基于 P2P 的 QoS 保证的流媒体分发服务的研究

计算机软件与理论

硕 士 生：蔡致暖

指导教师：林小拉 教授

摘 要

近年来，流媒体在 Internet 上得到了迅猛的发展，成为推动未来宽带应用的主动力。然而，传统的流媒体分发方案如 C/S 模式、CDN、IP 组播等，在系统的可扩展性、可靠性和经济性等方面存在瓶颈，不能满足大规模流媒体分发服务的需求。基于 P2P 网络的流媒体分发技术，将流媒体数据分发的中心从服务器转向众多普通的节点，在不改变现有网络配置的前提下具有很高的性价比，同时具有良好的可扩展性和鲁棒性，是一种具有广泛应用前景的流媒体分发方案。

本文主要研究 P2P 流媒体分发服务的相关理论和技术，在研究分析了几种典型的 P2P 流媒体分发模型的基础上，提出了一种 QoS 保证的 P2P 流媒体分发模型——QCast 模型。QCast 模型是基于单组播树拓扑模型的，同时结合了网状拓扑模型的多邻居节点结构，这样既具有单组播树拓扑模型中组播树控制方式简单的优点，又具有网状拓扑模型中鲁棒性高的优点。

QCast 是基于分布式哈希表（DHT）的应用层组播模型，在模型的设计上充分利用了 DHT 的路由、容错等机制。在 QCast 模型中，通过采用一种 QoS 保证的节点 ID 分配算法，考虑了网络中节点的异构性，使得组播树的构建过程具有“QoS 感知”（QoS-aware）的特点；同时，在节点的加入算法上也考虑了 QoS 因素，通过计算优先级函数来挑选节点的数据发送节点。多邻居节点结构的采用，减少了因单个节点退出系统而造成的数据传输延迟，提高了系统的鲁棒性。针对这种多数据发送者的结构，在节点的数据传输上，QCast 模型提出了推拉结合的数据传输调度算法，该算法能够提高系统中节点的数据下载带宽，保证流媒体分发的连续性和稳定性。

此外，本文还实现了 QCast 模型的系统原型，并针对模型中的主要性能指标设计了仿真实验，实验结果表明了 QCast 模型对系统的服务质量保证、数据传输

延时、鲁棒性、可扩展性等性能进行了优化，并且在这些性能指标之间做出了较好的平衡。

关键词：**P2P**，流媒体，**QoS**，分布式哈希表

Research on QoS-aware Streaming Media Dissemination Based on P2P Technology

Computer Software and Theory

Name: Zhinuan Cai

Supervisor: Professor Xiaola Lin

Abstract

Media streaming over the Internet is booming nowadays. It is becoming the main driving force for future applications in broadband network. However, traditional systems of streaming media broadcast, such as C/S mode, CDN, IP multicast, incur capacity bottleneck regarding to scalability, reliability and cost. It is difficult for them to adapt to large-scale application. The P2P media streaming, however, can distribute data from the center of streaming media server to host nodes without changing the current deployment of Internet. It is cost-effective, scalable, and robust, and contains good application value in future.

This thesis is to study the theories and technologies of P2P streaming media broadcast service. In terms of analyzing the current several typical models of the P2P streaming media broadcast, a QoS-aware P2P streaming media broadcast model, *QCast model*, is proposed. This model is based on tree topology model while using multiple neighbors of mesh topology model to transmit data, therefore it is easy to control the multicast tree and also achieves good robustness.

QCast is a multicast model in application level based on *distributed hashing table* (DHT). It can easily exploit the DHT's routing and failure recovery functions to organize the multicast tree. In QCast model, an ID assignment method based on QoS requirements of nodes is used to take the heterogeneities of nodes into account, which enforces QoS-aware multicast tree construction. Also, the QoS requirements of nodes are considered when a node joining the system. The joining node chooses data sending nodes by computing nodes' priority of the selection strategy. Using multiple

nodes to transmit data in QCast reduces the rate of withdrawal from the individual node and the transmission delay, and improves the system's robustness. A push-pull streaming method to fetch data from neighbors is also presented. This method can improve nodes' download bandwidth, thus enhancing the continuity and stability during the streaming dissemination.

In addition, we have developed a prototype system of QCast model. Simulation experiments are also conducted to evaluate the performance of the proposed system. Simulation results show that the system's QoS guarantee, data transmission delay, robustness and scalability have been optimized in the proposed QCast model.

Keywords: P2P, media streaming, QoS, DHT

论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：蔡致暖

日期：2008 年 5 月 8 日

学位论文使用授权声明

本人完全了解中山大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版，有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆、院系资料室被查阅，有权将学位论文的内容编入有关数据库进行检索，可以采用复印、缩印或其他方法保存学位论文。

学位论文作者签名：蔡致暖　　导师签名：

日期：2008年 5月 8 日　　日期：2008年 5月 8日

第1章 绪论

近年来，随着计算机技术、压缩技术以及网络技术的发展，网络中的流媒体业务也得到了飞速的发展和应用。其广泛应用在诸如远程教学、网络直播、宽带网视频点播、视频会议等领域。流媒体正在成为推动未来宽带应用的主动力。因此，对流媒体分发技术的研究具有十分重要的意义。

1.1 研究背景

随着Internet技术的不断发展，计算机技术已经进入了“以网络为中心计算”的时代。当前，Internet正在向高速率骨干网和高速率接入网进步，宽带网的迅猛发展，为用户提供了更高的传输带宽和多媒体实时数据传送的能力。同时，个人计算机处理能力也在不断增强，人们已经不再满足于Internet上传统的网页浏览、文件下载、在线文字聊天等应用，流媒体以其特有的娱乐性和交互性正成为推动未来Internet发展的主要动力。

下面一组数据可以看出流媒体在Internet上的飞速发展：国外相关调查表明，约有51%的上网用户经常使用流媒体，流媒体业务正变得日益流行[1]；在2000年，中国网络电视用户约为1万户，这一数据在2004年增长到219万户[2]，而国家广电总局科委副主任杜百川预测我国的网络电视用户到了2008年可能达到2000万户；Multimedia研究机构预测全球网络电视收入将从2004年的52亿元增长至2008年的592亿元。因此，流媒体系统具有广阔的市场前景，而其中作为网络电视核心的流媒体直播技术必然也有着良好的发展前景。

流媒体的广阔市场前景也推动着流媒体服务的基础环节（服务器性能、网络带宽、编解码及传输技术等）飞速发展，但由于流媒体分发具有传输实时性要求高，带宽资源消耗大和传输过程持续时间长等特点，若采用传统的C/S架构模式，服务器必须通过网络给每个客户端发送多个相同的内容，随着客户端数目的增加，必然导致I/O负载压力大、可扩展性差和系统部署成本高等问题。因此，在高并发的服务请求条件下，基于C/S架构模式的流媒体系统所提供的服务质量（Quality of Service, QoS）难以得到保证，将导致较低的用户播放体验（QoE）。

为了解决传统的基于C/S架构模式的流媒体系统服务能力有限、不能适应大规模流媒体应用的问题，学术界和工业界提出了代理服务器、IP组播、内容分发网络等技术。采用IP组播技术[3][4]来进行媒体数据分发，可以保证网络上只有唯一的数据包在进行传输，每个客户端都能接收到这个数据包，这极大地减轻了服务器的带宽需求，能够有效降低服务器和网络的负载，但IP组播在可靠传输、拥塞控制、安全性、网络和终端系统的异构性等方面尚存在问题，提出多年之后仍然难以在Internet上大规模部署[5]。近年来内容分发网络CDN（Content Delivery Network）[6]在Internet上得到了广泛的部署，它通过在Internet“边缘”部署缓存代理服务器，将媒体内容推送到距离用户更近的网络“边缘”节点上，从而降低中心内容服务器和骨干网络的负载压力，改善了用户的使用体验。然而，这只是部分改善了系统的可扩展性，由于CDN的“边缘”分发节点采用的仍然是传统的C/S架构模式，这导致了整个CDN系统能够支持的并发用户数仍然与系统的部署成本投入呈线性增长关系。为提供大规模的商用流媒体服务，运营商需投资大量的服务器硬件设备和网络带宽，从而产生高昂的部署成本。

综上所述，流媒体分发系统的广泛应用必须提高其在可扩展性（Scalability）、鲁棒性（Robust）、服务质量保证（QoS）等方面的性能。而传统的流媒体分发技术和方案在系统的可扩展性、可靠性和经济性等方面均未能满足大规模流媒体分发服务的需求。因此，针对大规模高并发的流媒体分发服务，如何提供具有高可扩展性和QoS保证的流媒体分发系统，并降低系统的部署成本，已经成为当前流媒体分发技术研究中的一个重要课题。

