一个网段内的主机也无法选择部分接收，占用多余的带宽。若单播发送，将形成很多重复的IP包，这些相同的IP包在网络中占用资源，一个组内有多少个主机，源主机就要发送多少个重复的IP包，增加网络负担[1]。所以，传统的单播方式和广播方式，都不能解决一个主机发送数据，处于不同网段的多个主机接收数据这一问题。在这种情况下，研究学者们提出了组播。组播是在IP网络中，一台主机上发送的数据包，可以被一个确定的主机集合接收，这个确定的主机集合，我们叫做组播组，网络尽力传递数据包，并不保证一定能传递到。而且只有处于这个确定的主机集合里的终端设备才能接收到数据，其余的终端设备接收不到。

组播技术解决了我们上面所提到的问题，可以一个主机发送数据，处于不同网段的、指定的多个主机接收数据。这样既可以高效率的在IP网络中传输数据，还可以解决单播和广播存在的占用网络带宽，源节点负载压力大的问题[2]。组播作为一种新的通讯方式，还有很多衍生的意义。在很多互联网提供的服务领域，如远程教育、网络电视、远程医疗和视频会议等，都是采用组播技术。图1.1显示了在单播情况下，数据的传输情况，图1.2显示了在组播情况下，数据的传输情况。

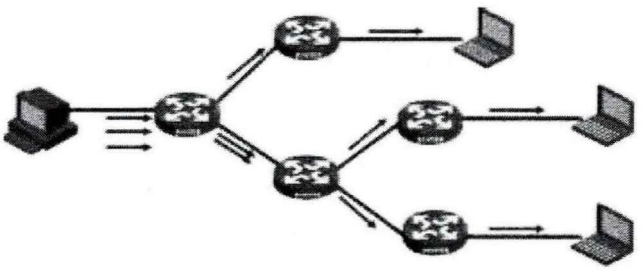

图 1.1 单播示意图

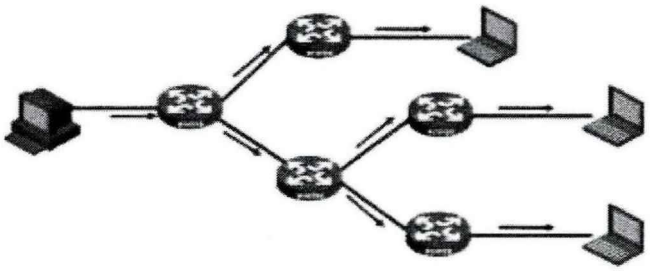

图 1.2 组播示意图

近 30 年来，研究人员都致力于 IP 组播的研究，但是由于 IP 组播需要路由器的支持，部署起来困难，所需要的成本高，所以一时间难以大规模的推广。因此专家学者们开始将眼光投向应用层组播。应用层组播是在终端设备上实现数据转发功能，不需要路由器作为支持，因此不受底层设备的限制[3]，推广使用起来成本低廉，得到广泛的研究和使用。

1.1.1 IP 组播

IP 组播[4]由组播维护协议和组播路由协议共同完成，是指在 IP 网络中，一个主机发送的数据包，发送给某个确定的网络节点的子集，采用尽力传送的方式，并不保证服务质量，这个网络节点的子集，我们称为组播组[5]。IP 组播的特点是，发送数据源的主机只需要向网络中发送一份数据包，路由器会在恰当的分支点对数据进行复制，最后，所有的组成员都可以接收到源主机发送的数据。IP 组播适用于实时不可靠的应用[6]。

IP 组播的实现过程如下：首先，网络设备在转发组播数据之前，会通过相关的组播协议，生成组播树；然后，源组播数据包通过组播树依次向下传递，最后到达组播树中各个成员节点。组播转发树有两种，一是有源树，二是共享树[7]，它们的不同之处是：在有源树中，由根节点向树中的其他节点发送信息；在共享树中，有一个独立的汇合点[8]，该汇合点可以是放在网络上的、某些可选择点。

虽然 IP 组播技术可以实现一个主机发送数据，处于不同网段的、指定的多个主机接收数据，在 IP 网络中，实现高效率的传输数据，解决了单播以及广播存在的占用网络带宽，源节点负载压力大的问题。但在实际研究和运用的过程中，

还存在着以下的不足：

可扩展性不高。路由器需要为每个组播组，单独保存一个状态。

推广使用困难。IP 组播要想实现组播功能，参加了组播的所有端系统之间的路由器，必须要支持组播功能。而 Internet 和 Internet2 不支持组播功能

组播算法的设计困难。在实际的生活中，不同的应用需要不同的组播，这些组播往往差距很大，专家学者们想要设计一个统一的组播模型来适应所有的应用，这几乎是不可能实现的。

管理方法不完善。节点在加入，退出组播树的时候需要很大的开销，而且组播树的管理也要占用很大的开支。

组播地址分配困难。IP 组播的地址空间太小。

在经济方面，传统的计费机制是通过进入流量计费，IP 组播与传统的计费机制不同，不再使用此种计费方法。

除此之外，在拥塞控制、安全等方面，IP 组播同样存在着很多问题。

1.1.2 应用层组播

由于技术上的原因，研究者通过多年的研究，也没有彻底解决 IP 组播遇到的一些问题。一部分专家学者开始尝试，通过单播的模式，以达到组播的效果，这就形成了应用层组播技术。应用层组播同 IP 组播最大的不同，就是应用层组播是以单播的方式来实现组播的功能。应用层组播方式在网络的终端系统对数据进行复制，而不是在路由器上，由终端成员主机完成对数据的路由、复制和转发功能，从而使组播不再受路由器等底层设备的限制。应用层组播形成一个逻辑覆盖网，该网络由组播成员组成，网络的构建与维护是通过应用层组播路由协议来完成，实现高效可靠的传输数据。应用层组播路由协议最主要的功能是维护节点成员的信息，包括节点的加入，离开和节点非正常离开时的处理情况，生成并维护转发树。

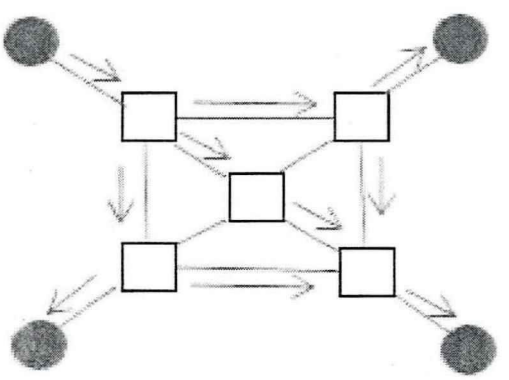

图 1.3 IP 组播

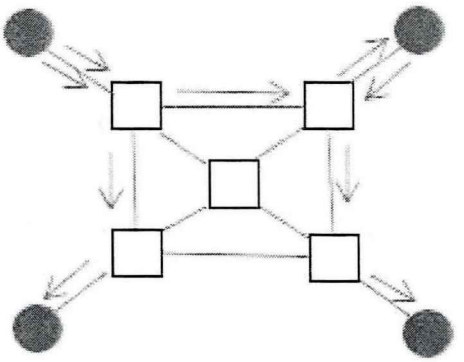

图 1.4 应用层组播

图 1.3 和图 1.4 反映了 IP组播和应用层组播在数据传递时的区别。由图 1.3 可以看出，IP 组播中，数据在路由器上的复制和转发是沿着物理链路方向进行的。而在图 1.4 的应用层组播中，数据的复制和转发功能在终端设备上完成，然后在逻辑链路上进行传递，数据从源节点到目的节点所走的路径，可能不是最短的物理路径，同一条物理链路上，可能存在多条逻辑链路[9]。

与在网络层完成组播功能相比较，在应用层完成组播功能有很多优势：

（1）简洁的地址结构和协议。IP 组播中，地址结构复杂，协议繁琐，应用层组播中，改进了这些方面，使用简略的地址和协议。

（2）可扩展性高。在应用层组播中，增强了业务的可扩展性，因为不用路由器去维持组播状态，而是在主机系统中完成这项功能，这样一来，网络能够负担更多的组播。

（3）简化部署。应用层组播不再受路由器等底层设备的约束，无需改变现在的路由器，可以很快进入应用。

（4）传输可靠及解决拥塞问题。IP组播采用尽力传递数据的方法，不保证质量，而应用层组播技术是基于网络连接所建立的，能够使用 TCP、UDP 服务，简化了组播的控制，同时实现可靠传输功能。TCP 协议是一种可靠传输协议，而且具有拥塞控制，所以应用层组播可以利用 TCP 协议的相关特性，解决 IP 组播不确保数据传递质量，容易拥塞的难题。

1.2 研究意义

在 2018 年 1 月 31 日，中国互联网络信息中心（China Internet Network Information Center），第 41 次发布了《中国互联网络发展状况统计报告》。报告的相关截图如图 1.5 和图 1.6 所示。报告指出，截至到 2017 年 12 月，我国网民的数量达到 7.72 亿人次，于整个 2017 年中，共新增加了 4074 万网民人口。有 55.8%的人口使用互联网，和 2016 年底相比增加了 2.6%。其中有 4.22 亿的网

民用户使用网络直播功能。游戏直播用户的数量比去年同期新增了 7756 万人，人口数量为 2.24 亿，是网民总数量的 29.0%。参与真人秀的直播用户数量比去年同期增加 7522 万人，总人数有 2.2 亿，是网民总数量的 28.5%。图 1.5 反映了从 2007 年到 2017 年，中国网民规模和互联网的普及率的变化趋势。图 1.6 反映了在 2017 年，游戏直播和真人秀直播的用户规模情况，以及它们占整体网民的比例。

图 1.5 中国网民规模及互联网普及率

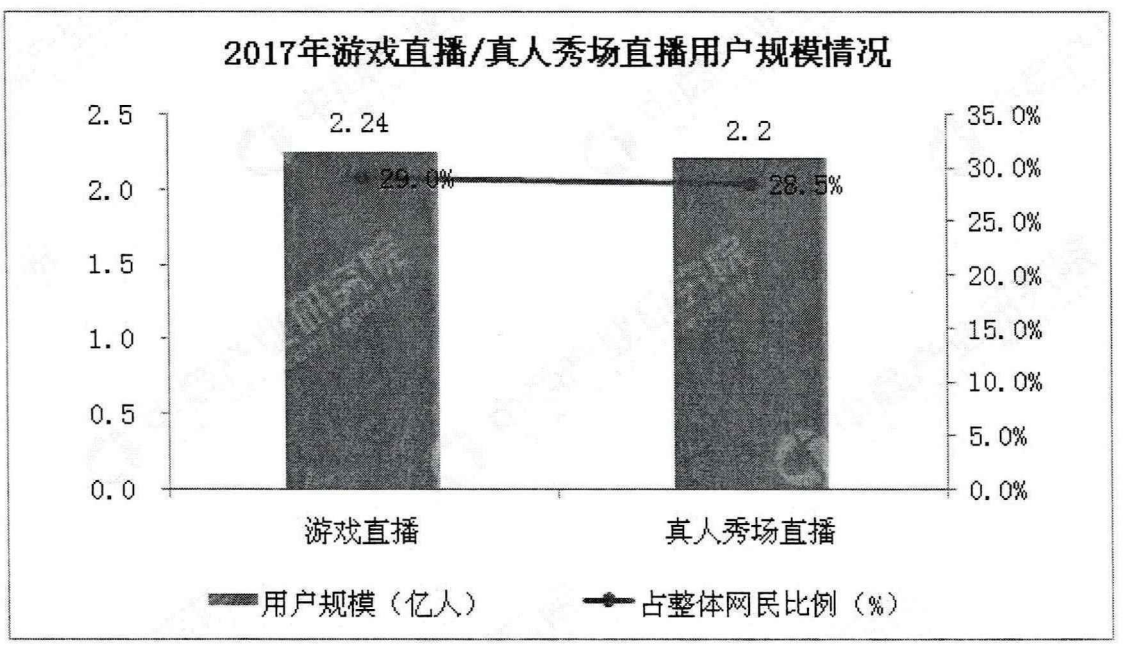

图 1.6 直播用户规模情况

而且近几年来，这些数据都在以加速度的方式增长。由此可见，在日常学习生活中，互联网发挥着越来越不可替代的作用，特别是近些年流行起来的视频直

播软件，深受大家的喜爱，虎牙直播、斗鱼、映客.....几十款直播软件，可以让网友们分享自己的生活，同时了解别人的生活。还有网络直播教学等项目，让生活在教育设施落后地区的孩子们，能够接受更好的教育。目前，计算机技术和通讯技术空前发展，流媒体技术正是在此基础上，得到更好更快的发展。同样的，流媒体的发展，也对信息技术和通讯技术的发展起到正面的推进作用。

流媒体对实时性、带宽和性能要求很高，需要较高的实时性、带宽和性能作为支撑。在传统的 C/S 模式下，服务器的性能和带宽，成为限制服务器能力难以突破的问题[10]，图 1.7 展示了 C/S 模式。因为服务器只能提供一定量的网络带宽资源，而流媒体在复制和传送数据时，需要占用大量的带宽，持续的时间一般也比较长，所以服务器很难满足流媒体的需求。即使使用高档的服务器，也只能为几百个客户提供服务，而高档服务器价格昂贵，根本不具有经济规模性。这就导致一些实时性的流媒体业务，如视音频通话等，无法保证通话质量，使用户体验度不佳，成为限制业务容量扩大的主要问题。另外，网络资源消耗严重也是一个大问题，因为服务器需要为每一个客户主机，单独发一份流媒体数据。对于流媒体，Internet 是不能保证服务质量的，距离服务器较远的客户机，发生包丢失，抖动等情况的概率更大，延时也更长。

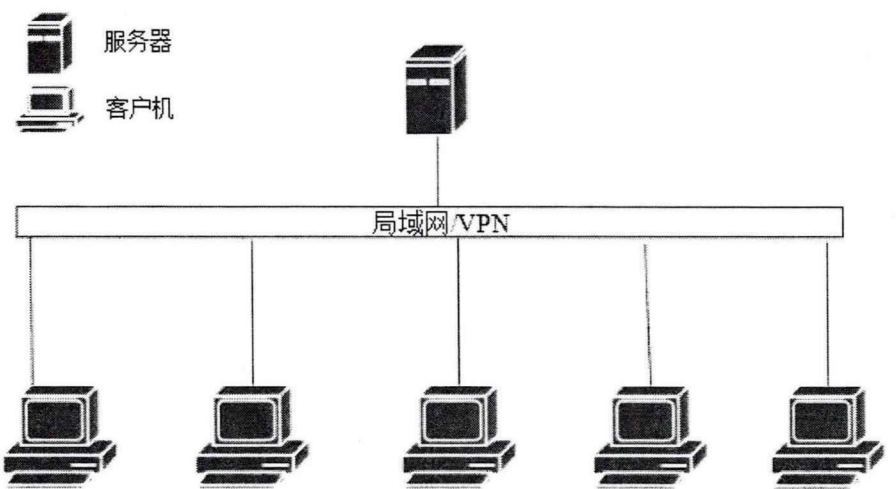

图 1.7 C/S 模式示意图

因此当前在流媒体方向的研究课题主要是：目前网民用户数量急剧增长，如何在有限的网络资源条件下，满足用户的需求，既不能降低网络服务质量，又要使服务成本降低。针对这个需求，目前主要的解决方法有两种，一是采用 Internet 内容发布网络技术，二是采用 IP 组播技术。采用 Internet 内容发布网络是通过代理服务器将流媒体内容复制，发送到网络各处，用户可以通过最近的代理服务器获得流媒体内容。但是代理服务器价格也很昂贵，所以这种方法成本很高，最终也难以大规模部署。IP 组播技术是在底层路由器设备上，实现对数据的复制和

转发，最后发送到组播群里所有的主机，这样主干带宽不需要随着用户数量的增加而增加，由此可以降低运营商的成本。但是由于 IP 组播在协议和技术等方面存在的问题，所以也没有得到大规模应用。于是专家学者们将眼光投向了应用层组播技术，而对等网络（即 P2P 网络）又给流媒体传输问题提供了一种新的思路。P2P 网络[11]里每一个对等实体既可以提供服务，又可以享受服务。在 P2P 网络里，服务不再只由服务器提供，而是由终端主机共同承担，这样在很大程度上减少了服务器的工作量，增加了系统的可扩展性。如今市场上绝大部分视音频直播系统，都是使用应用层组播技术来实现的。正是由于应用层组播在网络资源节约方面，比 IP 组播好很多[12]，所以才能得到大规模的应用。

1.3 研究现状

美国 CMU 大学在 2000 年率先完成了第一套 P2P 视频直播系统的原型 ESM[13]，并取得了极大地成功，该系统是由华人科学家张辉带领的团队实现的。由此开始，对基于 P2P 和流媒体原型系统的研究得到学者们的广泛关注。P2P 网络又被称为点到点或者端到端的网络，如图 1.8 所示。2002 到 2003 年间，各种关于应用层组播的协议出现并迅速发展，得到广泛研究。卡耐基梅隆大学的 Y.H 在 2000 年 6 月发表了一篇关于端系统组播[14]的论文。发表在 ACM SIGMETRICS 上的这篇论文，是应用层组播的研究进入热点的开端。

近年来提出的一些应用层组播有终端系统组播、可扩展应用层组播和应用层组播体系结构。

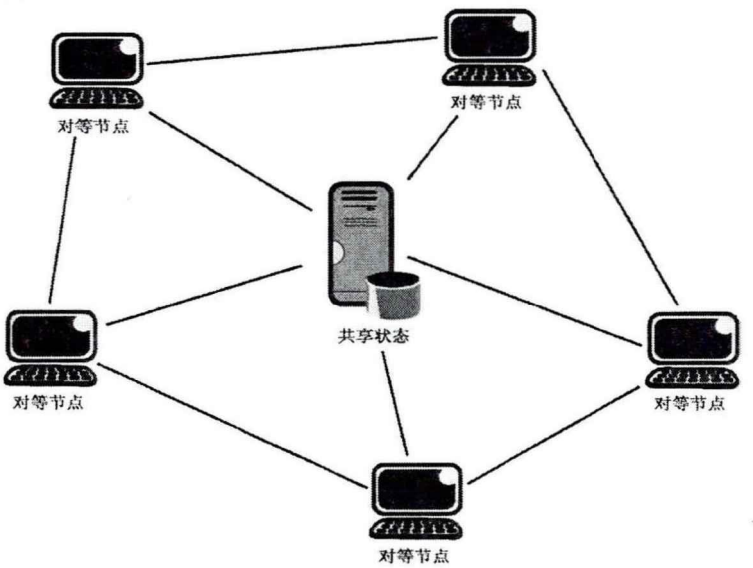

图 1.8 P2P 网络示意图

1.3.1 终端系统组播

Narada 协议[15]是由终端系统组播方案提出的，该协议是最早提出的众多应用层组播策略中的一个。Narada 协议的主要思想是：首先利用成员节点，在它们之间建立一个 Mesh 网，该网络拓扑必须具有网状连接，并不断优化此网络拓扑；然后在该 Mesh 网的基础上，建立一棵伸展树，伸展树的每一个根部都是一个数据源。由于我们是根据 Mesh 网来建立的组播树，所以建立组播树的关键问题在于形成良好的网状拓扑。

Narada 协议指定一个特定的主机作为汇聚点 RP（Rendezvous Point），RP 中保存了所有加入组播的成员节点的状态信息。当一个主机想加入组播的时候，首先从 RP 中获取 Mesh 网中的成员信息，并随机从中选择一些成员节点，向他们发送请求加入成为邻居的信息。这些成员节点中，加入的请求只要被其中的一个节点同意，该节点就算成功加入到 Mesh 网中。Mesh 网中的成员节点会周期性的同相邻节点交换自己所知道的所有成员信息，以此来维持更新网状拓扑。无论是有新节点加入，还是有成员节点退出，最终都会传入到每个节点成员。由于所需要的控制开销大，因此终端系统组播是一种小规模多源的组播方案。

1.3.2 可扩展应用层组播

NICE 协议[16]最主要的一个特点就是可扩展性高，该协议是利用集群思想，来构建一个拓扑结构，是一个分层控制结构。在 NICE 协议中的一切节点，都要加入到组播树的第零层，然后被分别归入到一个个簇丛中，每个簇中选择一个簇首，多个簇首又在第一层形成一个个簇，最终在最上一层只有一个簇首。组播数据传输时，如果发送数据的节点是最底层的一个节点，该簇里的其他节点会首先收到数据信息，然后该群的簇首会把数据发送给同层的其他簇首，由簇首发送给该群的其他节点。保证数据传输路径质量的关键是分层拓扑的构建。该策略最基础的操作，就是在有节点加入、离开时，对这个分层结构进行创建与维护。图 1.9 显示了 NICE 协议的分层结构。

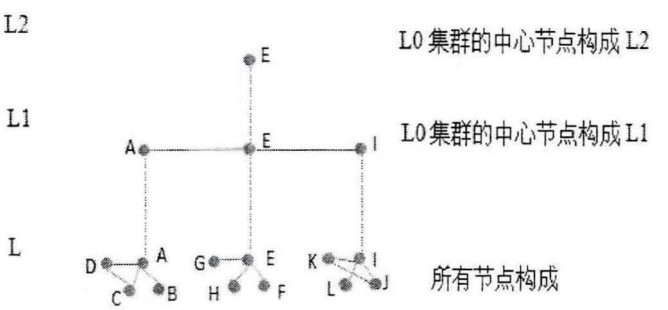

图 1.9 NICE 协议的层结构

1.3.3 应用层组播体系结构（ALMI）

ALMI[17]组播系统，也是针对节点数量较少的情况，采用集中式管理方式，一个会话控制器，加上所有组播成员，形成组播树。会话控制器是一段程序实体，它必须运行在所有的成员都可以访问到的位置上，其功能是注册新加入的成员节点和维持组播树。利用控制器集中管理组播成员并对组播树进行构造。会话控制器传输控制消息，该过程通过它与各个成员之间存在的单播连接完成，数据讯息的分发传输则是沿着组播树进行的。ALMI 利用控制协议来传输控制消息，控制消息负责性能监控、成员管理和路由等功能，其表头格式如图 1.10 所示。

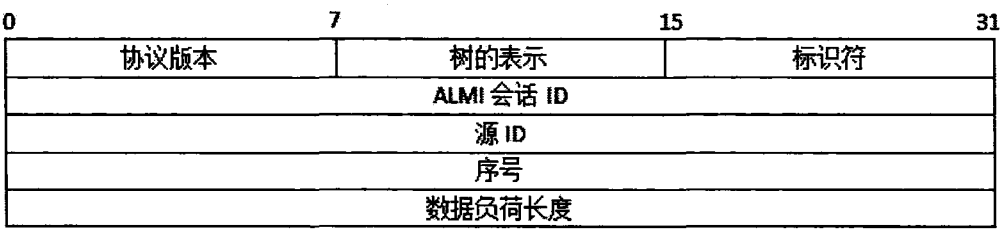

图 1.10 RTP 表头格式

其中树的表示域指明树的版本数，标志位的功能是分发树信息、连接请求和回应、邻居监测更新信息、风力信息和性能监测信息。

1.4 研究内容

应用层组播虽然得到了广泛的应用，但是由于终端设备不稳定导致组播树稳定性差的问题，仍然是应用层组播面临的一大问题。影响应用层组播树稳定的因素有很多，如终端设备的性能、主机的最大转发能力、节点的在线时间和时延等等。

论文对基于 FFMpeg 的稳定应用层组播流媒体直播系统进行了一定的研究，主要研究内容和研究工作包括以下三点：

（1）对应用层研究现状及开发应用层组播流媒体直播系统所需技术进行了归纳总结。对近些年来提出的终端系统组播、可扩展应用层组播和应用层组播体系结构三种方案分别做了介绍，同时对开发基于 FFMpeg 的流媒体直播系统中用到的 FFMpeg 编程技术、RTP/RTCP 协议、MySQL 数据库技术、视音频编码原理等进行了系统的介绍。

（2）文章提出基于效率稳定度的应用层组播算法。针对应用层组播的稳定性问题，对设备的性能、节点的时延和在线时间进行综合考虑，给出基于效率稳定度的定义。当有节点请求加入组播树时，考虑节点的基于效率稳定度和最大剩余出度，选择最优的父节点作为数据转发节点，以此来提高组播树的稳定性。

（3）设计和实现了基于 FFMpeg 的应用层组播流媒体直播系统。该系统从

最底端的视音频采集、编码、传输、接收和解码播放做起，不依赖现有的直播框架，提高了系统的可扩展性。在系统的上层，使用基于效率稳定度的应用层组播算法形成组播树，提高系统的稳定性。

1.5 本文的组织结构

第一章在课题背景、课题意义方面对论文进行介绍，对当前的相关研究做了简要描述，并分别介绍了IP组播和应用层组播，提出本篇论文的研究内容和方向。

第二章是对相关技术的介绍，这些技术在流媒体播放系统的设计过程中都会使用到，主要介绍FFMpeg视音频编解码技术和JRTPLIB技术等。

第三章对本文提出的基于效率稳定度的应用层组播算法进行详细说明，介绍了相关概念及定义，并给出实验结果来验证该算法的优势。

第四章主要在整体架构方面对基于FFMpeg的流媒体直播系统进行解释说明，介绍服务端和客户端的具体功能。对服务端的相关功能截图进行展示，并对客户端流媒体的传输过程及显示进行详细解释。

第五章，对整篇论文进行概括总结。探讨进一步的钻研方向。

第二章 关键技术介绍

2.1 FFMpeg

FFmpeg 是一款多媒体框架，并且在全球范围内处于领先的地位，可以对多种格式的音频数据和视频数据实现运行。同时，它也是一套开源的项目工程，在严格遵守 LGPL/GPL 协议的前提下，可以将数字视频数据、音频数据转变成数据流的形式，FFMpeg 中有一整套完整的解决方案，实现将视音频数据转变成数据流。Libavcodec 作为一款 LGPL 自由软件编解码库，他的技术非常先进，它包含了全部音频/视频编解码库。并且，Libavcodec 是从最底层的开发做起，为编码和解码的质量提供保证，同时大大增强了软件的可移植性。

FFMpeg 具有良好的可移植性，可靠性也很高。FFMpeg 能够在多个平台（Linux、Windows、Microsoft 、MacOSX 等）和架构（arm、x86、mips 等）中编译和运行。这个项目的发起人是 Fabbrice Bellard，从 2004 年到 2015 年，对 FFMpeg 进行维护的是 Michael Niedermayer。我们现在用的 FFMpeg 是放在 Mplayer 的服务器之上，Mplayer 是一个项目组。MPEG 视频编码的标准，正是 FFMpeg 名字的来源。

FFMpeg 具有的功能多种多样，包括转换视频格式、采集视频数据、给视频增加水印、抓取视频图片等等这些。FFMpeg 之所以在直播项目中能得到很好的使用，归功于它具有强大的视频采集功能，它可以对视频采集卡进行采集，对 USB 摄像头捕捉到的图像进行采集，对屏幕进行录制，还可以通过 RTP 的形式，将流数据传递给流媒体服务器。我们平时可以接触到很多不同格式的视频文件，例如 avi、rm，播放时需要把他们转变成我们的播放器支持的格式，使用 FFMpeg 可以很容易的完成转变工作。

FFMpeg 中包含的类库有八个，分别是：

（1）AVFormat ：是 FFMpeg 的文件格式和协议库，是最重要的模块之一。用于解析、生成音视频的各种封装格式。AVFormat 封装了 Protocol 层、Demuxer 层和 Muxer 层，使得协议和格式对开发者来说是透明的。

它的功能有以下三个方面：

①获取解码所需信息；

②生成上下文结构，该结构用于解码上；

③读取视频和音频的帧数据；

（2）AVCodec ：编解码库。用于实现编码与解码功能，编解码的对象是声音或者图像。这个模块也是一个必不可少的部分。AVCodec 封装了 Codec 层，

但是有一些 Codec 是具备自己的 License 的，FFMpeg 是不会默认添加像 libx264 等这样的库的。但是 FFMpeg 就像一个平台一样，可以将其他第三方的 Codec 以插件的方式添加进来，然后为开发者添加统一的接口。

（3）AVUtil ：这是一个核心工具库，里面有公共的工具函数；该模块也是一个重要的部分。其他的很多模块都要依靠该模块，对视音频做一些基本的处理。

（4）AVFilter ：是视音频滤镜库。用该模块，可以完成对视频特效和音频特效的处理，在实际的开发中，使用该模块，可以方便高效的完成对视音频的特效处理。

（5）AVDevice ：是一个设备库，用于数据的输入输出。FFMpeg 中有一个播放视频和音频的工具 ffplay，需要编译出该工具时，必须要确保该模块是处于打开状态的。由于该设备模块播放视音频使用的都是 libSDL，所以同时也需要 libSDL 的预先编译。