最近几年来，对等计算（Peer-to-Peer Computing，或P2P计算）技术引起了越来越多人的关注。相对于传统的C/S模式，P2P模式一个非常显著的特点就是节点无需依赖中心服务器资源，每个节点同时扮演客户机和服务器的双重角色，即每个节点在利用其他节点的资源的同时也为其他节点提供服务。这一特性使得P2P系统的服务能力随着用户数的增加而自然增长，具有“与生俱来”的可扩展性，这就解决了传统C/S架构模式下服务器过载和资源瓶颈等问题。同时，P2P系统采用节点自组织的方式工作，能够很好地适应节点的随机加入和退出，因而在容错性、数据高可用性等方面具有不可替代的优势。

P2P技术在文件共享等应用领域中已经取得了很大的成效，鉴于P2P技术良

好的可扩展性，把P2P技术应用到流媒体分发领域同样能带来革命性的突破，因此，P2PStreaming技术被提了出来。基于P2P网络构建流媒体分发系统，可以从根本上解决大规模流媒体分发服务所必须具备的高可扩展性、高容错性等要求，同时通过将服务器的负载分散到P2P网络中的每个节点，可以显著降低服务器硬件和网络带宽投入成本，具有非常高的经济价值。

1.2 P2P 分发技术的发展及挑战

1.2.1 P2P 分发技术的发展

目前，学术界和工业界对对等网络（Peer-to-Peer Network）还没有一个标准的定义。IBM对P2P的定义是：P2P系统由若干互联协作的计算机构成，且至少具有如下特征之一：系统依存于边缘化（非中央式服务器）设备的主动协作，每个成员直接从其他成员而不是从服务器的参与中受益；系统中的成员同时扮演服务器与客户端的角色；系统中的用户能够意识到彼此的存在，构成一个虚拟或实际的群体。P2P最根本的思想在于网络中的每个节点（Peer）既是资源或服务的请求者，同时又是资源或服务的提供者，即扮演客户端和服务器的双重身份。P2P网络中每一个节点所拥有的权利和义务都是对等的，包括通信、服务和资源消耗。

Napster[7]是最早出现的P2P应用系统，提供MP3音乐文件共享服务，自1999年9月开通后，短短几个月内就吸引了超过2000万用户。由于版权问题，Napster被迫于2001年3月关闭，但这并未阻止它引发起一场互联网上的革命。在Napster之后，以Gnutella[8]为主的P2P系统不断发展壮大，新系统如KaZaA[9]、FreeNet[10]、Morpheus[11]、BitTorrent[12]等不断涌现，P2P用户数量也持续快速增长。到目前为止，P2P技术已经从第1代的P2P网络Gnutella, FreeNet等发展到第2代的Pastry[13], Chord[14]、CAN[15]等乃至第3代的Viceroy[16]；而P2P的思想和技术也迅速从文件共享领域拓展到诸如分布式计算、协同工作、分布式存储及应用层组播等领域。

将基于P2P的应用层组播技术引入到流媒体服务中，是指在不改变Internet现有基础设施的条件下把组播的功能从网络层转移到应用层，把P2P技术应用到流媒体，每个流媒体用户也是P2P网络中的一个节点，这就充分利用了以往被忽略的端节点的资源（CPU、存储、网络带宽等），使得每个端节点既作为客户端又

作为服务器，使得流媒体数据的分发分散化，从而减轻服务器的I/O负载和网络带宽需求。为构建大规模流媒体分发服务提供了一种低成本的解决方案。

1.2.2 P2P网络基础架构

P2P网络基础架构主要有三种：中心目录服务器模型、非结构化模型和结构化模型。

1. 中心目录服务器模型

顾名思义，这种模型具有类似C/S架构的中心目录服务器，所有节点都必须与中心目录服务器建立连接，由中心目录服务器负责索引它们上面的内容。在内容定位上，首先节点向中心目录服务器发出请求，中心目录服务器根据请求返回符合要求的节点，然后文件交换就直接在请求节点和返回节点之间进行。这种模型仍然具有C/S架构的缺点，当网络中节点的数量增多时，服务器端的存储和带宽等成为限制。Napster的实现是这种模型的代表。

2. 非结构化模型

这种模型完全没有了索引服务器的概念。它采用一种称为“洪泛”方式的定位机制：每个节点广播请求消息给所有和它直接相连的节点，如果这些节点都没有所请求的内容，则继续广播这个请求消息给所有和它们直接相连的节点，这个过程一直持续到找到所请求的内容或者广播的次数超过了某个值为止。该模型中的“洪泛”定位方式会消耗很大的带宽，因此可扩展性较差。然而这种方式非常有效，并且可以通过设置请求消息中的TTL参数、缓存搜索过的路径等方式来改善性能。代表系统有Gnutella，KaZaA等。

3. 结构化模型

结构化模型是基于分布式哈希表（Distributed Hash Table，DHT）技术的一种P2P网络模型。网络中的每个节点和对象都会被赋予一个标识（ID），并且每个节点维护部分其它节点的信息。当文件被发布到网络中时，根据该文件的名字和内容，用某种hash算法（如SHA1）生成一个文件ID，然后由DHT把该文件路由给网络中节点ID最接近文件ID的节点，所有参与路由的节点同时保存一份该文件的拷贝。当某个节点请求某个文件时，请求就会被转发给节点ID最接近所请求文件ID的节点，然后文件就可以被路由给请求节点。相比于中心目录服务器模型，

这种模型实际上是把索引的功能分散到了网络中的每个节点，是一种纯正的P2P系统；相比于非结构化模型，采用DHT的定位机制不存在过度消耗带宽的问题，可扩展性好。Pastry、Chord和CAN等是这种模型的代表。

1.2.3 P2P流媒体分发技术面临的挑战

与P2P文件共享等应用相比，将P2P技术应用于流媒体分发服务的研究所面临的挑战更加严峻，这是因为P2P网络和流媒体技术本身具有的特殊性，例如Internet上众多提供服务的Peer节点的服务能力有限，不同的Peer节点具有不同的服务能力，Peer节点加入和离开系统具有随机性，流媒体带宽资源占用高，流媒体对数据的播放有较为严格的时限和顺序要求等。归纳起来，基于P2P的流媒体分发技术主要面临如下几个方面的挑战：

1. Peer节点的定位机制

在P2P流媒体系统中，Peer节点一般需要请求从其它Peer节点获取流媒体数据，因此，新节点请求加入系统时首先需要搜索定位能为其提供服务的Peer节点，而在节点加入系统后，一旦提供服务的Peer节点离开或失效，也需要重新搜索定位新的Peer服务节点。一种最直观的解决方法是采用中心目录服务器模型，引入中心索引服务器来记录并维护系统中所有Peer节点的状态信息，Peer节点在加入或重新加入系统时也直接从中心索引服务器上获取Peer服务节点。这种方式虽然简单易行，但至少有两点缺陷：一是中心索引服务器容易成为系统瓶颈。当用户数量达到十万甚至百万规模时，中心索引服务器本身就可能被与Peer节点状态维护相关的信息所淹没；二是中心索引服务器容易成为系统的单点失效节点，从而削弱系统的鲁棒性。因此在大规模的P2P流媒体应用环境下如何建立有效的Peer节点搜索定位机制，是P2P流媒体分发技术研究中所面临的第一个挑战；

2. Peer节点的退出检测与处理

在P2P网络中，Peer节点是自组织的，它们可以随时加入系统，也可以随时离开系统，或因为发生故障而失效，这会导致系统中其它部分节点数据传输服务中断。这种Peer节点的离开或失效行为在今天的Internet环境下是不可避免且可能频繁发生的，因此，如何从P2P流媒体分发系统体系设计的角度出发来避免或减少Peer节点离开或失效行为的影响，或者建立某种类型的快速反应机制来减少其

它节点服务被中断的时间，成为P2P流媒体分发技术研究中所面临的又一挑战；

3. QoS服务质量保证

当前流行的各种视频压缩标准产生的视频流码率都比较高，如MPEG-1的码率为1.5Mbps, MPEG-2的码率为2-10Mbps, MPEG-4的码率约为几十Kbps到几Mbps，超过或低于此码率都会导致解码时缓冲区发生上溢或下溢，影响播放质量，因此在流媒体数据的分发过程中必须尽量保证数据传输的实时性。然而数据包在网络传输过程中往往会发生丢包、延迟等现象，这些都会影响接收方的QoS服务质量。此外，P2P网络中Peer节点之间的网络带宽资源有限，Peer可能随时离开或失效，这些因素将导致P2P网络环境下流媒体分发服务的QoS服务质量问题更加突出。因此在P2P流媒体分发服务中如何对Peer节点的QoS服务质量提供保障已成为P2P流媒体分发技术研究中的核心问题；

4. Peer节点的异构性处理

P2P网络中存在众多的Peer节点，这些Peer节点所具备的网络带宽资源、主机处理能力（CPU、存储能力等）实际上是有差异的；并且不同Peer节点对流媒体播放的质量也有不同的需求，这种质量差异往往用分辨率、帧率等指标来度量。因此，不同的Peer节点在对数据的接收处理能力和对外服务能力等方面存在异构性。在基于P2P的流媒体分发服务中，如何适应并利用Peer节点之间的这些异构性，也是P2P流媒体分发技术研究中的挑战之一；

5. 激励机制及其他

在P2P系统中，所有Peer节点都是平等的，它们既是客户端同时又是服务器，这种对等模式使得P2P系统具有诸如自组织、高可扩展性等特征。然而在现实中，P2P系统中的节点往往更多地表现出自兴趣（self-interest）和理性（rationality），个体节点的目标往往是最大化自身的网络效用（Network Utility），这就导致了P2P网络中“搭便车”（free-riding）问题，即系统中绝大多数节点并不贡献资源，整个网络的运行只是依赖于少量节点的无私奉献。Adar等人对Gnutella网络的用户行为做了研究[17]，研究发现，70%的Gnutella用户并不共享任何文件，接近50%的查询命中仅来自1%的Gnutella用户。这种“搭便车”的行为导致了Internet空闲资源的利用率较低，不能充分发挥P2P模式的优势。因此如何建立一套有效的激励机制，使得Peer节点在行使消费者角色的同时，主动承担为其它节点提供服务

的职责，也是P2P流媒体分发技术中一项值得研究的课题。此外，如何评估共享资源的真实性和可靠性，如何有效地检测并抵御恶意Peer节点所实施的攻击或欺骗等行为，这些都是P2P流媒体分发技术研究中需要解决的问题[18]。

本文的研究主要针对上述挑战中的前四点，在研究过程中假设所有Peer节点都是无私的，即并不涉及对第五点提及的相关内容的研究。

1.3 相关研究现状

如何为大规模并发用户建立一个高效稳定的 P2P 流媒体分发系统已经成为当今研究的热点。其中，基于 P2P 技术的应用层组播方法，可以在不改变 Internet 现有基础设施的条件下在应用层实现组播。虽然它的组播效率相对 IP 组播方法低，但是其良好的可扩展性和经济性使得其在近年来得到广泛关注。1998 年出现的 Webcast 是最早利用 P2P 技术实现大规模流媒体点播和直播的系统，它主要利用一棵二叉组播树在用户之间进行实时多媒体数据的传输和共享。此后由于 P2P 流媒体直播服务相对容易实现，首先得到快速发展。2000 年出现的 ESM（End System Multicast）[19]，是第一套 P2P 流媒体直播系统，它标志着 P2P 流媒体直播技术进入了系统发展期。ESM 系统采用网状拓扑结构进行互连构造最优媒体数据组播树的方法在用户间传播实时的多媒体内容。由于算法限制，这套系统只能扩展到几千人同时在线。此后，各种 P2P 流媒体直播系统、高度可扩展的应用层组播协议大量涌现。其中典型的系统有 Stanford 大学的 Peercast[20]和德国的 P2PRadin，而应用层组播协议有马里兰大学的 NICE[21]、思科的 Overcast[22]、微软的 CoopNet[23]和 SplitStream[24]、伯克利大学的 Gossip[25]等。这些系统和协议为 P2P 流媒体分发技术的发展打下了坚实的理论基础。2004 年 5 月欧洲杯期间，香港中文大学的张欣研博士开发出了 Coolstreaming[26]原型系统并在 Planetlab 网上试用获得成功。

在基于P2P技术的应用层组播的各种流媒体分发服务方案中，分别存在按覆盖网络拓扑结构、按播放类型、按有无预部署服务节点的基础设施这三种类型划分。其中第一种划分存在单组播树拓扑、多组播树拓扑和网状拓扑三种拓扑结构；第二种划分包括了直播和点播两种不同的流媒体播放模式；第三种划分则存在有无预部署节点的两种部署模式。下面将分别按照这三种不同的分类方法对现有的

基于P2P的流媒体分发技术的研究状况进行论述和分析。

1.3.1 P2P 流媒体分发系统的拓扑结构

目前，P2PStreaming分发模型可以分为三类：基于单组播树拓扑的协议、基于多组播树拓扑的协议和基于网状拓扑的协议。

1. 基于单组播树拓扑的数据分发

基于单组播树拓扑的协议把节点组织成一棵应用层组播树，数据的传输方式通常采用“推”（PUSH）的机制，即由树的父节点负责为子节点传送数据。当父节点退出或失效时，它的子结点将暂时得不到数据，服务质量受到影响，这时就要求系统尽快重建连接。早期的流媒体分发系统多采用这种结构，典型代表有ESM[19]、Peercast[20]、NICE[21]、ZIGZAG[27]等。

基于单组播树的方案控制方式相对简单，可以达到很好的数据传输效率，研究的主要问题包括：如何设计组播树构造方法和组播协议，以达到特定的性能指标要求；如何增强系统的容错性，即如何减少节点离开或失效行为对其它节点的影响。此外，基于单组播树拓扑的协议还存在一个难题：所有的叶结点只接收数据而没有对系统做出贡献，这会造成大量节点能力的浪费和系统的不公平。

2. 基于多组播树拓扑的数据分发

微软研究院的CoopNet[23]和SplitStream[24]采用的是基于多组播树拓扑的方案。这两个模型都利用了多描述编码(Multiple Description Coding，MDC)，构建以源节点为根节点的多棵组播树。采用MDC编码后的媒体流，分成多个层分别同时沿多个组播树进行传输，每个节点从它参与的多棵组播树上获取数据，然后再将各层数据整合，解码还原成可以播放的媒体数据。当某棵组播树上的父节点离开或失效时，只会导致一条MDC子流的传输被中断，节点还可以从其它组播树上继续接收其它的MDC子流，由于MDC流在解码时不存在依赖关系，某些层的数据缺失并不会造成媒体数据无法播放，只会影响其播放质量，这在很多情况下是用户可以接受的。

多组播树模型研究的目标主要是如何实现将一个节点放在不同组播树的多个位置上，这些位置可以是随机选择或者是采用某种确定性算法来实现。此外，节点带宽资源异构、路由节点选择等问题也是多组播树模型研究的内容。

相比单组播树方案，多组播树方案可以充分利用系统中每个节点的带宽资源，采用MDC编码使系统能够较好地适应节点的动态性和节点之间带宽的抖动，具有良好的容错性；但多组播树的维护开销大于单组播树，且维护操作相对复杂。

3. 基于网状拓扑的数据分发

在这种分发模型中，节点首先自组织形成一个网状的控制拓扑，节点之间可能存在一条或多条连接路径，然后根据协议选取一个边的子集计算一棵数据分发树，用以传输数据。网状拓扑模型的研究重点关注的是怎样提高、优化覆盖网的效率，然后利用已有的算法构建数据分发树。较为典型的系统包括CoolStreaming[26]、GridMedia[28]、Bullet[29]。它们的主要思想均为利用Gossip协议来构造一个网状随机拓扑结构。在传输过程中，每个节点动态地和其他节点交换本地缓存的数据视图，并根据播放进度和数据视图向邻居节点发起数据请求，然后从多个节点并行接收媒体数据。这种以存储转发为基础，先获取对方节点所拥有的数据状态信息再发起请求的传输方式就是“拉”（PULL）的机制。结果对于每个节点都形成一个多对多的数据传输模型。

与单组播树和多组播树的方案相比，基于网状拓扑的分发方案使得节点可以从任何相邻的节点获取自身需要的数据，具有更好的鲁棒性，在较高节点扰动（Churn）的情况下，对节点播放质量的影响相对较小，但是系统存在与上层传输覆盖网络的拓扑不匹配、需要较大缓存、启动延迟比较大等问题。

1.3.2 P2P流媒体的直播和点播

P2P流媒体分发技术按播放类型可以分为直播和点播。

1. P2P流媒体直播

P2P流媒体直播是指基于应用层组播的有同步时序要求的流媒体技术。在P2P直播系统中，节目内容首先被压缩、编码，然后由直播源节点推送到由众多Peer节点组成的覆盖网络中，通过在线的Peer节点对媒体数据进行中继传输，接收节点收到编码数据之后再进行解码观看。现有的P2P直播系统包括CoolStreaming[26]、GridMedia[28]、PROMISE[30]、PRO[31]等。衡量P2P直播系统的重要指标有同步丢失延时、启动播放延时和抖动延时。同步丢失是指在经过网络传输和Peer节点中继传输后，媒体内容的视频帧在多个Peer节点播出时出现不同