（6）SwrRessample：这是一个用于视音频重采样的模块。可以对数字音频进行声道数、数据格式、采样率等多种基本信息的转换。

（7） SWScal：该模块的功能是实现对图像的格式转变。例如，将 RGB 的图像数据转换为 YUV 格式的数据。

（8）PostProctor：后期处理模块。在使用 AVFilter 模块时，必须首先打开该模块的开关，应为 Filter 中会使用到该模块的一些基础函数。

目前我们所接触的很多视音频相关的软件，都是以 FFMpeg 为内核开发的。

使用 FFMpeg 作为内核视频播放器：Mplayer，ffplay，射手播放器，暴风影音，KMPlayer，QQ 影音等。

使用 FFMpeg 作为内核的 Directshow Filter：ffdshow，lavfilters 等。

使用 FFMpeg 作为内核的转码工具：ffmpeg，格式工厂等。

2.2 SDL（Simple DirectMedia Layer）

SDL，是 Simple DirectMedia Layer 的缩写，代码由 C 语言编写而成，但是在 C++语言下，也可以轻松工作，是一个用于开发多媒体的库，代码开源，支持跨平台使用。使用者在处理视频和音频的时候，看不到底层的工作过程，操作简单。SDL 提供了很多方法，这些方法可以用来管理图像和声音的输入输出工作，软件开发人员可以开发出在多个平台都可以运行的软件，只需要使用 SDL 里一些相似代码就可以做到。通过运用一些在外部扩展的库，例如 SDL_ ttf、SDL_ image，可以很容易的完成对平时常见类型图像的使用和加载。

SDL 的数据结构如图 2.1 所示。

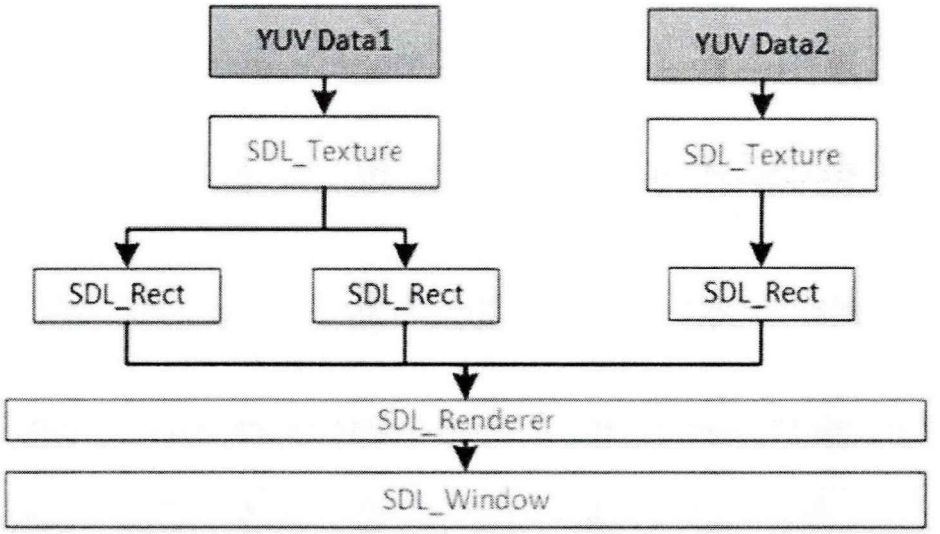

图 2.1 SDL 数据结构示意图

2.3 RTP/RTCP 协议

RTP 是 Real-time Transport Protocol 的缩写，是一种实时传输协议。在传输实时流媒体的过程中，RTP 是一种标准协议，同时是一种不可缺少的技术。利用它能够在不同的网络环境下，实时传输流媒体数据，包括单播网络环境与多播网络环境。在传输数据的时候，RTP 协议大部分是采用 UDP 协议，有时候根据用户需求的不同，也可以运用 TCP 等协议。

RTCP 的功能是提供反馈信息，反馈的信息是数据的分发质量，这项功能是 RTP 协议功能的一部分。RTCP 是一个传输层的标识符，它具有持久性的特点，由 RTP 的源所携带。在数据的发送和接收过程中，接收者要利用它跟踪其中的每一个参加者，因为在传送过程中，出现冲突或者重启程序时，标识符就会改变。

协议分析 ：头部和负载两部分组成了一个完整的 RTP 数据报，头部的每一部分都有特定的意义，负载主要是数据报所携带的实际数据，这些实际数据可以是视频数据，也可以是音频数据。

RTP 的报文格式如图 2.2 所示。

图 2.2 RTP 报文格式

2.4 JRTPLIB

JRTPLIB 是一个代码开源的 RTP 库，面向对象编写。该库提供对 RTP 协议的支持。它使发送和接收 RTP 数据包变得非常容易。它对用户来说是透明的，在协议内部将所要实现的功能进行处理，对用户不可见。这个库实现了网络通讯的功能，实现该功能是通过 socket 机制。所以在多个操作系统上，都可以运行，例如 Linux、Windows 等。

JRTPLIB 函数的使用流程是：

1、在进行流媒体数据传输之前，先生成一个会话实例，表示 RTP 会话，再通过 Create（）进行初始化。

2、通过 setTimestampUnit()函数设置时间戳。

3、设置目的地址。

4、使用 SendPacket（）方法，发送流媒体数据。

5、接收方接收数据报，数据报可以是 RTP 类型，也可以是 RTCP 类型。

JRTPLIB 函数的使用流程如图 2.3 所示。

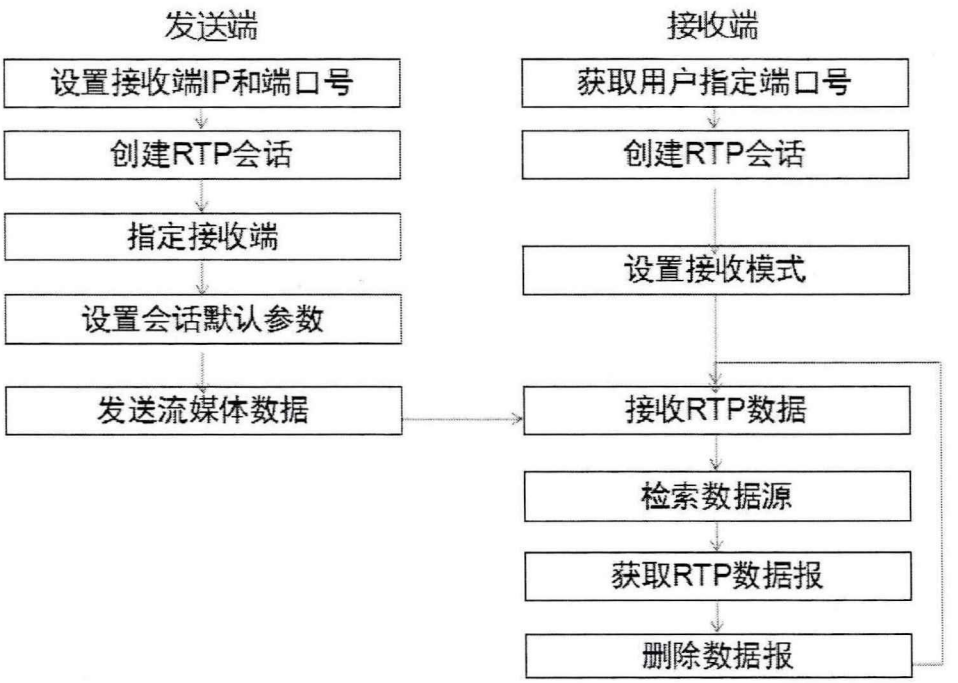

图 2.3 JRTPLIB 程序流程图

2.5 MySQL

MySQL 是一个管理数据库的系统，而且它是一个小型的系统，代码开源，并且是关联式的。在因特网中的小型或中型网站里，绝大部分使用的数据库都是 MySQL。这主要得益于 MySQL 代码开源这一特点。除此之外，还得益于它运行速度快，体积微小，使用者所需要投入的成本低廉等这些特点。MySQL 的社区版，可以和其他服务器一起，组建成性能良好的开发环境。所以被广泛使用。

MySQL 的系统特征如下：

1、使用 C 语言和 C++语言编写，并采用多种编译器来测试，从而确保源代码的可移植性。

2、可以在多种操作系统下工作运行。

3、为多种编程语言提供了 API。这些编程语言包括 C、C++、Python、Java、Perl、PHP、Eiffel、Ruby 和 Tcl 等。

4、支持多线程，充分利用 CPU 资源。

5、SQL 的查询算法简单易用，大幅度增加了查询的效率。

6、MySQL 既能够自己独立运行，也可以和其他软件一起运行。

7、可以支持多种语言。

8、数据的连接途径和连接方式多样化，目前最常使用的是 JDBC，其余的

还包括 ODBC 和 TCP/IP 等。

9、提供管理工具，来实现对用数据库的相关操作，如优化、检查、管理等。

10、处理数据能力强，即使是大型数据库，拥有很多数据，一样可以处理。

11、支持多种存储引擎。

2.6 编码技术

我们平时所接触到的编码[18]，就是依据一定的格式来记录数字数据，这些数字数据是经过采样以及量化后形成的，比如顺序存储和压缩存储等等。压缩编码的原理实际上就是压缩掉冗余信号。我们的听觉能够听到一定范围内频率的声音，频率过高或者过低，都没办法听到。当有两种声音同时响起时，人类只能听到声音较大的一种，较小的声音会被掩盖掉，被掩盖掉的声音我们也无法听到。像这种在我们能够听到的范围之外的音频信号和被隐蔽掉的音频信号，我们称为音频冗余信号。常用的音频编码方式有 WAV 编码、MP3 编码、AAC 编码和 Ogg 编码。对于视频来说，存在有更大量的冗余数据，因为视频数据具有相关性，而且相关性很强[19]。视频是由一帧帧画面连续出现形成的，连续的帧画面中会有很多重复的部分，这些重复的部分就是冗余数据，通常使用帧间编码技术，来去除这些多余的信息，这些被除去的信息又分为在时间方面的冗余信息和在空间方面的冗余信息，具体包括以下几个部分。

运动补偿 ： 这是一种预测活动，先对之前的局部画面进行分析，进而预测出来现在的局部画面，这是一个很有效果的方法。

运动表示 ： 是在描述图像的运动信息时，运用不同的运动矢量，来描述处于不同地域里的画面。

运动估计 ： 是一套完整的技术，该技术是从排好序的视频队列里抽取图像的运动信息。

在去除空间上的冗余信息方面，该技术也能很好的完成。

ISO 制定了 MPEG 图像编码标准，MPEG 这种压缩算法，适用于处理动态的视频。处理动态视频时，该算法可以取得一个很好的压缩比，因为 MPEG 不仅会对每一帧单独的画面编码，同时还会使用画面中的一些相关原则，去除多余的信息[20]。MPEG 图像编码技术目前包括的主要版本有：Mpeg1（用于 VCD）、Mpeg2（用于 DVD）和 Mpeg4（现在流媒体使用最多的版本）。

相比于 ISO 制定的 MPEG 图像编码技术，ITU-T 制定的一系列视频编码标准是一套单独的体系。其中 H.264 编码技术吸取了以往 H.263 等标准的精华部分，设计方法简洁，集多种优点于一身[21]，所以推广简单，比 MPEG 更容易被接受。H.264 编码标准是现在使用最多的技术，它创造了多种新的压缩技术，例如多参

考帧，帧内预测，整数变换，多块类型等，在很大程度上提升了图像的压缩性能，也进一步完善了系统。

2.6.1 IPB 帧

在视频中，每一个静止的画面都是一帧数据，在实际的视频压缩过程中，我们会采用各种各样的算法，来减少帧数据的容量，IPB 帧就是众多压缩算法中的一种[22]。

I 帧：英文名称是 intra picture，是帧内编码帧[23]。MPEG 编码技术中，有一种 GOP 视频压缩技术，一般情况下，I 帧就是 GOP 的第一个帧，I 帧是一幅静态的画面，可以作为一个参考点，用于视频的随机访问。I 帧是一幅图像经过适当的压缩处理后形成的。图像经过压缩处理形成 I 帧的过程，就是除去视频中的空间冗余信息的过程，通过 6:1 的压缩比压缩过后，用户也不会感觉视频变的模糊了。

P 帧：英文名叫 predictive-frame，是前向预测编码帧，P 帧也被称为预测帧，P 帧前面是已经经过编码处理了的帧的图像序列，通过除去这些帧之间的时间冗余信息，来实现减少传输数据量的效果。P 帧去除掉的是视频的时间冗余。

B 帧：英文名称是 bi-directional interpolated prediction frame，是双向预测内插编码帧，又被称为双向预测帧。在 B 帧的编码过程中，需要同时考虑源图像序列之前的已经编码过的帧和源图像序列之后的已经编码过的帧，来达到减少传输数据量的目的，P 帧去除掉的也是视频的时间冗余

根据以上对 I帧，P帧和 B帧的定义，从编码的角度来看：

I 帧通过视频解压算法的解压过程处理后，能够形成一张完整的视频图像，可单独存在，不需要参照任何其他的帧，因此 I 帧去掉的是空间冗余信息。

P 帧要想解码成一张完整的视频图像，必须要参考前面的一帧画面，可以是 I 帧，也可以是 P 帧。所以 P 帧去掉的是时间冗余信息。

B 帧要解码成一幅完整的图像，必须参考它前一帧数据和后一帧数据，前一帧数据可以是 I 帧也可以是 P 帧，后一帧数据是 P 帧。因此 B 帧去掉的也是视频的时间冗余信息。

2.6.2 PTS 与 DTS

DTS 是解码时间戳，英文名称是 Decoding Time Stamp，是在视频解码的过程中使用的。它的作用就是告诉播放器，这一帧数据应该在什么时候被解码。PTS 是显示时间戳，英文名称是 Presentation Time Stamp，主要是在视频解码阶段，控制视频的播放与输出。PTS 是用来告诉播放器，这一帧数据什么时候应该被显示。

在 FFMpeg 技术中，图像解码前的压缩数据，或者编码后的压缩数据，是存放在 AVPacket 结构体里面的。由 DTS 来决定，AVPacket 里面的数据什么时候可以被解码。一幅图像解码后的数据，或者压缩前的源数据，是存放在 AVFrame 结构体里面的。所以对于视频而言，AVFrame 就相当于是一幅图画，由 PTS 来决定它什么时候在播放器中显示。如果在一个视频中，所有帧的编码次序都是按照帧的显示次序依次进行，那么视频中，帧的解码次序应该和显示次序一致。但实际情况是，在大多数像 H.264、HEVC 编解码的标准中，解码次序和显示次序是不一致的，所以我们才会使用 PTS 与 DTS。虽然 PTS、DTS 是用来指导播放器的播放的，但是它们并不是在播放器里形成的，而是在编码的时候就形成了，由编码器生成。在没有 B 帧的情况下，PTS、DTS 是相同的，由于 B 帧打乱了他们的顺序，所以在有 B 帧的时候，两者的顺序就不相同了。

2.6.3 GOP 的概念

GOP：Group Of Picture，就是在两幅 I 帧中间形成的图像组。在编码器中，有一个参数 gop_size，代表的是在两个 I 帧之间有多少帧图像。gop_size 参数的值越大，代表在两个 I 帧之间有更多的画面，所示视频就可以更加清晰，画面质量更好，我们可以根据实际业务的需要，来为 gop_size 设置不同的值。在一个 GOP 中，I 帧所占的容量最大。因为 I 帧的解码不需要参考其他的帧，可以独立解码出一帧图像，I 帧是随机访问的开始帧。在解码端，必须要从一个 I 帧开始，才能解码出正确的原始画面。在实际应用中，想要提高视频画面的清晰度，可以尽可能多的使用 B 帧，因为 B 帧的压缩比最大，这样可以节省存储空间来存放更多的 I 帧，提高画面质量。

2.7 FFMpeg 相关 API

2.7.1 通用 API

（1）av_register_all

该函数用来注册所有 config.h 里面开放的编解码器，然后会注册所有的 Muxer 和 Demuxer（也就是封装格式），最后注册所有的 Protocol（即协议层的东西）。这样，在 configure 过程中开启（enable）或者关闭（disable）的选项就能作用到运行时。

（2）av_find_codec

该函数里面其实包括两部分的内容，一部分是寻找解码器，一部分是寻找编码器。在第一步的 av_register_all()方法里面，已经把编码器和解码器都存放在一个链表中了，在这里寻找编码器或者寻找解码器都是从第一步构造的链表里进行

遍历查找，通过 Codec 的 ID 或者 name 进行条件匹配，最终返回所要查找的 Codec。

（3）avcodec_open2

该函数是用来打开编解码器（Codec）的，无论是编码过程还是解码过程，都会用到该函数。此方法有三个参数。第一个是 AVCodecContext，解码过程由 FFMpeg 引擎填充，编码过程由开发者自己构造，如果想要传入私有参数，则为它的 priv_data 设置参数；第二个参数是上一步通过 av_find_codec()函数寻找到的编解码器（Codec）；第三个参数一般会传 NULL。

（4）avcodec_close

该方法是 avcodec_open2()方法的一个逆过程，找到对应的实现文件中的 close 函数指针所指向的函数，然后该函数会调用对应第三方库的 API 来关闭掉对应的编码库。

2.7.2 解码 API

（1）avformat_open_input

该函数会根据所提供的文件路径，判断出文件的格式，其实就是通过这一步来决定使用的到底是哪一个 Demuxer(封装格式)。例如，如果是 flv，那么 Demuxer 就会使用对应的 ff_flv_demuxer，所以对应的关键生命周期的方法 read_header、read_packet、read_seek、read_close 都会使用该 flv 的 Demuxer 中函数指针指定的函数。

（2）avformat_find_stream_info

该函数非常重要，作用是把所有 Stream 的 Metadata 信息填充好。内部方法会首先找到对应的解码器，然后打开对应的解码器，接着利用 Demuxer 中的 read_packet()函数，读取一段数据，进行解码，当然解码的数据越多，分析出的数据流就会越准确。对于网络资源来说，不能很快的得到准确的信息，因此该函数有几个变量，可以控制读取数据的长度，一个是 probe size，一个是 max_analyze _duration，还有一个是 fps_probe _size，这三个参数一起作用，共同控制解码数据的长度。

（3）av_read_frame

该方法用于读取数据，读取出来的数据是 AVPacket。首先，该函数会委托到 Demuxer 的 read_packet 方法中去，read_ packet 通过解复用层和协议层的处理后，会将数据再返回到这里，在该方法中进行数据缓冲处理。

（4）Avcodec_decode

该方法也包含两部分，一部分是解码视频，一部分是解码音频。通过上面的函数分析可以知道，解码是委托给对应的解码器来实现的，在打开解码器的时候

就找到了对应解码器的实现。其中会有对应的生命周期的实现，init，decode 和 close 三个方法，分别对应解码器的打开，解码和关闭过程。

（5）avformat_close_input

该函数负责释放对应的资源，首先会调用对应的 Demuxer 中的生命周期 read_close 方法，然后释放掉 AVFormatContext，最后关闭文件或者远程网络连接。

2.7.3 编码 API

（1）avformat_alloc_output_context2

该函数内部，需要调用一个方法来分配一个 AVFormatContext 结构体，该方法就是 avformat_alloc_context()，最关键的还是根据上一步注册的 Muxer 和 Demuxer 部分（封装格式）去找对应的格式，如果没有找到对应的格式，也就是说在 configure 选项中没有打开这个格式的开关。就会返回一个错误提示，提示用户找不到对应格式。

（2）avio_open2

该函数首先调用 ffurl_open 函数，构造出 URLContext 结构体，这个结构体中包含了 URLProtocol（需要去第一步 register_protocol 中已经注册的协议链表中寻找）。接着会调用 avio_alloc_context 方法，分配 AVIOContext 结构体，并将上一步构造出来的 URLProtocol 传递进来。最后把上一步分配出来的 AVIOContext 结构体赋值给 AVFormatContext 的属性。

2.8 本章小结

本章重点介绍了开发基于 FFMpeg 的流媒体直播系统所用到的主要技术。分别介绍了 FFMpeg 编程技术，RTP 和 RTCP 协议，JRTPLIB 函数的使用流程以及 MySQL 数据库技术，还详细介绍了与视音频编解码相关的知识和 FFMpeg 的有关 API。

第三章 应用层组播稳定性研究

应用层组播在实时网络中占有很大的优势，所以很多专家学者一直致力于对应用层组播的研究，提高组播的稳定性，并取得了很好的成果。

3.1 应用层组播的稳定性问题

3.1.1 问题描述

在 IP 组播中，通过路由器，来实现数据的复制、转发功能，端主机只需要接收路由器发送来的数据即可，所以在 IP 组播树中，路由器作为中间节点复制转发数据，终端主机只能作为叶子节点。但是在应用层组播中，我们将数据的转发和复制功能转移到终端设备上来实现，处于中间节点的端主机即要接收数据，还要完成对接收数据的复制和转发，将数据发送给其孩子节点。由于主机具有高度的动态性，不像路由器那么稳定高效，这种将复制转发功能由路由器移至主机的方案，将引起两个严重的问题：组播树不稳定和组播延时大。在本模块，将重点讨论应用层组播的这两个问题。

在 IP 组播树中，由于端主机只能是组播树中的叶子节点，如图 3.1 所示，所以任何主机的加入和离开都不会对组播树产生影响。在应用层组播树中，如果一个主机是叶子节点，那么它的加入、退出和失效不会对组播树产生任何影响；如果一个主机是中间节点，它不仅要接收数据，还要将接收到的数据复制并转发给它的孩子节点，所以该节点的失效或离开，会使其孩子节点接收不到数据，进而影响组播树的稳定性。

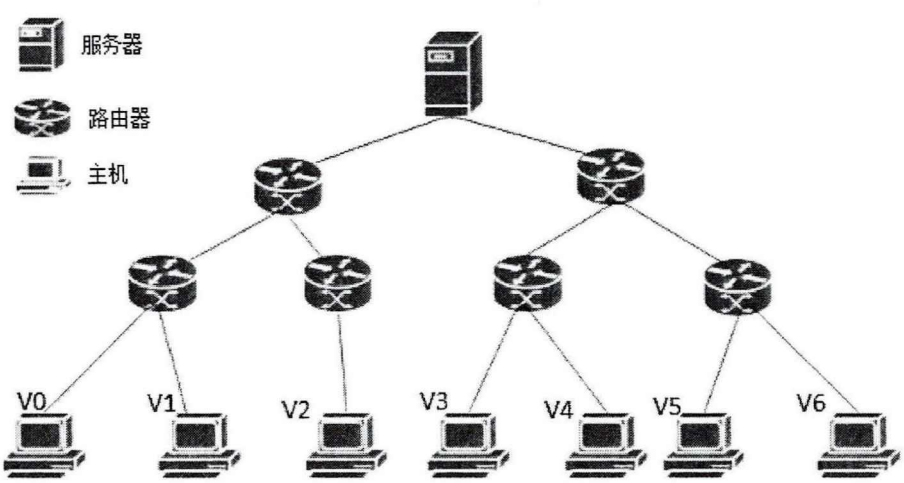

图 3.1 IP 组播示意图

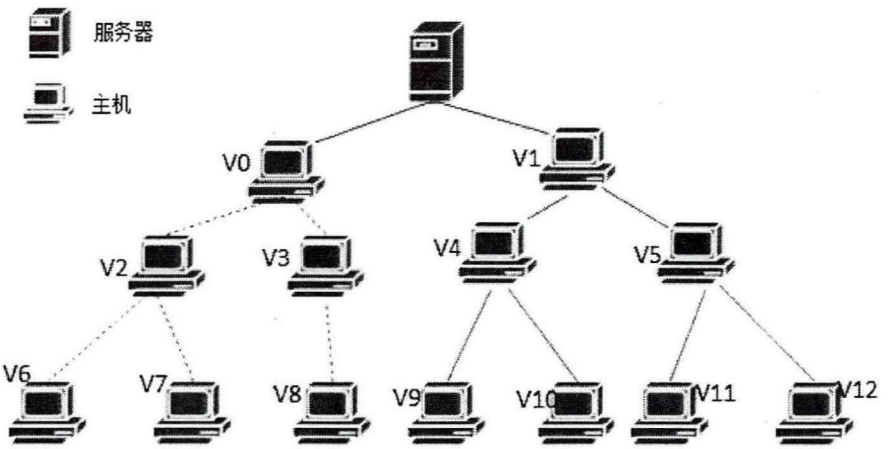

图 3.2 应用层组播示意图

如图 3.2 所示，在应用层组播中，如果主机 V0 退出组播，它的子孙节点与组播树的连接都会断开，在重新加入到组播树中之前，这些子孙节点都无法接收到服务器发送的数据。如果频繁地出现节点退出和失效的问题，将会严重影响用户的体验效果，降低系统的效率。

3.1.2 相关研究

对于以上描述的应用层组播出现的问题，这些年研究学者们一直致力于提高应用层组播的稳定性[24]。来自国防科技大学计算机学院的苏金树，在应用层组播稳定性提高综述[25]一文中总结了影响应用层组播稳定性的 3 方面因素：（1）节点离开事件的发生频率。也就是在单位时间里，节点离开组播树的次数；（2）节点离开事件发生时，组播树中有多少成员节点受到影响，它能表示该成员离开事件的影响区域范围；（3）节点离开后，因此受到影响的节点，重新请求并加入到组播树中需要的时间。它表示在连接中断后，组播树恢复的快慢情况。由此可知，要提高应用层组播的稳定性，就要从这三个影响因素方面入手，其对策分别是：（1）减少节点离开事件的发生频率；（2）缩小节点离开事件的受影响范围；（3）节点离开后，减少组播树的恢复时间。

处理减少节点离开事件的发生频率[26]问题，本质思想是提高节点的稳定性，尤其是中间节点的稳定性，中间节点的退出和失效直接关系到组播树的稳定性，采用专职代理型应用层组播，是目前采取的最具有代表性的方法。端主机要想享受组播服务，需要通过访问代理服务器获得，代理服务器通常是由 ISP[27]（Internet Service Provider）专门部署的高性能节点，与普通主机节点相比，稳定性高，所以发生节点失效情况的概率更低。代理型应用层组播结构如图 3.3 所示。

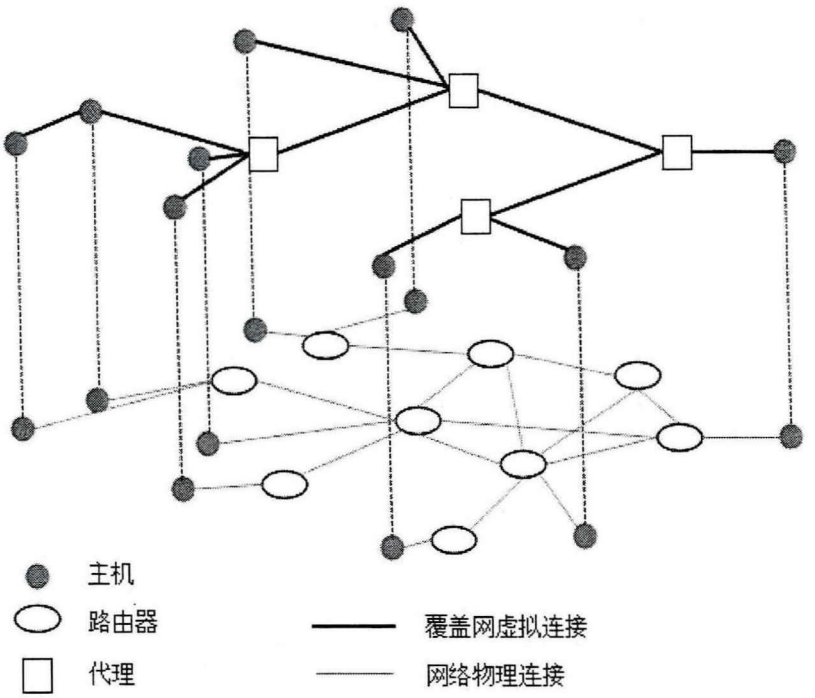

图 3.3 代理型应用层组播示意图

解决减少节点离开事件发生后，对组播树的影响范围问题，主要利用节点的属性或者采用基于多树的组播。根据组播树中，节点所处的位置不同，离开事件发生后，影响范围也不同这一特点，越靠近根节点的成员节点，拥有的子孙节点数目就越多，离开时传输中断的节点数目也就越多；而靠近叶子节点的成员节点所拥有的子孙节点数目较少，它离开时，影响到的节点数目也较少；叶子节点没有子孙节点，离开或失效对组播树没有影响。根据组播树的这一特点，在距离根节点较近的地方，使用稳定性较高的节点，以此提高组播树的稳定性。

来自法国的 Roca V, El-Sayed A 提出 HBM[29]协议。将节点根据其性能分为两类：稳定节点和非稳定节点。稳定节点成为组播树的中转节点，非稳定节点只能成为叶子节点。HBM 协议为每个节点分配一个能力值属性，如果节点的能力值高于一个阈值，则可作为组播树的中间节点，此时节点的能力状态为中转节点状态（transit_possible）；如果节点的能力值在一个阈值范围内，则只能成为叶子节点，此时节点的能力状态为叶节点状态（leaf_only），如果节点的能力值低于一个阈值，则节点不能加入组播树，此时的能力状态为断开状态（disconnected）。HBM 协议把节点稳定度作为构造应用层组播树的主要考虑因素。

基于多树的组播思想是在所有节点之间形成不同的多棵组播树，多棵组播树的根节点是同一个源节点，如图 3.4 所示。利用多描述编码 MDC[30]技术，即 Multiple Description Coding 技术，将组播数据的内容，编码为多个数据带，这些