步的情况。同步丢失延时就是在发生同步丢失的情况下，Peer节点播放初始视频帧的时间相对于该帧被视频源输出的延时。在物理相邻节点出现视频内容不同步时，此类延时要求尽量减小。启动播放延时主要是指Peer节点启动播放操作之后到节目开始播放之间的延时。而抖动延时主要是指因邻居节点失效、退出和网络拥塞等原因导致节目播放失败后，再次恢复播放所需要的时间。以上三种延时对用户的直播体验有较大的影响。

P2P直播通常采用的拓扑结构包含了单组播树、多组播树和网状三种拓扑模型。在基于单组播树的拓扑结构中，每棵组播树上的节点共享从源服务器一个频道所流出的数据，每个节点只参与到一棵组播树中，这种数据分发方式在服务提供者和服务消费者数目关系上属一对多模式。它存在叶节点带宽资源得不到有效利用、中间节点的离开或失效行为对子节点的影响较大、QoS难以保障等问题。在基于多组播树的数据分发方案中，数据流在源服务器端被分割成多条子流，每条子流用一棵单独的组播树进行传输，每个Peer节点可以根据自己的带宽资源情况加入到单棵或多棵组播树中。这种直播系统播放质量高，但存在网络拓扑维护代价高和操作复杂等问题。在基于网状拓扑的数据分发方案中，新节点在加入时随机从现有系统中挑选出多个Peer节点作为其邻居节点，并通过运行相关的数据调度算法来从多个邻居节点同时获取数据。由于节点可以同时从多个节点获取数据，因此可较好的解决节点扰动（Churn）对系统造成的影响，系统的播放质量和鲁棒性较高。现在流行的P2P直播系统大多采用这种方式。然而，这种系统也存在网络拓扑不匹配、需要比较大的缓存、启动延迟比较大等问题。

2. P2P流媒体点播

P2P流媒体点播也是基于应用层组播的流媒体技术，相对于直播系统，它具有更强的异步时序特征。在P2P流媒体点播系统中，Peer节点之间播放的视频时序并不要求与视频源或其它节点的视频帧同步，而只是与用户播放操作行为相关，用户对播放的进度可以自由控制。例如暂停、快进和快退等操作功能，此类操作提供了更贴近用户需求的操作自由度。文献[32]的P2Cast就是典型的P2P点播方案。在P2P点播系统中，如何对用户异步播放操作进行快速响应是急需解决的核心问题。由于用户的异步操作使得播放进程具有独立的播放时序，这就导致了Peer节点之间的播放时序重合度低，在缓存空间有限的条件下，本地缓存对邻居

节点可用的概率也会随之降低，从而相对直播而言Peer节点更难得到需要的媒体数据。对此问题，学者们通常采用的方法就是优化Peer节点媒体数据的缓存和搜索算法。通过对缓存算法的优化，提高本地缓存数据对邻居节点的可用度；通过优化搜索算法，提高节点搜索到所需要的媒体数据的速度。此外，对于P2P点播技术的异步请求要求系统具有快速响应特性，在普遍存在节点扰动和网络状态不稳定的情况下，多数点播方案采用了网状拓扑结构来进行数据传输。

1.3.3 基于基础设施的 P2P 流媒体分发技术

在基于基础设施的P2P流媒体分发方案中，需要事先在网络上部署一定数量的组播服务节点MSN（Multicast Service Nodes），这些MSN节点在流媒体数据分发的过程中构成核心组播分发树，而每个Peer节点必须先连接到某个MSN节点上，才能接收到流媒体数据。这种由MSN节点组成的核心传输网络被称为覆盖组播网络（Overlay Multicast Network，OMN）。通常，MSN节点配置为服务器，并预先部署在有较高出口带宽的机房中。相比普通的Peer节点，MSN节点具有更高的处理能力、更高的网络带宽、更好的稳定性和安全性，并可以同时为若干个Peer节点提供服务。相比完全基于纯Peer节点的P2P流媒体服务平台，基于基础设施的P2P流媒体分发方案虽然在服务器和网络上付出了更大的成本代价，但可以为大规模并发用户提供更高QoS保障的流媒体分发服务；同时，在相同的大规模用户数情况下，相比基于C/S架构模式的流媒体服务系统更具有单用户服务成本优势。

Banerjee等提出的OMNI[33]和Xu等在文献[34]中提出的P2P点播系统是典型的基于基础设施的P2P流媒体分发方案。这些方案大多注重MSN节点拓扑结构的构建及优化，并且仅停留在系统原型层面。而针对具有高QoS保障需求的P2P流媒体分发服务，它们无论是在MSN拓扑结构的构建和优化上，还是在普通Peer节点之间流媒体数据调度、缓存等的设计上都比较欠缺，尚不能形成一个可具体实施的解决方案。

1.4 论文研究内容

根据上述研究现状，可知当前P2P流媒体分发服务的关键技术基本可以划分为两类，第一类是与数据分发体系（或者说数据分发系统的拓扑结构）相关，包括了Peer节点组织成何种类型的网络拓扑、新节点如何加入系统、如何处理节点的离开或失效、在源服务器端对流媒体内容采用何种类型的编码、系统是否具有良好的可扩展性等；第二类是与数据分发调度相关，主要是指如何把流媒体数据从发送节点分发调度到接收节点，特别是在多对单的数据传输模式中，如何以协同的方式把流媒体数据从多个发送节点调度传输到单个接收节点，以最大化满足接收节点的QoS需求，或兼顾满足其它的系统目标。本文着重研究的是P2P技术在流媒体直播传输中的应用。在深入研究了典型的P2P流媒体分发模型的基础上，提出了一种QoS保证的P2P流媒体分发模型，其中所展开的研究既涉及数据分发体系，也涉及数据分发调度相关的算法，主要研究内容包括以下几点：

1. 研究了如何基于分布式哈希表（DHT）和应用层组播（Application Level Multicast）来实现对直播的流媒体内容进行数据分发。通过采用基于DHT的P2P网络结构，简单、高效地实现了节点间的信息交互，建立起一个自组织、高扩展性、高容错性的覆盖网。其主要目标是让新节点能够快速有效的加入系统，而当节点离开或失效时，其子节点能够快速准确地找到新的父节点，并尽量保证子节点在此中断过程中流媒体内容的完整接收；同时，通过在应用程序层进行流媒体内容的分发，不需要IP组播或其他类似的技术支持。
2. 基于分布式哈希表，从P2P网络的拓扑结构出发，研究了如何在基于P2P的流媒体分发体系中提供QoS保证。首先针对Peer节点的异构性（带宽、CPU处理能力、存储能力、服务质量要求等）设计一种新颖的节点ID分配算法，保证了离根节点越近的节点具有越高的QoS能力，使得组播树的构建过程具有“QoS感知”的性质；其次，设计一种简单有效的节点加入算法，该算法综合考虑了节点的QoS能力和传输延迟等因素，通过优先级函数来选择节点的父节点和邻居节点，从而在拓扑结构上保证了流媒体数据传输的实时性和连续性。
3. 从数据分发调度的角度出发，研究了如何以协同的方式从多个Peer服务节点向单个Peer接收节点传输流媒体数据，以最大化满足Peer接收节点的QoS

要求。主要研究了在多对单的数据传输模式下，一种以接收方为驱动的数据分发调度策略。在研究过程中同时考虑了 Peer 节点之间在网络传输过程中的丢包、延迟等特性。

4. 在开源的分布式哈希表项目 FreePastry[35]的基础上，设计了一个高质量的 P2P 流媒体分发系统 QCast，包括所有核心算法的实现和系统的架构设计，以展示基于 DHT 和应用层组播技术的 P2P 流媒体分发系统的实现流程。

本论文研究的内容包括了P2P流媒体分发技术研究中的节点搜索和加入、节点离开或失效处理、QoS服务质量保证、节点异构性处理等挑战。

1.5 论文组织结构

本文由五个章节构成，论文的结构安排如下：

第一章为绪论，首先介绍了课题的研究背景和相关研究现状，包括流媒体技术在Internet上的发展、传统流媒体分发技术存在的问题、P2P流媒体分发技术的发展和研究现状，分析了P2P技术在流媒体分发服务中的必要性和关键作用；最后介绍了本文的主要研究内容。

第二章研究了三种典型的P2P流媒体分发模型，并对这三种模型进行比较分析，总结了不同模型的优缺点和典型的P2P流媒体分发模型所追求的性能指标。

第三章提出了一个QoS保证的P2P流媒体分发模型QCast，并从数据分发体系构成和数据分发调度策略出发详细论述了QCast的设计和QoS保证的机制。

第四章实现了QCast的原型系统，包括系统框架设计和核心算法的实现，并针对QCast的主要性能指标设计了仿真实验，对QCast模型进行性能评估。

第五章对论文进行了总结，并展望了研究前景和进一步努力的方向。

第2章　现有 P2PStreaming 典型模型分析

本章主要研究了目前国内外在 P2PStreaming 领域内的主要成果，并针对基于单组播树、基于多组播树、基于网状三种拓扑结构各选取一种典型的 P2P 流媒体分发系统进行深入的分析和比较。