数据带分别在不同的组播树中传输。由于一棵组播树中的中间节点可能是另一棵树的叶子节点，所以当一个节点离开或失效时，在一棵组播树中受影响的节点在另一棵组播树可以正常接收数据，这样节点虽然会少收到一部分数据，导致数据不完整，但是在对数据质量要求不高的项目中，仍然可以正常播放。

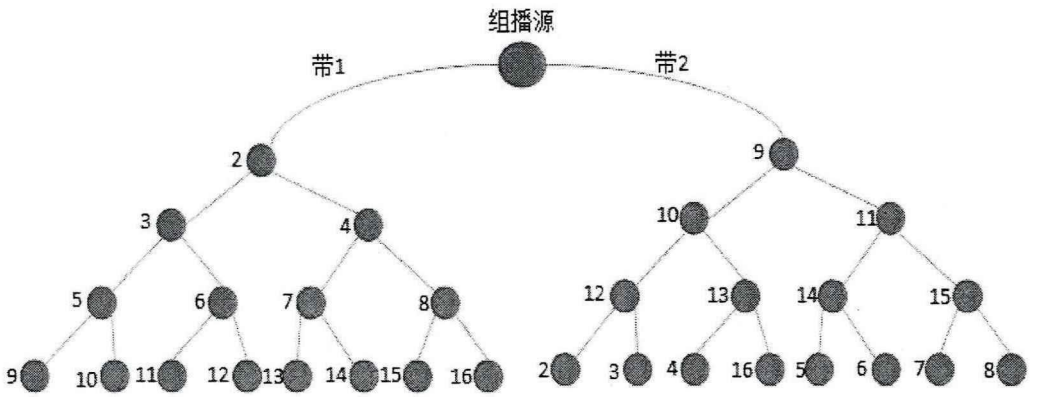

图 3.4 基于多树的组播示意图

缩短组播树的恢复时间，就是在节点离开事件发生后，缩短受影响的节点重新加入组播树的时间。恢复组播树分为发现节点离开和重构组播树两个步骤。组播树中的节点通过心跳机制和探测机制来检测节点的失效行为。重构组播树时，包括前向式重构、后向式重构，前向式重构是在节点离开之前，就计算出组播树的方案；后向式重构是在节点离开事件发生后，通过一定的方案重新加入到组播树中。Banerjee 等人提出的 PRM[29]（Probabilistic Resilient Multicast）算法、EL-Sayed 等人提出的 RVL[31]（Redundant Virtual Link）算法、Wong 等人提出的 LER[32]（Lateral Error Recovery）算法等都是前向式重构算法。Bawa 等人在全面研究后向式重构策略[33][34]后，提出 RAT(RooT-All)、GFA(GrandF ather-All)、RT(RooT)、和 GF(GrandFather)策略[35]。

3.2 基于效率稳定度的应用层组播算法

3.2.1 稳定性指标

计算机的性能[36]一般主要包括处理器性能，内存性能，显卡性能和磁盘性能，这些性能的好坏可以由专业的软件测试得到一个分值，如鲁大师、PCmark。

处理器，是电子计算机里最核心的配件，通常被称为 CPU。对计算机数据进行处理，对电脑发出的指令进行解释，是 CPU 最主要的功能。显卡，它的主要任务是对电脑的数字信号进行转换，将转换的模拟信号进行输出显示。内存，作为电脑的重要组成部分，实现了与 CPU 的衔接沟通。软件的响应速度、计算机的读写速度，这些是受到磁盘性能影响的。

应用层组播中，主机的主要功能是：接收、复制和转发数据。在计算机的四个主要性能中，CPU 的主要功能是对计算机数据进行处理，对电脑发出的指令进行解释，它作为计算机的核心单元，性能的强弱同时也反映了这台计算机的性能，CPU 越高，计算机的运行也就越流畅，所以对于计算机来讲，CPU 的性能指标尤为重要。而且由图 3.5 鲁大师性能测试结果可见，处理器的性能得分占据电脑综合性能得分的绝大部分，所以在此部分中，以 CPU性能代表计算机综合性能更为合适。电脑性能和电脑的稳定性息息相关，电脑的性能越好，稳定性越高。

图 3.5 电脑性能测试结果

文献[37]指出，用户的在线时间，是符合对数正态分布。罗建光等人统计分析了 1000 多万条直播系统的用户真实记录，最终验证了这一结论。并通过对用户在线时间做进一步的研究分析，发现随着用户已经在线时间的增长，使用者的平均剩余在线时间也呈现增长的趋势，这两者是呈正比例的关系。因此可以得出，用户已在线时间越长，该节点的稳定度越高。即节点稳定度与在线时间成正比。

应用层组播的一个重要应用领域是实时传输，例如视频会议，网络直播等，因此要严格的控制时延[38]。节点的时延表示数据从源节点出发，到达该节点时所经历的时间。发送时延、处理时延、传播时延、排队时延，这四个方面共同构成了节点的时延。经调查研究发现，人们在浏览网页或者视频会议时，能够容忍的最大时延是 250ms，超过这个时间，会严重影响用户的体验效果。

3.2.2 效率稳定度的定义

根据以上说明，综合考虑节点的在线时间，性能和时延，定义节点的基于效率稳定度（Efficiency-based stability）为：

$$s = \frac{time * PE_i'}{delay} \tag{3.1}$$

其中，*time* 是节点的在线时间，节点的在线时间表示为，从节点加入到组播树开始计起，到当前时刻的时间。*delay* 是节点的时延，时延表示数据从根节点出发，到达该节点所经历的时间。PE_i'是节点性能 PE_i 经过归一化处理后的值，PE_i 是指主机节点的 CPU 性能，由鲁大师软件测试得到。根据图 3.5 鲁大师的测试结果可以看出，CPU 性能的得分是个很大的数值，而在线时间和时延是一个较小的数值，所以要对性能做归一化处理。

假如一棵组播树中共有 n 个节点，节点 h_i 的节点性能为 PE_i，归一化后节点的性能为 PE_i'，PE_i'的计算公式为：

$$PE_i' = \frac{PE_i}{\sum_{j=1}^{n} PE_j} \tag{3.2}$$

S 值越高的节点，其稳定性越好，接收数据的时间较短，可以放在组播树中距离根节点较近的位置。

3.2.3 算法描述

本文所提出的基于效率稳定度的应用层组播算法（A-EBS），除了考虑到基于效率的稳定度 S 外，还考虑了节点的度，将剩余出度较大的节点放在组播树的较上层位置，可以使组播树高度较低，减少整体时延，从而提高稳定性。节点的出度由节点的性能和可用带宽共同决定，在节点加入组播树之前就已经确定。节点的剩余出度计算公式为：

$$d_{rem}(v_i) = d(v_i) - CN(v_i) \tag{3.3}$$

其中 $d_{rem}(v_i) \epsilon N+$，表示节点 v_i 的剩余出度，$d(v_i) \epsilon N+$，表示节点 v_i 的出度，$CN(v_i) \epsilon N+$，表示节点 v_i 的孩子节点数量。

算法的核心思想是：当有节点请求加入组播树时，对组播树中还具有剩余出度的节点按照 S 值降序排列，选择前 α 个节点，然后对这 α 个节点按照剩余出度降序排列，选择第一个节点作为待加入节点的父节点。

算法的具体实现过程为：

开启直播的节点为组播树的根节点 r，将 r 加入到候选父节点队列 *CPN* 中。当有节点 q 请求加入时，执行以下过程：

①计算候选父节点队列中，每一个节点的 S 值，按降序排列纳入到可选父节点队列 *OPN*。

②从队列 *OPN* 中选取前 α 个节点，选择剩余出度最大的节点 p，作为新加

入节点的父节点。

③p 的剩余出度=p 的剩余出度-1，判断 p 的剩余出度是否大于 0，如果不大于 0，将 p 移出队列 *CPN* 并更新剩余出度表。

④判断新加入节点 q 的出度是否大于 0，如果大于 0，将 q 加入到候选父节点队列 *CPN*，更新剩余出度表。

其中，α 的取值根据项目对 S 值和剩余出度的需求程度来决定。如果项目要求生成的树更矮一些，α 取值就大一些；如果要求新加入节点的父节点稳定性更高，则 α 取值相对较小。

算法的伪代码描述如下：

Algorithm 3.1 Application Layer Multicast Algorithm of Efficiency-based Stability

```
Input: q, CPN
Output: T(V,E)
Procedure:
1：T=Φ, OPN=Φ, PE_total=0
2：If CPN==Ø then
3：    V(T,E)← q.
4：    Return T(V,E)
5：Else
6：    For   j from 1 to n
7：        PE_total+=PE_j
8：    End for
9：    For   each node in CPN
10：     S=time*(Node.PE_i+PE_total)/delay
11：     Node.S← S
12：     OPN.add(node)
13：   End for
14：   OPN←OPN is sorted in descending order of S and ake the first α values
15：   OPN is sorted in descending order of remaining degrees
16：   father←select the first node of OPN
17：   T(V,E)← q
18：    degree[father]=degree[father]-1
19：    If degree[father]<=0 then
20：        CPN.delete(father)
21：    End if
```

```
22:     If degree[q]>0 then
23:         CPN.add(q)
24:     End if
25: End if
26: Return   T(V,E)
```

通过伪代码可以看到，该算法执行一层 for 循环，所以算法的时间复杂度为 O(n)。申请了一个临时空间 OPN，所以算法的空间复杂度也是 O(n)。

3.3 仿真实验

本文通过仿真实验，在平均加入时延、最大加入时延、链路伸展度这三方面，将基于效率稳定度算法和度优先算法、NICE 算法与随机算法的性能进行对比。度优先算法在选择父节点时优先考虑节点的度，NICE 协议是一种分层分簇协议，随机算法是选择父节点时，随机选择有剩余出度的节点。

3.3.1 仿真环境

本算法在实验过程中，模拟环境采用 OMNet++，代码的编写和实验的模拟是以 OverSim[39]为基础的。

OMNet++是一种用于网络仿真的软件，在网络仿真这一领域里，占有举足轻重的地位，它的代码是开源并且免费的。OMNeT++是最近这些年，在工业领域、科学领域比较常用的一种网络仿真平台。OMNeT++作为一款用于离散事件的仿真软件，具有强大并且完善的图形和界面接口。OverSim 在 OMNET++的仿真环境下完成工作，是一个覆盖网仿真的框架，并且代码也是开源的。在这个 P2P 仿真器里，包含有很多的 P2P 协议。通过复用 OverSim 中的代码，可以容易的实现自己的覆盖网络模型。下图 3.6 是当有 150 个节点数目时，基于效率稳定度算法的模拟仿真架构图。

图 3.6 基于效率稳定度算法的模拟仿真架构图

3.3.2 参数设置

节点的最大出度是该节点所能接受的最大数目的子节点，参考文献[40]和[41]的实验部分，本文在仿真实验中，最大出度服从整数区间[2，6]的均匀分布，即节点的最大出度取值为 2 到 6 及其之间的整数。

本实验设置组播树的更新调整维护周期是 200s。因为在文献[42]中，通过对视频日志的统计分析发现，节点在线时间少于 200s 时，大概有 70%多的节点在连接以后的 200s 内的时间里退出。

在仿真实验中，节点的性能是通过随机函数生成的一个随机数。

3.3.3 性能指标

平均加入时延：

$$t_{ave}=\frac{1}{n}\sum_{i=1}^{n}(t(v_i)-t_0) \tag{3.4}$$

其中节点 i 表示为 v_i，t_0 表示节点 v_i 请求加入组播树的时刻，$t(v_i)$ 表示父节点向请求加入节点发送回应消息的时刻，n 代表组播树中节点的总个数。平均加入时延表示所有节点从请求加入组播树开始，到加入到组播树中所经历的时间的平均值。平均加入时延越小，证明节点加入组播树所用的时间越少。

最大加入时延：

$$t_{delay}=t_{max}-t_1 \tag{3.5}$$

其中，t_{max} 表示父节点向请求加入节点回应同意加入消息的时刻，t_1 代表加入节点 v_i 发送请求加入的时刻。最大加入时延表示组播树的一切节点中，从请求加入组播树开始，到加入到组播树中所经历的时间最长的节点的时间值。

平均链路伸展长度：

$$Leg(T)=\frac{1}{n}\sum_{i=1}^{n}Leg(v_i) \tag{3.6}$$

平均链路长度是衡量数据路径质量的一个标准。它表示数据从组播源出发，到达该节点时，所经历的路径长度的平均值。

3.3.4 实验结果分析

将 A-EBS 算法同度优先算法、NICE 算法和随机算法[43]进行对比。

平均加入时延：下图 3.7 反映了 A-EBS 在平均加入时延方面与 NICE 算法、随机算法和度优先算法的对比情况。由图可以看出，只有在节点个数为 20 时，A-EBS 算法的平均加入时延略小于随机算法的平均加入时延，在其他节点个数下，A-EBS 算法的平均加入时延都是远远小于 NICE 算法和随机算法。随着节点数量的增加，A-EBS 算法和度优先算法的平均加入时延基本保持不变，而且 A-EBS 算法的平均加入时延始终低于度优先算法。NICE 算法和随机算法的平均加入时延随着节点数目的增多而增加，所以相比于 NICE 算法和随机算法，节点数目越多，A-EBS 算法在平均加入时延方面的优势就越大。

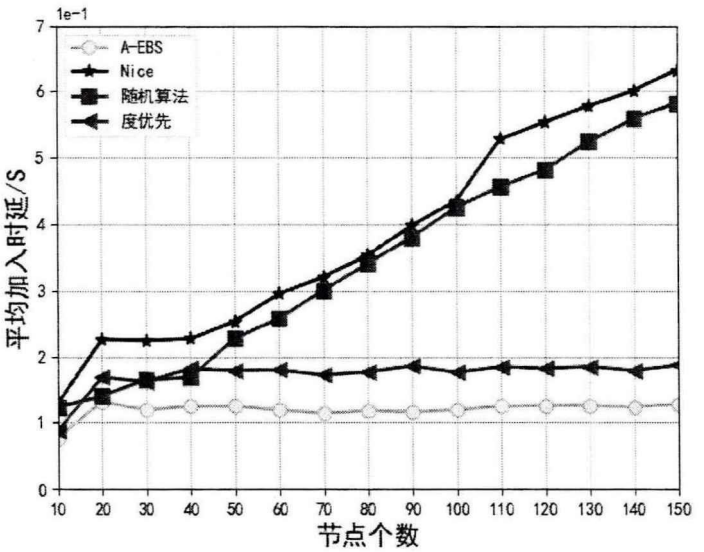

图 3.7 平均加入时延和节点数目关系图

最大加入时延：图 3.8 反映了 A-EBS 算法在最大加入时延方面与 NICE 算法，随机算法和度优先算法的对比情况。从图中我们可以看出，随着节点数目的增加，四种算法的最大加入时延都呈现增长的趋势，NICE 算法和随机算法增长明显。NICE 算法在节点个数从 40 个增长到 50 个时，最大加入时延由 0.81 增加到 1.58，增加幅度巨大。随机算法的最大加入时延同样增长明显，不过增长幅度比 NICE 算法小。在节点个数处于 20 到 40 之间时，随机算法的最大加入时延小于度优先算法，在其他情况下，均大于度优先算法。A-EBS 算法和度优先算法增长缓慢。节点数目由 20 增长到 150 时，度优先算法的最大加入时延从 0.55 增长到 0.59，增长量很小，A-EBS 算法的最大加入时延从 0.41 增长的到 0.42，相比于度优先算法，最大加入时延较小而且增长量更少，所以比度优先算法更有优势。在节点数目为 150 的时候，A-EBS 算法的最大加入时延为 0.42，更是远远小于 NICE 算法的 1.74。

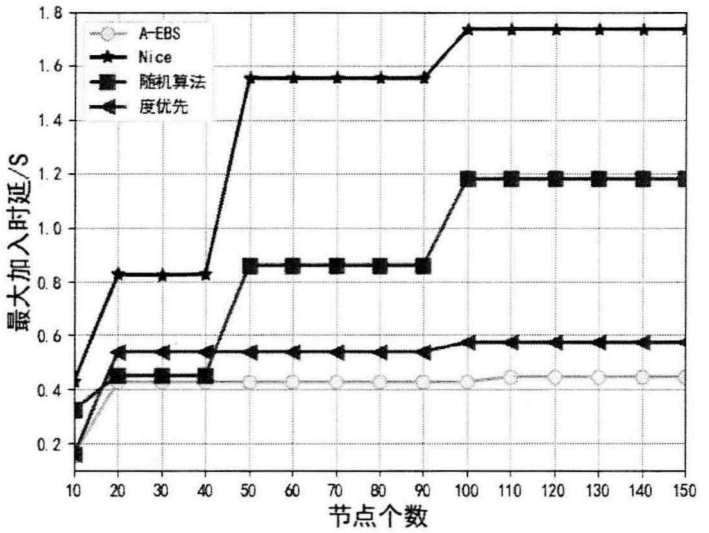

图 3.8 最大加入时延和节点数目关系图

平均延伸链路长度：下图 3.9 反映了 A-EBS 在平均延伸链路长度方面与 NICE 算法，随机算法和度优先算法的对比情况。从图中我们可以看出，在节点个数小于 270 个的时候，A-EBS 算法的平均延伸链路长度小于其他三种算法，节点数目在 270 到 300 之间的时候，A-EBS 算法的平均延伸链路长度小于 NICE 算法和随机算法，大于度优先算法，而在节点数目大于 300 以后，A-EBS 算法的平均延伸链路长度大于其他三种算法，随着节点数目的增多，增长较快。所以 A-EBS 算法在平均延伸链路长度方面较其他三种算法没有优势。

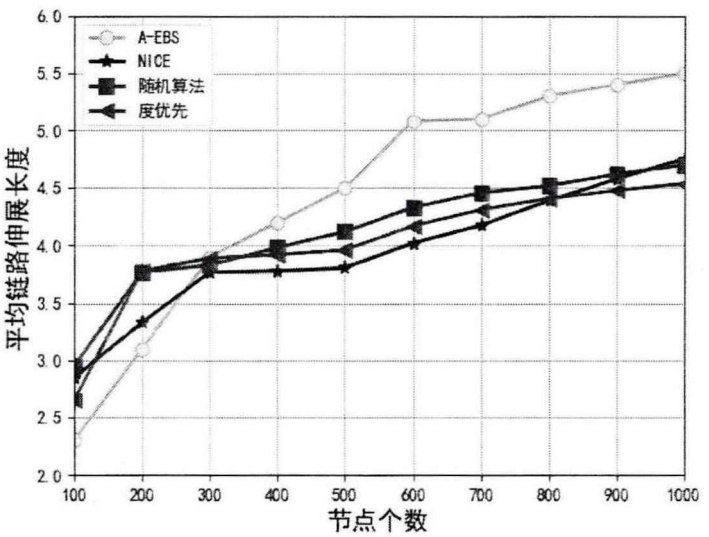

图 3.9 平均延伸链路长度与节点数目关系

3.4 本章小结

本章首先描述了应用层组播的稳定性相关问题，并介绍了近些年来对应用层组播稳定性的相关研究。最后着重介绍所提出的基于效率稳定度的应用层组播算法。对算法的相关定义和过程进行细致说明，并对实验结果进行了分析，表明该算法在降低组播树的平均加入时延和最大加入时延方面有较好的效果，但是在降低平均链路长度方面效果不佳。

第四章 基于 FFMpeg 的流媒体直播系统设计与实现

本系统是基于 FFMpeg 技术开发的一款应用程序，由服务器和客户端两部分组成。实现了流媒体直播的功能，为了简化服务器的功能，同时突出点到点[44]的传输模式，以开启直播的客户端为根节点，与请求观看该直播的客户端形成一棵应用层组播树。服务器不用接收客户端的流媒体信息，也不必向客户端发送流媒体信息。服务器实现的主要功能是对用户进行管理、处理用户请求、构建和维护组播树以及根据当前的组播信息形成相应的组播树。这样做可以减轻服务器的压力，数据流不通过服务器，服务器只作为一个协调决策者。

4.1 基于 FFMpeg 的流媒体直播系统整体设计

本系统的服务器端包含用户管理和组播树管理两个模块，用户管理模块方便管理员对用户的相关操作，组播树模块用于动态的显示当前直播的组播情况。

客户端的主要功能是响应用户的控制指令，通过 Socket 编程，将用户的操作信息发送给服务器。客户机之间通过 RTP 协议和 FFMpeg 编程技术在网络中实时发送流媒体数据。

采用 Java 网络编程技术来实现对整个系统的控制。所有客户机之间的消息传递，都是通过 Socket 类里的相关方法实现的[45]。在发送端，使用 FFMpeg 封装的相关方法，将数据发送出去，数据包是以 RTP 实时数据流的形式传输的。同样通过 FFMpeg 的相关功能实现数据的接收，最后通过 SDL 库的方法将数据流播放[46]。

在系统的上层，当有用户开启直播时，服务器就会为该用户开启一棵应用层组播树，组播树的根节点就是开启直播的节点；当有用户请求观看直播时，服务器会根据第三章介绍的基于效率稳定度算法为请求加入的节点寻找父节点，加入到组播中。在系统的下层，主机使用 FFMpeg 编程技术实现流媒体数据的接收，解压缩，转码播放并发送给其他节点等。

4.2 系统服务端的设计与实现

只有管理员才可以登录服务系统并进行相关的操作，所以在进入服务端之前，要进行身份认证，正确输入用户名和密码，才可以进入。正确输入用户和密码之后，管理员可以进入应用层组播的后台管理系统，包含有用户管理模块和组播树管理模块，如图 4.1 所示。

图 4.1 系统界面图

用户登录模块类图如图 4.2 所示。

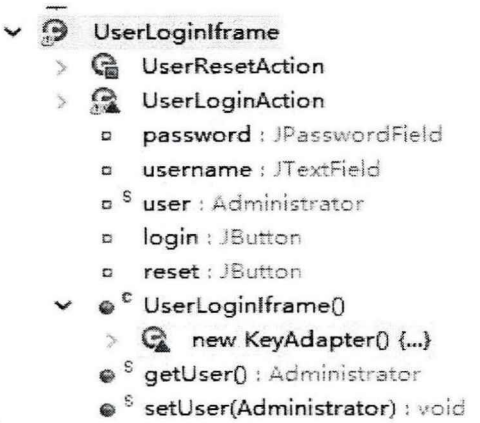

图 4.2 用户登录模块类图

4.2.1 用户管理模块

用户管理模块提供了管理员对数据库中，有关用户信息的增删改查等可视化操作，包括对普通用户的管理和对直播用户的管理。对普通用户的管理，包括添加用户和修改删除用户信息两种操作。对直播用户的管理，包括添加直播用户和修改删除直播用户信息两种操作。用户管理模块结构如图 4.3 所示。

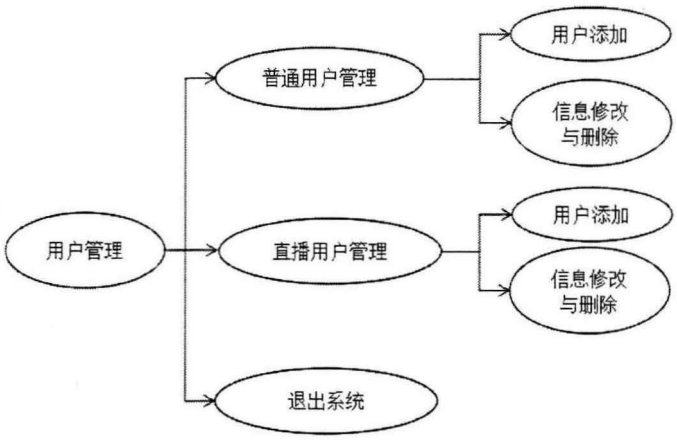

图 4.3 用户管理模块结构图

4.2.2 组播树构造

在本系统中，应用层组播树的构造采用第三章中介绍的基于效率稳定度的应用层组播算法。以开启直播的节点为根节点，有客户机请求观看直播时，服务器寻找合适的节点作为转发节点，并将此节点的信息发送给请求主机，客户主机接收到父节点消息后向父节点发送数据流转发请求。节点加入过程如图 4.4 所示。

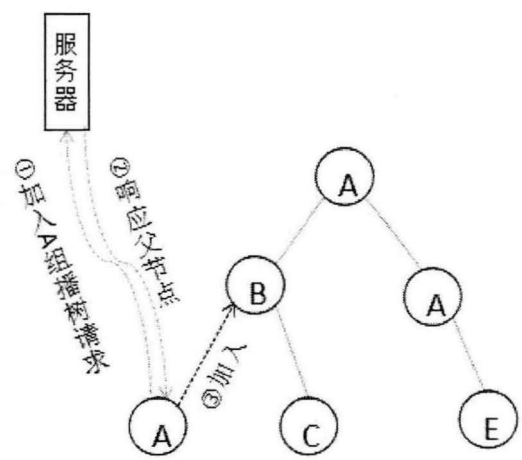

图 4.4 节点加入示意图

4.2.3 组播树维护

在本系统中，所有的节点都是终端设备，根节点也是端主机。端主机最大的缺点就是稳定性差[47]，随时可能离开。如果节点正常的离开，会向父节点和子节点发送离开消息[48]。如果是非正常的离开，就不会向其他节点发送消息。这时候，节点要有一种检测机制，可以检测到节点的离开事件。本系统中采用的是心跳检测机制。在心跳机制中，每个节点会周期性的向它的相邻节点发送心跳报文，如

果节点的一个已知邻居节点，长时间没有发送过来心跳消息，该节点就会认为这个邻居节点已经失效。

由于本系统中的根节点是端节点，所以节点的离开分为四种情况：

（1）非根节点正常离开

如图 4.5 所示，普通节点正常离开时，会首先向服务器发送消息，通知服务器更新组播树。服务器收到离开消息后，在组播树中将该节点删除。然后离开节点会向父节点和子节点都发送离开消息，父节点收到消息后，会停止向该节点发送数据并从邻接表中删除该节点，子节点收到消息后，重新向服务器发送加入组播树的请求。

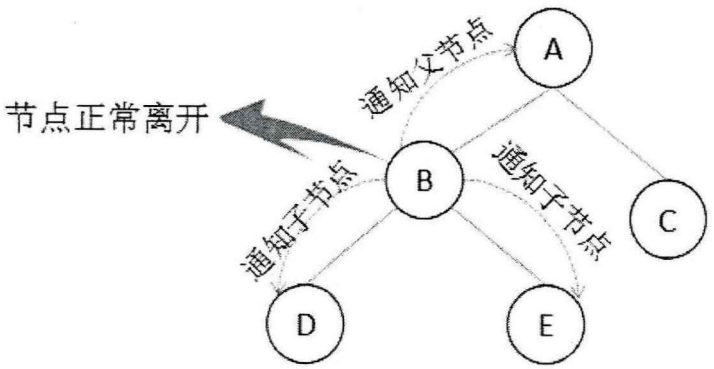

图 4.5 一般节点正常离开

（2）根节点正常离开

如图 4.6 所示，根节点正常离开组播树时，先向服务器发送直播根节点退出消息，向子节点发送离开消息。服务器收到消息，将以该节点为根节点的组播从数据库中删除，然后向其子节点发送退出消息，收到消息的子节点退出组播并向它的子节点发送退出消息。依此方式，直到所有的节点退出。

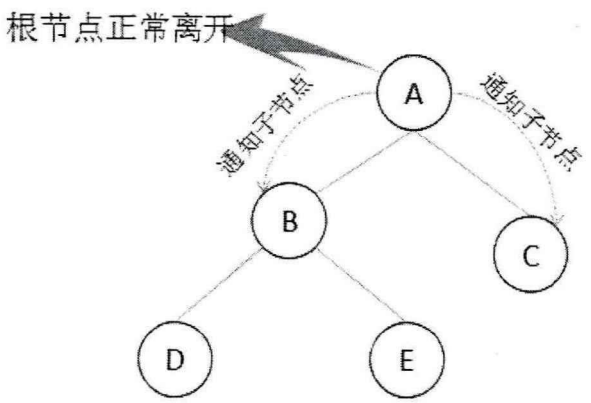

图 4.6 根节点正常离开

（3）非根节点非正常离开

如图 4.7 所示，有节点异常退出时，可由心跳机制检测到。当心跳机制检测到有节点失效时，会向服务器发送节点异常退出消息并请求更新组播树。服务器收到消息后，从组播树中将异常退出的节点删除。异常退出节点的父节点停止向该节点发送消息，子节点请求重新加入组播树。

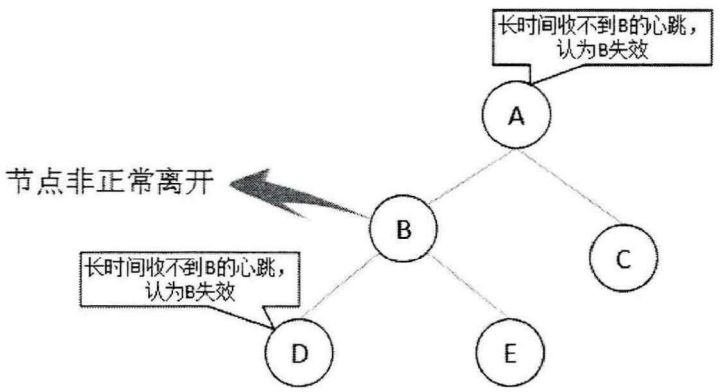

图 4.7 节点非正常离开

（4）根节点非正常离开

如图 4.8 所示，根节点异常退出时，子节点检测到根节点退出，向服务器发送根节点异常离开消息。服务器收到消息后，从数据库中将该直播删除，并向发送根节点异常退出消息的节点发送退出消息。收到退出消息的节点退出组播，并向其子节点发送退出消息。依此方式，直到所有的节点退出。