2.1 概述

目前，P2PStreaming 分发模型包括了基于单组播树拓扑的模型、基于多组播树拓扑的模型和基于网状拓扑的模型。

在基于单组播树拓扑的模型中，首先要解决的问题是组播树的构建，最简单的模型是 Peercast[20]。在 Peercast 中，节点的加入和离开策略都很简单，但也容易导致树的不平衡。当父节点退出或失效时，将导致子孙节点服务被中断的时间较长。对此，Banerjee 等提出了一种分层结构的分布式自适应组播树构造协议 NICE[21]，它将端节点组织到不同的簇中，并在簇间形成层次结构的数据转发和拓扑管理体系，其贡献在于改善了单组播树的可扩展性，并将协议负载控制在很低的程度，但仍存在高层节点负载过重的问题。ZIGZAG[27]在 NICE 的基础上将簇的管理和数据转发功能分开，由不同的节点负责完成，降低了高层节点的负载，进一步提高了系统的可靠性和可扩展性。

在 P2P 分发系统中，无法预期的用户行为导致任何节点在任何时候都有可能退出系统。对于单组播树模型，每个节点都只有一个父节点，子节点的服务质量依赖于父节点，一旦父节点离开或失效，子节点则需要被重新加入到组播树中。系统对树的恢复速度将严重影响这部分子节点的服务质量。为了解决这个问题，PROMISE[30]模型在树结构中预先为每一个节点确定若干个备用父节点。一旦父节点离开，子节点可以通过备用父节点迅速恢复。

基于单组播树拓扑的模型采用单发送者（single-sender）方式，对节点间端到端带宽的要求很高，并对节点动态性极其敏感。为此出现了基于多组播树拓扑的模型，典型模型为 CoopNet[23]和 SplitStream[24]。这两种模型都应用了多描述编码（MDC），每个组播树组播一个描述，节点把接收到的所有描述进行叠加以提

高视频质量。

无论是单组播树还是多组播树模型，都显式定义了节点之间的关系，数据总是从根节点开始，按照定义好的路径进行分发。在基于 Gossip 协议的网状拓扑模型中，采用的是多发送者（multi-senders）和数据驱动的方式，节点随机地给系统中的部分节点发送消息，每个接收到消息的节点又继续向其它一些节点发送消息，重复这个过程直到所有节点都接收到这个消息为止。这个过程中节点需要动态地和其它节点交换数据。典型的系统有 CoolStreaming[26] 、GridMedia[28]等。

本章剩余部分将详细介绍并分析三种典型的 P2P 流媒体分发系统模型。

2.2 典型的 P2P 流媒体分发系统

从 2000 年开始陆续出现了很多基于 P2P 的流媒体分发系统，尤其是 P2P 流媒体直播系统，其中 Stanford 大学的 Peercast、微软研究院的 SplitStream 以及香港中文大学张欣研博士等设计的 CoolStreaming 分别是基于单组播树拓扑、基于多组播树拓扑和基于网状拓扑的典型代表。

2.2.1 Peercast

Peercast 是 Stanford 大学 P2P 研究小组的研究成果。Peercast 采用典型的单组播树结构，通过建立一个以媒体源为根节点的应用层组播树来实现流媒体数据的分发。为了优化节点间的传输延迟，Peercast 引入了节点饱和（Saturated）和重定向（Redirect）的概念，来保证组播树中各个节点的负载平衡。如图 2-1 所示，系统中所有节点在逻辑上组成一棵组播树，流媒体数据按照树的拓扑结构由根节点 Root 开始从上至下进行分发。

1. 节点加入算法

Peercast 中，新节点加入系统是一个从源服务器（即根节点）出发沿组播树进行搜索，直到发现某个非饱和节点的过程。首先新节点向源服务器请求服务，如果源服务器有足够的资源，则允许新节点作为它的子节点加入，否则源服务器将这个请求转发给它的某个直接子节点。这个子节点又根据自己的资源情况判断是否允许新节点作为它的子节点加入，以此类推，直到为新节点找到一个父节点。

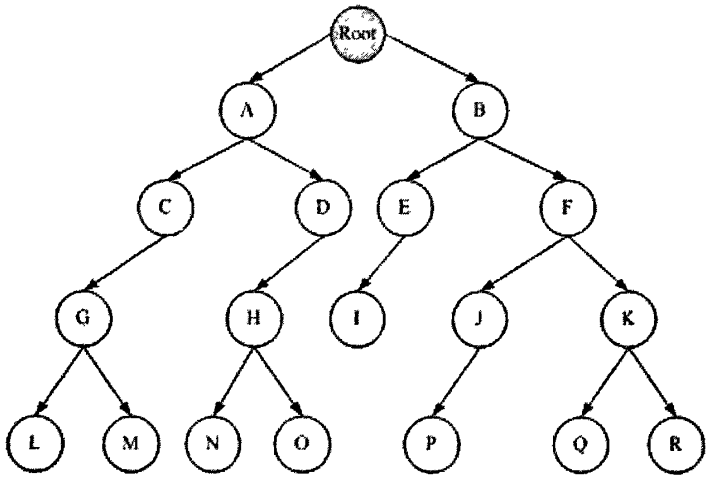

图 2-1 Peercast 的单组播树结构

在这个模型中，每个节点仅维护自己的父节点和直接子节点的信息。如果被请求节点没有足够的资源，则需要把请求重定向到其他节点，Peercast 给出了四种重定向策略：①随机选择；②Round-Robin；③根据物理位置选择（Smart-Placement）；④根据带宽选择（Smart-Bandwidth）。

2. 节点离开算法

节点的离开包括节点主动离开和节点失效两种情况。当节点主动离开系统时，首先需要向其父节点进行注销，并向其每个子节点发送一条重定向消息，以便让它们直接加入离开节点的父节点或从根节点重新开始加入搜索过程。为了检测节点的失效，在所有父子节点之间维持心跳功能，如果某个节点在某段时间内都没有收到子节点的心跳信号，则认为该子节点已经失效，从而回收服务该子节点的资源；同样，如果子节点在某段时间内都没有收到父节点的心跳信号，则认为父节点已经失效，并重新发起加入搜索过程。

3. 数据传输策略

在 Peercast 中，节点加入和离开时，需要通过不停的建立新的连接才能获得媒体数据，所以节点的动态变化对媒体播放质量有很大的影响。对于低码率的应用，Peercast 通过缓存较多的数据，尽量保证节点在重新连接期间可以正常播放。Peercast 的数据传输主要采用推的机制，同时引入了存储转发的机制使得节点有更多的时间去调整组播树的结构，从而达到较好的播放质量。

2.2.2 SplitStream

SplitStream 的关键思想是通过 MDC 编码，把媒体数据分成 k 个 stripes，对每个 stripe 使用一棵独立的树进行组播。Peer 节点希望接收多少个 stripe，它们就可以加入到多少棵组播树中，同时可以指定它们愿意转发的 stripe 数目的上界。SplitStream 所面临的挑战就是构建这样一个由内部节点正交的组播树构成的森林。所谓内部节点正交，是指一个节点最多在一棵树中作为内部节点（非叶子节点），而在其他树中都是叶子节点。SplitStream 还保证该节点所指定的带宽限制条件能够得到满足。这样就保证了转发负载被分布到所有参与节点上。

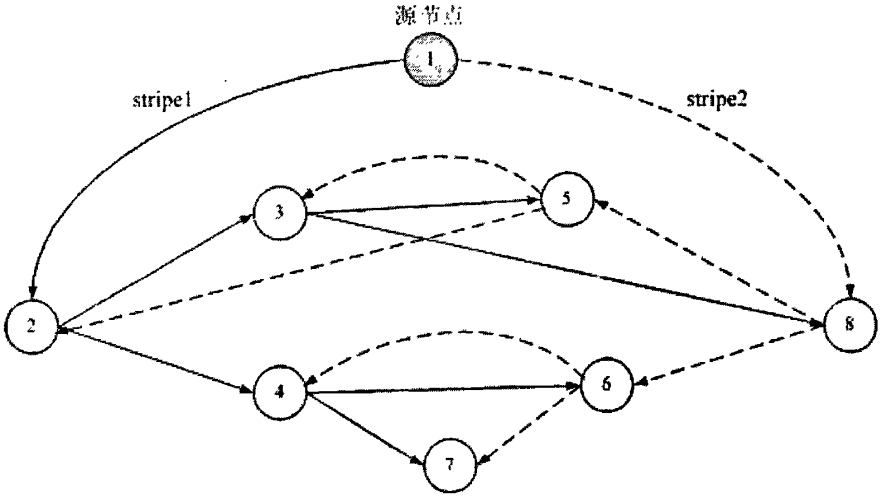

图 2-2 SplitStream 的多组播树结构

SplitStream 的多组播树结构如图 2-2 所示。源节点 1 产生两个 stripes。stripe1 到达节点 2 后，组播树先将它发送到节点 3 和 4，然后发送到节点 5，6，7 和 8。stripe2 到达节点 8 后，组播树先将它发送到节点 5 和 6，再发送到节点 2，3，4 和 7。在这个过程中，假设节点 3 失效退出系统，结果仅影响到节点 5 和 8。因此，多组播树结构充分发挥了每个节点的转发能力，降低了对单个节点的依赖，同时可以减少数据包丢失对播放质量的影响，增强了系统的容错性。

1. 底层架构

SplitStream 的实现依赖于 Pastry 和 Scribe。Pastry[13]和 Chord[14]、CAN[15]等都是完全结构化的、可扩展的、自组织的 P2P 基础架构。Scribe[36]是构建在 Pastry

之上的应用层组通讯系统。

Pastry中，每个节点和对象都随机分配了一个128-bit的标识，分别称为nodeId和Key。所有可用的标识形成一个环状的标识空间。发送消息时节点需要指定一个Key，Pastry通过最长前缀匹配算法把消息路由给nodeId数值与Key最接近的那个节点。图2-3显示了Key为e46a1c的消息从节点75b2fc开始在Pastry网络中路由的过程，该消息最后被成功路由到节点e46875，因为该节点的nodeId在标识空间中是与Key数值最接近的。在Pastry系统中，节点标识是在节点加入系统时随机分配的。这种随机性导致了标识空间中位置相邻（nodeId最接近）的任意两个节点在处理能力、网络配置等方面都可能存在很大的差异。

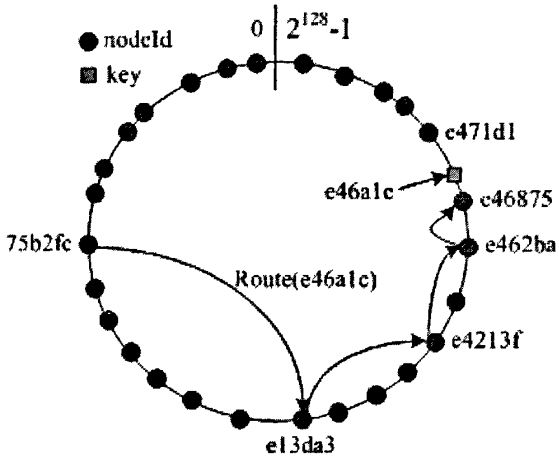

图2-3 Pastry的消息路由：从节点75b2fc开始路由消息e46a1c

Scribe中，每个组用一个Pastry的Key作为groupId，把nodeId与groupId最接近的那个节点作为组的根节点，组的其他成员通过Pastry的路由机制建立到groupId的组播树。最后，组的消息由根节点以相反的路径传输给各个成员。

2. 组播树的构建

SplitStream的目标就是构造多棵组播树，并使得每个节点只在某棵树中作为内部节点，而在其它的树中都为叶子节点。满足这个条件的多棵树称为interior-node-disjoint trees。因为Pastry的路由原则是寻找nodeId数值与Key有最长前缀匹配的节点，在Scribe中，组播树的构建是使所有的成员都路由到groupId，因此整个路由经过的所有中间节点必然与groupId拥有相同的前缀。因此，只要

保证 groupId 最前面的数不相同，就可以保证不同的组的中间节点不相交。

3. 限制节点的出度

为了限制节点的出度，SplitStream 还应用了 Scribe 内建的“push-down”方法。当一个已达到最大出度的节点收到另外一个节点的加入请求时，就把它的当前孩子节点列表发送给后者，后者接着寻求一个有最小延迟的未饱和节点成为它的父亲节点。这个过程在组播树中从上到下递归进行，直到满足条件的节点被选择为止。而且，为了充分利用一些强结点的空闲带宽，SplitStream 把这些强结点组织成一个独立的组，称为能力空闲组。所有孩子节点的数目少于它能转发的 stripe 数目的节点都属于这个组。当一个转发请求得不到提供时，SplitStream 将在能力空闲组中寻找一个能满足要求的节点。

2.2.3 CoolStreaming

CoolStreaming 是一个实用的基于 Gossip 协议的应用层组播系统。Gossip 协议是经典的分布式消息扩散机制，是一种大规模的组通信方式。它采用随机策略选择发送节点，需要根据网络规模确定消息扩散轮次 i 和每轮的消息扩散数量 k。在 CoolStreaming 模型中，每个节点采用 Gossip 协议来维护系统中其他部分节点的数据视图，并通过一定的调度算法在节点之间交换数据。这种系统通常需要较大的缓存，系统延迟相对也较大，但是由于每个节点的数据来源并不依赖于某个特定的父节点，系统具有更好的容错性和稳定性。

1. 节点的加入与管理

CoolStreaming 中每个节点有一个唯一的标识，比如 IP 地址，并且有一个维护系统中其它成员信息的列表 mCache。当新节点 A 加入时，首先与源节点联系，源节点从它的 mCache 中随机地选择一个节点 P 作为新节点的代理节点，A 再与 P 联系，将 P 的 mCache 内容存入自己的 mCache 中，并与自己 mCache 中的节点建立连接。加入系统后，每个节点利用 Gossip 协议周期性地生成一个消息来报告自己的存在。该消息格式为<seq_num，id，num_partner，time_to_live>，其中，seq_num 是消息的序列号，id 是节点标识，num_partner 表示节点的伙伴数，time_to_live 表明消息生存的有效期。节点接收到这样的消息后，根据 seq_num 判断是否为新消息，如果是，则根据 id 更新 mCache 中相关节点的信息，如果

mCache 中没有该 id，则创建一新项。mCache 中每项的格式为<seq_num，id，num_partner，time_to_live，last_update_time>，其中 last_update_time 表示最后一次更新的时间。节点将周期性地检查 mCache 中的每一项，计算该项的剩余生存时间：time_to_live = current_local_time – last_update_time，如果剩余生存时间等于或小于 0 时，说明此节点信息已经过期，将其从 mCache 中删除，并且不再转发。节点可以根据需要从 mCache 中选择节点并与它们建立连接形成一个覆盖网。

2. 数据调度策略

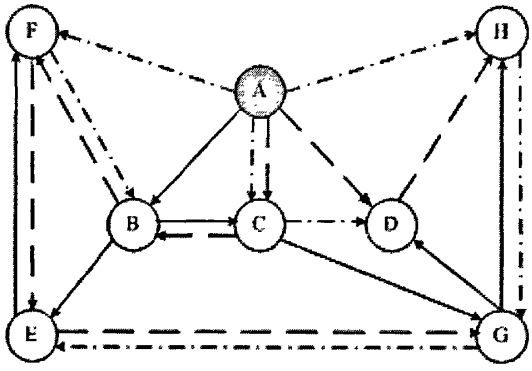

图 2-4 CoolStreaming 的节点关系和数据传输路径示意图

CoolStreaming 中节点的关系和数据的传输路径如图 2-4 所示。A 为源节点，实箭头为 seg1 的传输路径，点箭头为 seg2 的传输路径，虚箭头为 seg3 的传输路径。在 CoolStreaming 中，邻居节点之间的连接可能是双向的，并且任何一个 seg 的传输路径不构成环。实际上每一个 seg 的传输路径还是一棵树。但这棵树的形成和基于组播树的模型是不同的。这棵树预先并不存在，节点之间通过交换彼此的缓冲区视图确定伙伴节点是否包含自己需要的数据，然后通过调度算法从某个伙伴节点处获取数据，从而使自己成为那个节点的子节点；在自己取得数据后，又能为别的节点提供数据，而成为别的节点的父节点。只有在数据被分发后，数据分发树才能显现出来。实际上，除源节点外，所有节点都是主动向它的伙伴节点请求数据的，而不像基于组播树模型那样节点永远都是被动地接收从父节点发来的数据。这种模式称为数据驱动的模式（Data-driven Mode）。

在 CoolStreaming 中，伙伴关系和数据传输方向都是动态的。流媒体数据被分割成大小相同的数据块，用一个缓存映射表（Buffer Map，BM）来表示缓存