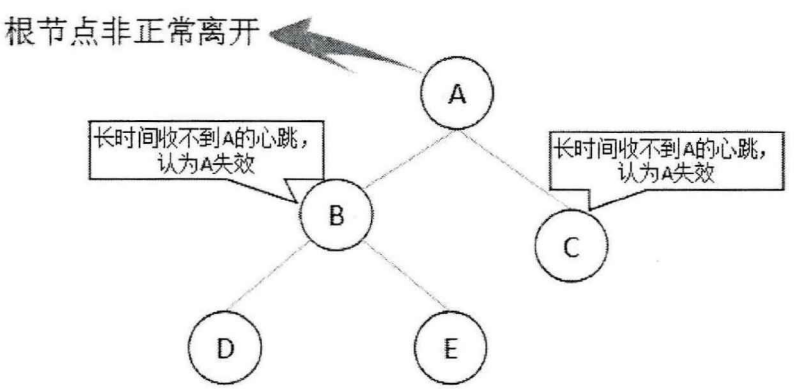

图 4.8 根节点非正常离开

4.2.4 组播树显示

在服务器端的组播树模块中，根据客户端反馈的组播相关信息，服务器会在界面上，动态显示当前网络中的组播树拓扑[49]。

此功能是通过 calculateSubtreeSize(TreeNode n)，calculateLocation（TreeNode

n, int left, int right, int top) 和 drawTree(Graphics2D g, TreeNode n, double px, double py, double yoffs)等函数实现的。calculateSubtreeSize(TreeNode n)函数在请求加入节点加入组播树之前，计算原有组播树的大小。calculateLocation(TreeNode n, int left, int right, int top)函数用来计算要插入节点的位置。drawTree(Graphics2D g, TreeNode n, double px, double py, double yoffs)函数根据之前计算出的节点位置进行画树。节点绘制成椭圆，并可根据内容的长度自动调节椭圆的大小。用直线段将椭圆以树的形式连接起来，椭圆里面画出获取到的主机 IP 地址，最终形成一棵组播树。组播树的显示结果如图 4.9 所示。

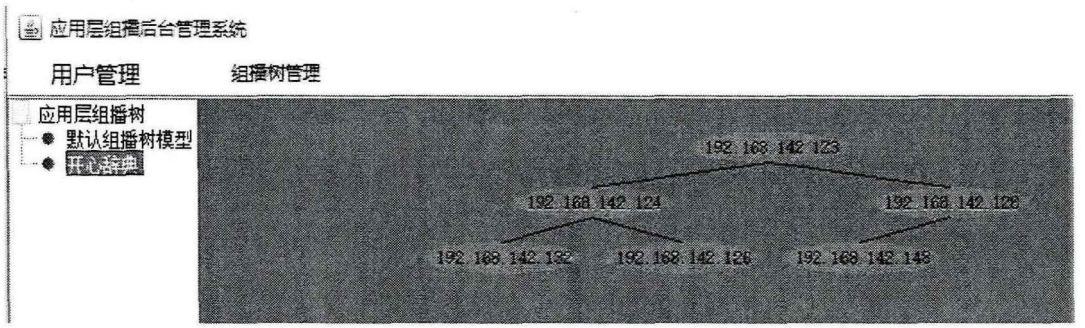

图 4.9 组播树显示示意图

4.3 系统客户端的设计与实现

该系统的客户端，目前主要包括流媒体直播模块。在流媒体直播模块中，直播用户借助摄像头和麦克风采集数据，采集到的视音频数据流数据量很大，在传输之前我们要对数据进行压缩处理，也就是视频编码的过程。之后采用 RTP 协议将视频转发给子节点客户端，接收客户端在接收到数据后进行两部分的操作，第一部分是对接收的数据进行还原处理，也就是解码操作，然后将解码后的数据进行播放，如果解码后得到的数据格式不是 YUV 格式[50]，还要进行转码处理；第二部分是判断是否需要对接收到的数据进行转发处理，如果接收节点还有子节点，则还需要对接收到的数据复制，然后转发给子节点。流媒体直播模块的工作流程如图 4.10 所示。

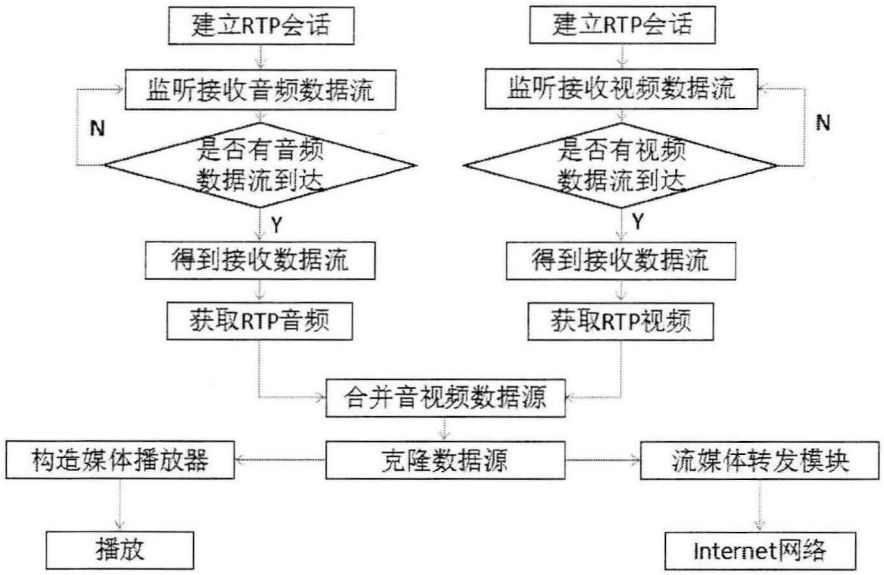

图 4.10 流媒体直播模块工作流程

4.3.1 编码

视频的编码过程用到了在第二章所介绍的很多函数。av_register_all()完成对所有组件的注册工作；avformat_ alloc _output _context2() 初始化一个AVFormatcontext 结构体；avc_new_stream()方法创建输出码流的通道；avcodec_find_ encoder()查找需要的编码器；avcodec_open2() 打开编码器；avformat_write_header()用于写入头文件；avcodec_encode_video2()是该过程中的一个重要方法，用于完成对一帧图像数据的编码工作；av_write_frame()方法将编码后写入 AVPacket 结构体的数据写入输出文件中；av_write_trailer()用于完成文件尾的写入。其编码流程如图 4.11 所示。

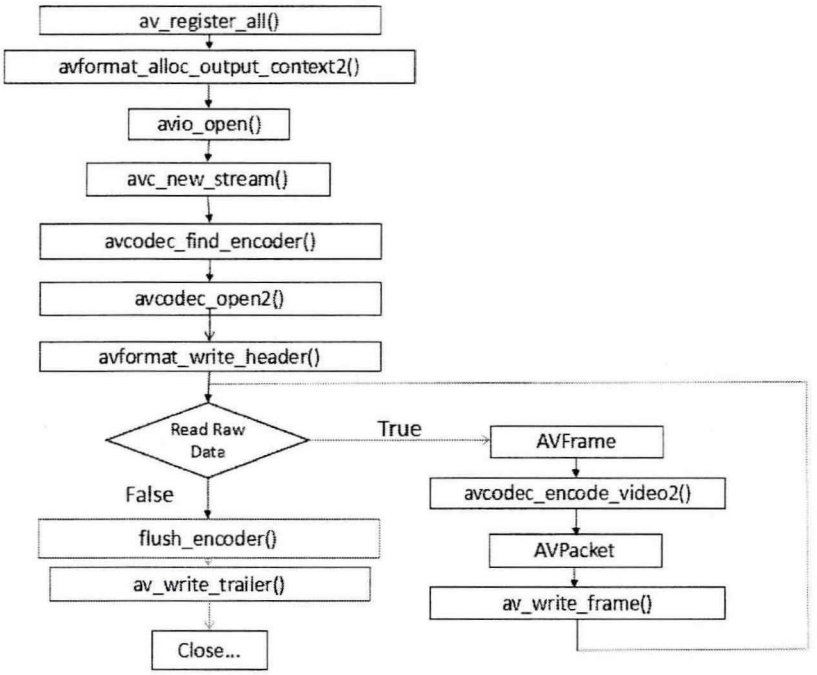

图 4.11 视频编码流程示意图

4.3.2 解码

解码过程是编码过程的逆过程。所使用的方法很多是相同的，最主要的区别是在解码一帧画面时，使用 avcodec_decode _video2()函数。前面也介绍过，AVPacket 结构体里存放的是压缩数据；AVFrame 结构体里存放的是图像本身的完整数据。所以在编码过程中，是将 AVFrame 里的数据进行压缩处理，将编码后的数据存放在 AVPacket 结构体里。在解码过程中，是将 AVPacket 结构体里的数据取出，经过解码过程处理，还原图像数据，在保存到 AVFrame 结构体里。解码流程如图 4.12 所示。

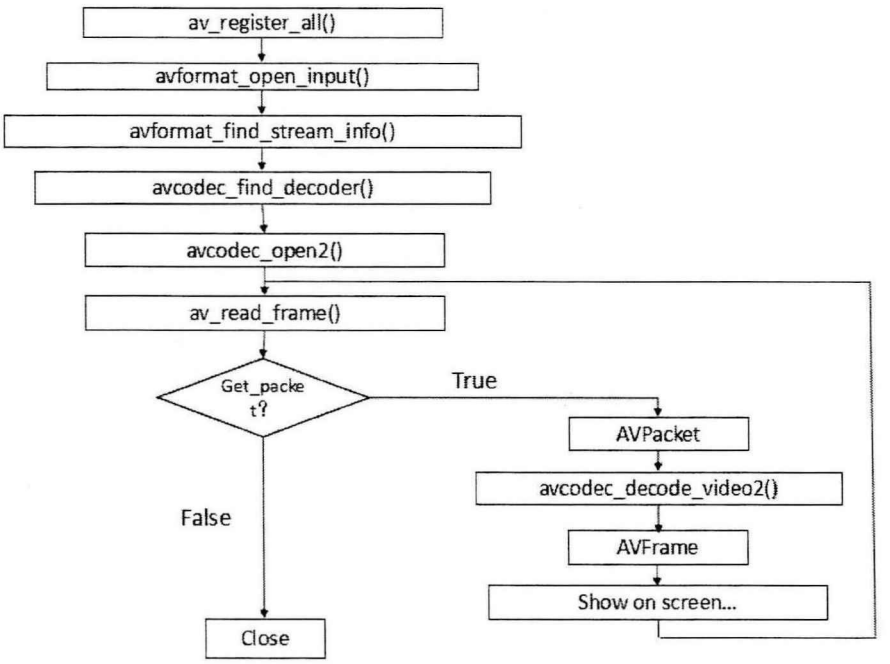

图 4.12 解码流程示意图

4.3.3 播放

在播放阶段，要实现播放功能，需要使用 FFMpeg 编程技术和 SDL 库共同完成。FFMpeg 完成将视频文件转换为 YUV 的视频格式。SDL 是将 YUV 格式的视频渲染到屏幕上。SDL 的播放流程如图 4.13 所示。播放流程主要分为初始化的过程和循环显示画面的过程。

其中初始化过程又分为四步，初始化 SDL、创建显示窗口、在窗口的基础上创建渲染器和创建纹理。所用到的函数有：

SDL_INIT()：该函数用于 SDL 的初始化。

SDL_CreatWindow()：创建一个窗口，用于播放视频，该方法包含有六个参数，分别是窗口的标题、窗口位置 X 坐标、Y 坐标、窗口的宽、高以及窗口的 flags。

SDL_CreatRenderer()：基于上一步形成的窗口来创建渲染器，其中第一个参数为 window，就是所要渲染的窗口。

SDL_CreateTexture()：在渲染器的基础上，创建一个纹理。该方法的五个参数分别表示纹理的目标渲染器、纹理的格式、变化频率及纹理的宽和高。

循环显示流程分为三步，第一是设置纹理数据，第二是将纹理复制给渲染目标，最后将画面显示出来。所用到的函数有：

SDL_UpdateTexture()：设置 Texture 的像素数据；该方法的原型为 int SDLC:\LL SDL_UpdateTexture(SDL_Texture * texture, const SDL_Rect * rect,

const void *pixels, int pitch)。参数 texture 表示目标纹理；该方法返回 0 表示执行成功，返回-1 表示执行失败。

SDL_RenderCopy()：将 texture 数据复制给渲染目标。该方法的原型是 int SDLCALL SDL_RenderCopy(SDL_Renderer * renderer, SDL_Texture * texture, const SDL_Rect * srcrect, const SDL_Rect * dstrect)。参数 renderer 表示渲染目标。同样的，该方法返回 0 是执行成功，返回-1 是执行失败。

SDL_RenderPresent()：用于最后的显示画面。SDL_RenderPresent()原型是 void SDLCALL SDL_RenderPresent(SDL_Renderer * renderer)。参数 renderer 指定渲染的目标。

综上所述，用户在使用 SDL 的时候，会弹出一个播放窗口，这个播放窗口就是 SDL_Window。在显示 YUV 格式的数据时，使用 SDL_Texture。一帧 YUV 格式的数据，就是一个 SDL_Texture。将 SDL_Texture 渲染到 SDL_Window 的过程，使用 SDL_Renderer。使用 SDL_Rect 来决定渲染显示的位置。