中的数据块是否存在。每个节点不断地和它的伙伴节点交换缓存映射表的内容，然后将自己的缓存中缺失的数据从声明拥有该数据的伙伴节点处请求过来，以补全自己缓存中的数据。在一个动态且异构的网络中，为了从伙伴中获取希望的数据块，最优的调度算法必须满足两个约束条件：(1)每个数据块需要在播放的 Deadline 之前获取；(2)每个伙伴的输出带宽具有异构性。如果第一个约束条件不能完全满足，那么应该使超过 Deadline 的数据块尽量少。这是一个 NP 问题，目前无法找到一个能够快速适应高动态网络的最佳方法。CoolStreaming 采用了一个快速响应的启发式算法：节点首先获取只有一个提供者的数据块，再获取有两个提供者的数据块，依次类推。在获取有多个提供者的数据块时，在保证时间限制的情况下，选择带宽最大的提供者。

3. 故障恢复与伙伴的优化

对于节点的正常离开或者失效退出，如果一个节点在一段时间内没有收到伙伴节点的缓存映射表或者在两者的 TCP 连接中没有收到数据，则认为该节点已经离开。此外，CoolStreaming 还采取了一些机制来加强系统的健壮性。节点在正常离开前，会预先发送一条离开的消息。在突然退出的情况下，如果一个节点检测到它的离开，将利用 Gossip 协议散播节点离开的消息。在节点突然退出的情况下，离开的消息可能被不同的伙伴产生，节点只有在第一次收到离开消息时才转发该消息，其他的丢弃。收到消息的每个节点都会更新它的 mCache。

最后介绍一下 CoolStreaming 的伙伴优化策略。在 CoolStreaming 中，每个节点会定期地从 mCache 中随机选择的一些节点建立新的伙伴关系。这个操作能够帮助节点在伙伴离开的情况下维护一个稳定的伙伴数目，同时能够使节点发现性能更好的伙伴节点。在 CoolStreaming 的实现中，每个节点 i 会为它的伙伴 j 计算一个分数 max(Sij，Sji)。其中 Sij 表示单位时间内节点 i 从节点 j 平均获取的数据片断数。因为每个节点既是提供者，也是接收者，所以考虑了两个方向的分数。在找到新的伙伴后，原有伙伴中分数最低的节点将从伙伴列表中删除。

2.3 现有系统的分析比较

从上节和前面几节的介绍可知，这几种典型的 P2P 流媒体分发模型在系统的设计上都围绕着一个中心问题：在流媒体传输的网络带宽、数据传输延时和系统

的鲁棒性之间寻找某种平衡。其中，网络带宽和数据传输延时关系到流媒体播放时的连续性和播放质量，这两个是关系到用户体验的主要性能指标。在流媒体直播系统中，应该尽量提高各个节点的下载带宽，减小各个节点接收媒体数据的延时。系统的鲁棒性是系统应对变化时能否保持正常工作的一个特性。具备良好的鲁棒性是系统应用于大规模流媒体分发服务的必要条件。

表2-1给出了上述三个典型系统的比较数据。

表2-1 典型P2P流媒体分发系统的比较

系统	支持带宽	数据交换	异构支持	质量改善	拓扑结构
Peercast	较低	PUSH	无	无	单树
SplitStream	较高	PUSH	无	多描述编码	多树
CoolStreaming	高	PULL	支持	多路径	网

在这三种P2PStreaming的典型模型中，基于单组播树拓扑的Peercast控制结构和维护操作简单，能够有效地取得网络带宽，但是组播树中内部节点的离开不可避免的会影响到相关的子节点，系统的鲁棒性较差。SplitStream采用了多组播树模型，部署了MDC，提高了节点的下载带宽，较好地解决了系统的容错性问题，但却引入了冗余的编码，即牺牲了部分网络效率。无论是Peercast还是SplitStream在数据传输上都采用推的机制，因此在组播树中离源服务器较远的高层节点不可避免的会有更多的延时。而CoolStreaming基于网状拓扑模型，采用拉的机制进行数据传输，即新加入的节点与系统中多个节点建立连接，形成邻居关系，并互相交换数据，从而较好地利用了每个节点的能力，使得数据的传输不依赖于单个节点，节点的退出对系统的稳定性影响较小；但是在覆盖网构造中的Gossip消息交换和缓冲数据的状态信息交换中，信息负载比较大，同时会产生大量的重复消息，这相当于占用了更多的网络带宽，并造成较大的数据传输延时。

2.4 小结

本章详细介绍了几个典型的P2P流媒体分发模型，并分析比较了不同模型的特点。所有这些模型都试图在带宽效率、数据传输延时、系统鲁棒性之间寻找某

种平衡。通过对这些模型的研究，有助于我们掌握 P2PStreaming 中所用的理论和技术，为后面研究、开发新的 P2PStreaming 系统模型奠定了基础，并使得新模型在上面几个主要的性能指标上能够做出较好的平衡。

第3章 一种 QoS 保证的 P2PStreaming 模型

在分析总结了现有研究的基础上，本章将提出一个高质量的 P2P 流媒体分发模型 QCast。在 QCast 模型中，除了考虑 Peer 节点的加入、Peer 节点的离开和失效处理、Peer 节点的异构性之外，还从数据分发体系和数据分发调度的角度出发，研究了若干提供 QoS 保证的算法。本章将详细介绍 QCast 模型的服务体系构成和数据分发调度框架。

3.1 概述

在目前的 Internet 上提供流媒体分发服务，如何提供 QoS 保证是一个核心问题[37-40]。本文的目标就是要构建一个基于 P2P 网络架构的 QoS 保证的流媒体分发模型 QCast，它通过基于分布式哈希表（DHT）的应用层组播技术为直播的流媒体内容提供数据分发服务。QCast 模型具有如下几个特点：

(1) 采用分布式的控制协议，通过在每个节点上维护有限个其他节点的状态信息，使得节点在加入系统时能够快速地找到合适的父节点，而当节点离开或失效时，子节点通过所维护的状态信息能够快速准确地找到新的父节点；

(2) 节点离开和失效等处理一般不涉及到源服务器节点，从而减轻了服务器的负载，系统具有良好的可扩展性；

(3) 考虑了节点在动态网络环境下流媒体的服务质量，系统具有良好的健壮性；

(4) 考虑了节点在处理能力、网络带宽等资源方面的异构性。

在已有的研究工作中，QCast 和 SplitStream 一样都是构建在 P2P 基础架构 Pastry 的基础上。它们两者的主要区别是：SplitStream 采用的是多组播树拓扑模型，并建立在 MDC 编码基础上；QCast 是基于单组播树拓扑模型的，同时结合了网状拓扑协议的一些特点，是一种混合结构模型。在 QCast 中，所有节点仍然自组织成一棵数据分发树，除源服务器节点外，其他节点都采用了网状拓扑协议中的多邻居节点结构，即每个节点有一个父节点和多个邻居节点，这主要通过 Pastry 内建的邻居节点列表来维护。在 QCast 的数据分发树中，第 0 层是源服务器节点，第一层的所有节点只从源服务器节点获取数据，从树的第二层开始，节

点在初次加入系统时只从选定的父节点处获取数据，当父节点的传输速率降低或者播放质量不能满足节点自身需要时，节点可以同时从多个邻居节点处获取数据。这种多邻居节点结构最大的好处就是可以适应网络的动态变化，有效减少节点的离开或失效对部分节点造成传输中断的影响，同时减少传输延迟，提高系统的稳定性和鲁棒性。因此，QCast 既具有单组播树拓扑模型控制结构简单和维护代价较小的特点，又具有网状拓扑模型鲁棒性高的特点。此外，在 QCast 模型的设计中，还考虑了 QoS 保证的问题，从网络的传输延迟、节点的异构性、数据的分发调度等不同角度出发研究 QoS 保证算法。不同于 SplitStream 中引入 MDC 编码造成协议的不兼容，QCast 采用的是现有主流的流媒体传输协议、编码器和播放器都是使用通用技术，因此不作为本文研究的内容。本文研究的重点是解决利用 P2P 技术来进行流媒体分发所面临的挑战问题，尤其是 QoS 保证问题。在本章中主要完成了 QCast 模型的框架设计。

3.2 基于 DHT 的应用层组播

应用层组播（Application Layer Multicast，ALM）的主要思想就是利用覆盖网（Overlay Network）技术，在 IP 网络上构建一个虚拟网络，将组播功能从路由器移到端系统，由端系统完成诸如成员管理，数据包复制和分发等功能。一个实用的应用层组播结构应该在充分利用覆盖网中每个节点的能力的同时符合下面两个性质：(1)可靠性，当节点频繁加入或退出时，组播结构能够保持连通和正常运行；(2)可扩展性，当用户规模增大时，组播结构依然能适用，算法的开销不能增大很多，不能造成某些节点负担过重。