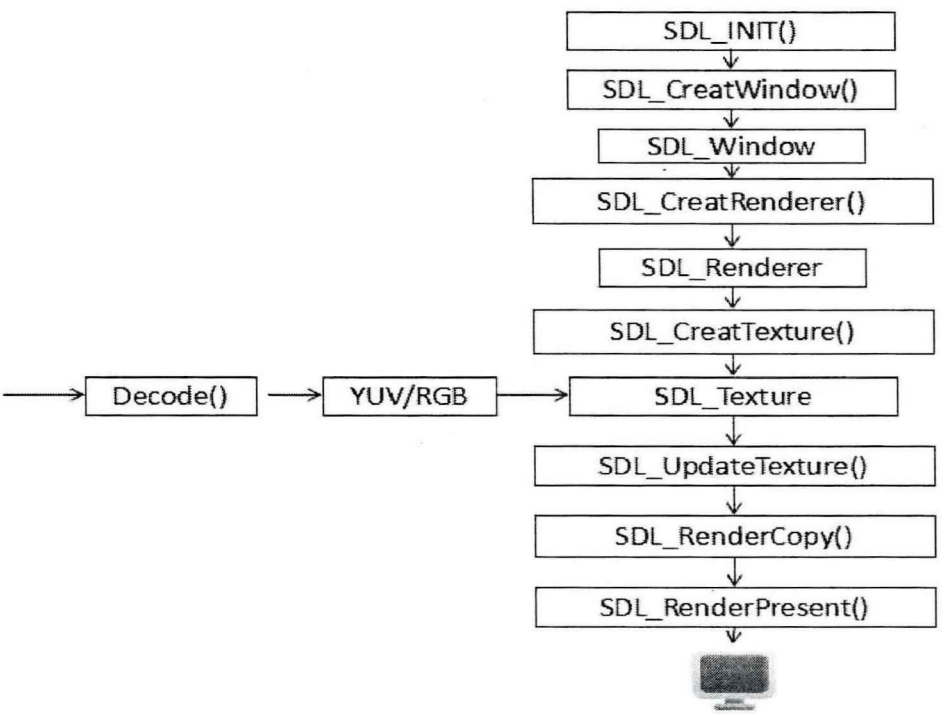

图 4.13 SDL 显示流程图

4.4 本章小结

本章对基于 FFMpeg 的应用层组播流媒体直播系统的整体架构设计做了详细介绍，展示了服务端和客户端所实现的功能以及相关功能界面截图。本章还着

重介绍了在系统的上层，应用层组播树的构造，维护和显示的过程。在客户端部分，介绍了流媒体直播模块的工作流程，以及所涉及到的编解码流程。

第五章 总结与展望

5.1 总结

本文首先分析了 IP 组播和应用层组播，并解释了应用层组播兴起的原因。然后对在基于 FFMpeg 的稳定应用层组播流媒体直播系统中用到的一些技术进行了详细的介绍，尤其对系统中使用到的，FFMpeg 中的相关 API，做了系统的介绍。

首先，通过对应用层组播稳定性的相关研究分析，提出了基于效率稳定度的应用层组播算法，将计算机的 CPU 效率引入到组播树的形成过程中，并综合考虑节点的在线时间，节点的出度，和节点时延这些因素。并将此算法应用到基于 FFMpeg 的流媒体直播系统中。在系统的上层，节点形成组播树的过程中，使用基于效率稳定度的应用层组播算法。

然后，对基于 FFMpeg 的流媒体直播系统的服务端和客户端的设计和实现分别做了详细的介绍。虽然现在该系统在功能上还有所欠缺，但它的优势是从底层开始做起。到目前为止，市场上还没有一款开源的应用层组播框架，所以很难被使用和改变。本系统使用 FFMpeg 编码技术，最底层的视音频传输、接收、解码、编码和转码等都是自己实现的，不受框架的约束，这样使用者可以根据需要随意改变，以后会对系统继续改进和完善。

5.2 展望

由于水平和时间有限，基于 FFMpeg 的应用层组播系统还是一个只具有一些基础功能的系统，还有很多需要完善的地方。以后我会在以下几个方面继续改进该系统：

1、完善系统功能：完善的功能是一个好的系统的基本要素，方便用户使用。目前该系统还有很多功能需要完善。例如客户端用户只能开启直播或者观看直播，不能保存用户之前直播的内容，没有点播功能等。在服务端，也只有对用户和组播树的相关管理，还可以添加上内存监视，捕获设备管理等功能。

2、提高组播树的稳定性：该系统在形成组播树时采用的是基于效率稳定度算法，该算法虽然在减少组播树的平均加入时延和最大加入时延方面效果显著，但是组播树的平均链路长度值较大，需要进一步改进。

3、降低组播拥塞事件的发生：该系统中应该添加一种应用层组播自适应的带宽调节策略，使各节点可以根据当前的带宽，自适应的增加或删除子节点，保

证组播树的平衡，同时降低节点接收数据的延时，来避免应用层组播拥塞事件的发生。

本篇论文日后的工作就是改善应用层组播算法，进一步提高该系统的稳定性和平衡性，同时完善系统功能。

参考文献

[1] 闲云. 组播技术白皮书[J]. 当代通信, 2003(9):28-33.

[2] 李想, 江敏. 浅谈 IP RAN 技术和应用[J]. 中国新通信, 2013(20):88-89.

[3] 张园园, 余镇危. 基于覆盖网络的应用层选播服务的设计与研究[J]. 微电子学与计算机, 2011, 28(12):72-74.

[4] 刘波. IP 组播通信机制及其实现[J]. 计算机工程, 2001, 27(6):131-133.

[5] 吴文单. IP 网络视频传输的负载均衡技术研究[D]. 武汉理工大学, 2010.

[6] Maughan D, Schertler M, Schneider M, et al. Internet Security Association and Key Management Protocol (ISAKMP)[J]. Rfc, 1998:248-253.

[7] Maughan D, Schertler M, Schneider M, et al. Internet Security Association and Key Management Protocol (ISAKMP)[J]. Rfc, 1998:248-253.

[8] 陈波, 陈世平. 应用层组播的研究与算法实现[J]. 计算机工程与设计, 2008, 29(20):5195-5198.

[9] 李晓燕. 基于树优先应用层组播技术的研究与实现[D]. 首都经济贸易大学, 2005.

[10] 冯侦探, 尤佳丽, 倪宏,等. 基于服务质量感知的P2P流媒体直播点播融合系统[J]. 高技术通讯, 2012, 22(3):263-270.

[11] LUO JianGuang, ZHANG Meng, ZHAO Li,等. A Large-Scale Live Video Streaming System Based on P2P Networks 基于 P2P 网络的大规模视频直播系统[J]. 软件学报, 2007, 18(2):391-399.

[12] 刘婷婷. 一种面向 P2P 流媒体的应用层组播协议算法[D]. 北京交通大学, 2009.

[13] Chu Y, Rao S, Seshan S, et al. Enabling conferencing applications on the internet using an overlay muilticast architecture[C]// Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM, 2001:55-67.

[14] Chu Y H, Rao S G, Seshan S, et al. A case for end system multicast[J]. IEEE

Journal on Selected Areas in Communications, 2006, 20(8):1456-1471.

[15] 赵露, 龚隽鹏, 张鹏洲. 应用层组播综述[C]// 全国互联网与音视频广播发展研讨会. 2010.

[16] Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast[J]. Proc Acm Sigcomm Aug, 2002, 32(4):205-217.

[17] Pendarakis D, Shi S, Verma D, et al. ALMI: an application level multicast infrastructure[C]// Conference on Usenix Symposium on Internet Technologies and Systems. USENIX Association, 2001:5-5.

[18] 张海涛. 视频压缩编码研究及应用[D]. 山西大学, 2008.

[19] 沈燕飞, 李锦涛, 朱珍民,等. 高效视频编码[J]. 计算机学报, 2013,

[20] Pan F, Lin X, Rahardja S, et al. Fast mode decision algorithm for intraprediction in H.264/AVC video coding[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2005, 15(7):813-822.

[21] Ugur K, Andersson K, Fuldseth A, et al. High Performance, Low Complexity Video Coding and the Emerging HEVC Standard[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2010, 20(12):1688-1697.
36(11):2340-2355.

[22] Wenger S. H.264/AVC over IP.[J]. IEEE Trans.circuits Syst.video Technol, 2003, 13(7):645-656.

[23] Jain J R, Jain A K. Displacement Measurement and Its Application in Interframe Image Coding[J]. IEEE Transactions on Communications, 1984, 29(12):1799-1808.

[24] LI Junsheng, YU Zhenwei, PAN Yun,等. A Survey of the Application-level Multicast 应用层组播综述*[J]. 计算机应用研究, 2004, 21(11):14-17.

[25] 苏金树, 曹继军, 张博锋. 应用层组播稳定性提高技术综述[J]. 计算机学报, 2009, 32(3):576-590.

[26] 陈爱玲. 基于稳定概率的最小延时应用层组播树构建及恢复算法研究[D]. 华中师范大学, 2016.

[27] 孙永辉, 姜昱明. HTTP 代理服务器的设计与实现[J]. 计算机工程与设计,

2003, 24(7):56-58.

[28] Roca V, Elsayed A. A Host-Based Multicast (HBM) Solution for Group Communications[C]// International Conference on NETWORKING. Springer-Verlag, 2001:610-619.

[29] Goyal V K. Multiple description coding: compression meets the network[J]. Signal Processing Magazine IEEE, 2001, 18(5):74-93.

[30] Banerjee S, Lee S, Bhattacharjee B, et al. Resilient multicast using overlays[C]// ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems. ACM, 2003:102-113.

[31] Chung W J, Chung W J, Youm Y. Inverse kinematics of planar redundant manipulators using virtual link and displacement distribution schemes[C]// IEEE International Conference on Robotics and Automation, 1991. Proceedings. IEEE, 1991:926-932 vol.1.

[32] Yiu W P K, Wong K F S, Chan S H G, et al. Lateral error recovery for media streaming in application-level multicast[J]. IEEE Transactions on Multimedia, 2006, 8(2):219-232.

[33] Bawa M. Transience of Peers and Streaming Media.[J]. 2010.

[34] Lin C S, Syu W T. A fine-grained balancing scheme for improved scalability in P2P streaming[J]. Multimedia Tools & Applications, 2010, 46(1):71-90.

[35] Veloso E, Meira W, Bestavros A, et al. A hierarchical characterization of a live streaming media workload[C]// ACM SIGCOMM Workshop on Internet Measurment. ACM, 2002:117-130.

[36] 廖小飞, 李德敏, 陈光,等. 一种基于树树结构的低时延应用层组播模型[J]. 小型微型计算机系统, 2016, 37(11):2493-2497.

[37] CAO Jia, LU ShiWen, 曹佳,等. A Minimum Delay Spanning Tree Algorithm for the Application-Layer Multicast 应用层组播的最小延迟生成树算法[J]. 软件学报, 2005, 16(10):1766-1773.

[38] Cui J Q, Lai M C, Jiang W B, et al. OverSim:A Scalable Application-Layer

Multicast Network Simulation Framework[J]. Computer Engineering & Science, 2012, 34(10):1-5.

[39] Fei Z, Yang M. Restoring Delivery Tree from Node Failures in Overlay Multicast.[J]. Ieice Transactions on Communications, 2005, 88-B(5):2046-2053.

[40] Fei Z, Yang M. Restoring Delivery Tree from Node Failures in Overlay Multicast.[J]. Ieice Transactions on Communications, 2005, 88-B(5):2046-2053.

[41] 罗建光, 赵黎, 杨士强. 基于用户行为分析的应用层组播树生成算法[J]. 计算机研究与发展, 2006, 43(9):1557-1563.

[42] 贺红, 马绍汉. The General Principles of Randomized Algorithms 随机算法的一般性原理[J]. 计算机科学, 2002, 29(1):90-92.

[43] 崔建群, 陈爱玲, 夏振厂,等. 一种高稳定性低延迟的应用层组播生成树算法[J]. 计算机科学, 2016, 43(6):77-81.

[44] 李倩. 基于 P2P 的流媒体直播系统关键技术研究[D]. 中南大学, 2007.

[45] 史继欣. 基于 P2P 的流媒体直播系统分析[J]. 科技创新与应用, 2015(34):95-95.

[46] 孟成, 陈亚. FFmpeg+SDL 实时播放摄像机视频设计[J]. 产业与科技论坛, 2017, 16(17):57-58.

[47] 霍林, 李德顺, 谭颖璐. 基于节点稳定概率和链路贡献度的应用层组播树生成算法[J]. 计算机研究与发展, 2012, 49(12):2559-2567.

[48] 叶咏佳, 崔建群, 范静,等. 基于 p-tractert 源路径发现技术的应用层组播系统:, CN 102655510 A[P]. 2012.

[49] 贾珂铭. 基于 JMF 的应用层组播流媒体播放系统的研究与实现[D]. 华中师范大学, 2012.

[50] 郭睿, 吕硕, 臧淼. 基于 Matlab 的 YUV 视频流色彩空间变换[J]. 现代信息科技, 2017, 1(4):83-85.

攻读硕士期间参与的科研项目

[1] 2016.9~至今　国家自然科学基金项目面上项目：“移动环境下应用层组播网络模型及稳定性问题的研究”，项目编号：61370108

[2] 2016.9~至今　国家自然科学基金项目面上项目：“移动机会网络中的低能耗节点移动模型及消息转发机制研究”，项目编号：61672257

致 谢

时光荏苒，如白驹过隙。第一天来华师报道的场景就像是在昨天，时间却过了两年，我已经在美丽的华师校园度过了两年的学习生活。记得第一次来华师面试的时候，它没有本科学校的宽敞广阔，却带有一种独特的儒雅韵味。虽然学校的空间不大，绿植却很多，随处可见一人难以环抱的树木，彰显着学校的沉稳与宁静。从第一次来便深深地爱上了这里。回首两年的研究生学习生活，有成长，有心酸，有收获，有留恋，但更多的是感谢。在这即将毕业的时刻，我要向所有教育我，陪伴我，关心我和帮助我的老师、家人、同学、朋友致以衷心的感谢。

首先我要感谢我的导师崔建群老师，崔老师虽然学识渊博，但是很平易近人。第一次看见崔老师，优雅又有气质，感觉完全不像自己想象的那种，有学识的人会不太注重打理自己的生活。崔老师不仅教我很多学术研究方面的知识方法，让我在学术方面有进步提高，更是对我在生活方面有很大的影响。崔老师会教我们如何提高生活质量，为人处世的道理，以最好的状态和精力去学习，这些都让我终身受益。回顾两年的学习生活，随处可见崔老师对我的指导和指正。借此机会，我想对崔老师表达我深深地敬意和感谢。

除此之外，还要特别感谢常亚楠老师和黄枫老师对我在学习中的指导，生活中的关心，所以在这里，同样想对常老师和黄老师表达我的敬意和感谢之情。

感谢计算机学院所有的老师们，谢谢你们不辞辛苦的传授知识，让我拥有更广阔的视角，顺利完成学业。感谢网络与通讯组的所有老师和同学、同门的师兄师姐和同学们，在我学习遇到困难和挫折的时候，谢谢你们毫不计较得失的帮助，让我可以在知识的海洋里更好的遨游，遇到你们很幸运。

还要感谢父母和我的舍友们。感谢你们对我生活上的帮助和支持。正是因为有你们的关爱和支持，我才可以坚定地走每一步。因为有你们，每天都可以感受到生活的温暖。

最后衷心感谢评审专家们能够在百忙之中抽出您宝贵的时间审阅我的论文，感谢各位评委老师的指正和赐教。