分布式哈希表（Distributed Hash Table，DHT）是结构化 P2P 网络的核心技术，由于其具有良好的可扩展性、鲁棒性和负载平衡等特征，在近年来开始被学术界作为覆盖网的构造和维护协议进行研究[24][29]。DHT 的核心思想是通过将存储对象的特征（关键字）经过哈希运算得到键值（Hash Key）后，对象的分布存储依据键值来进行。在基于 DHT 的应用层程序中，数据对象（文件、数据块等）通过哈希算法获得键值后，该键值被提交给 DHT，返回结果就是键值所在节点的 IP 地址。图 3-1 显示了这种应用结构。

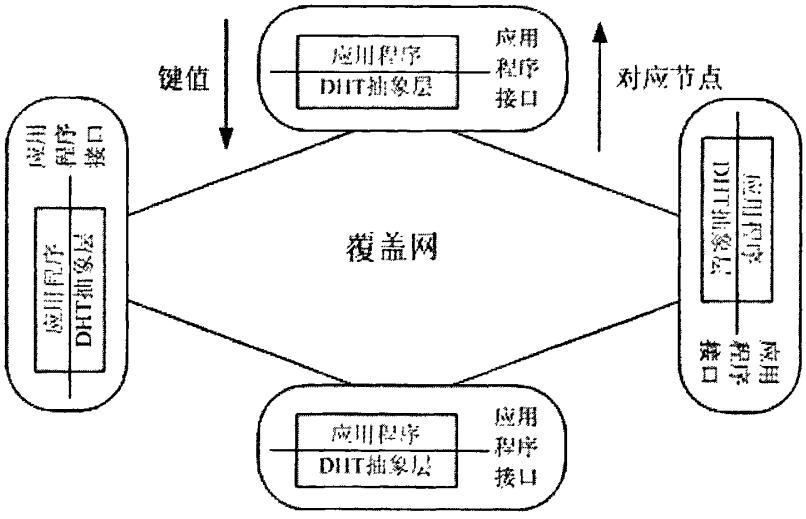

图 3-1 DHT 的应用结构

基于 DHT 构建应用层组播有两大优点：(1)应用层组播程序可以轻易地利用 DHT 的路由机制、消息触发机制和故障恢复机制来组织覆盖网，从而免除了处理网络动态变化和节点维护等繁重的工作；(2)同一个基于 DHT 的覆盖网可以被多个应用层程序或者多棵组播树同时使用，便于构建各种各样的组通讯应用。[41]

QCast 就是一种基于 DHT 的应用层组播模型。在 QCast 中，利用分布式哈希表协议 Pastry 来建立一个自组织的、高扩展性的、高容错性的覆盖网。同典型的 Pastry 节点一样， QCast 的每个节点都拥有一个 128-bit 的标识（nodeId），每个节点维护一个路由表 R(Routing Table)，一个邻居节点集 M(Neighborhood Set)和一个叶子节点集 L（Leaf Set），它们一起构成了节点的状态表。路由表的每行包括了 2^b -1（b 为系统参数，典型值为 4）个表项，每个表项记录了一个节点的信息，包括节点标识、IP 地址、当前状态等。第 n 行的表项所记录的节点和当前节点存在着这样的关系：它们的 nodeId 的前 n 位相同，而第 n+1 位则分别取从 0 到 2^b -1 的值（当前节点 nodeId 第 n+1 位的值除外）。叶子节点集合 L 中存放的是在标识空间中与当前节点距离最近的|L|个节点的信息，其中|L|/2 个节点标识大于当前节点，|L|/2 个节点标识小于当前节点。邻居节点集合 M 中存放的是在真实网络中与当前节点“距离”最近的|M|个节点的信息。在 QCast 中，利用叶子节点集合来存储每个节点的父节点和子节点等在标识空间中距离最近的|L|个节点的信息，其中，父节点的信息存放在叶子节点集合的第一项中；同时利用

邻居节点集合来存储网状拓扑协议的多邻居节点结构，使得每个节点的数据可以从多个节点中获取，其中的“距离”是指综合了网络传输延迟、传输路径带宽等因素后所得的开销。QCast 中，|L|和|M|的典型值都为 $2*2^b$。

QCast 完全利用了 Pastry 的路由机制，2.2.2 小节中介绍过这种机制，它采用的是最长前缀匹配的算法。当收到路由消息时，节点首先检查消息键值（Key）是否落在叶子节点集内。如果是，则直接把消息转发给叶子节点集中节点标识和消息键值最接近的节点；否则根据最长前缀优先的原则从路由表中选择一个节点作为消息转发的目标。如果不存在这样的节点，当前节点就从所有节点集合中选择一个 nodeId 最接近消息键值的节点作为转发目标。在这个过程中，每一步路由和上一步相比都更靠近目标节点，因此整个过程是收敛的。总的来说，Pastry 网络中的平均路由步骤 Z 满足：

$$Z < \left\lceil \log_{2^b} N \right\rceil \tag{3-1}$$

3.3 QCast 的分发体系及构成

3.3.1 QoS 保证的节点 ID 分配算法

在利用 DHT 进行 P2P 应用层组播的研究中，有研究者提出了一种 OM-QoS 的思想[42]，目标是在组播树的构建过程中引入 QoS 保证的思想。而在基于 DHT 构建的覆盖网中，每个节点都有一个唯一的标识（ID），通常这个 ID 是在节点加入系统时随机分配的。在 2.2.2 节中我们提到过，Pastry 网络中节点的 ID 是随机分配的 128-bit 的长数字。这种随机性并没有考虑到标识空间中位置相邻的任意两个结点在处理能力、网络配置、QoS 需求等方面都可能存在的差异性。并且这种节点之间的异构性在今天的 IP 网络上是很常见的。本文将参考 OM-QoS 的思想，通过在 Pastry 中引入一种新的 ID 分配算法，这种 ID 分配算法充分考虑了节点之间的异构性，同时利用这种异构性为 QCast 中组播树的构建提供 QoS 保证。

1. OM-QoS 方法

OM-QoS（Overlay Multicast Quality of Service）的目标是构建一种框架来为基于结构化 P2P 网络的应用层组播程序引入 QoS 保证的思想。这种方法已经在

EuQoS 项目[43]中得到了推广应用。OM-QoS 的思想包括：

(1) 定义若干 QoS 类（QoS class），用来表征节点的 QoS 能力，不同的 QoS 类之间是一种全序关系，即是可以比较大小的，并且 QoS 类的总数目是有限的；

(2) 在组播树的构建过程中，根节点拥有最大的 QoS 能力（QoS capabilities）；

(3) 在组播树中，孩子节点的 QoS 能力必须小于或等于父节点的 QoS 能力，即：

$$QoSCapabilities(child) \le QoSCapabilities(parent) \quad (3\text{-}2)$$

换句话说，就是在组播树中，所有从根节点到叶子节点的端到端的路径都具有 QoS 能力单调递减的性质。图 3-2 展示了一棵满足这种性质的组播树的例子。树中所有端到端的路径都满足这个性质：从根节点到叶子节点的所有节点中，QoS 能力要么保持相同，要么不断减少。在 OM-QoS 的思想中，节点的 QoS 能力是通过映射到 QoS 类来进行比较的。而节点属于哪个 QoS 类则可以综合节点的 QoS 需求、传输带宽、丢包情况等因素进行考虑。

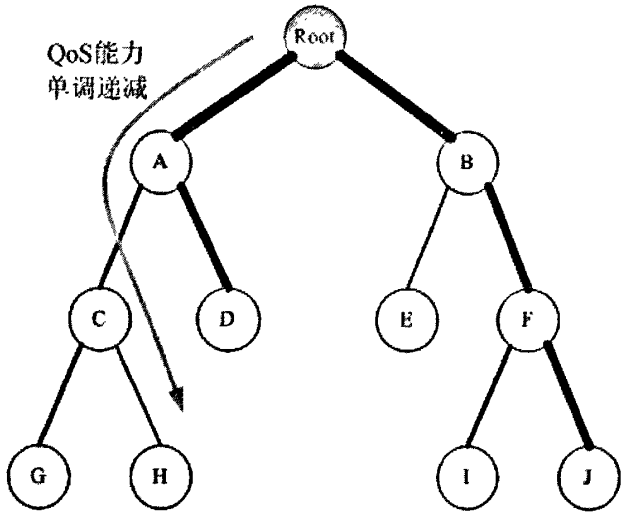

图 3-2 一棵满足 QoS 能力从根节点到叶子节点单调递减的组播树

2. 一种新的节点 ID 分配算法

在 2.2.2 节中我们介绍过 Scribe，它是一个构建在 Pastry 上的应用层组通讯系统。如果我们分析 Scribe 中组播树的构建的话，可以看出 Scribe 的组播树是不满足式(3-2)的，这是因为 Scribe 中节点的 ID 是随机分配的，这种随机分配算法并不考虑节点 QoS 能力的异构性，因此所有端到端的路径中从根节点到叶子节

点都是随机选择的，只有偶然的巧合才满足 QoS 能力单调递减的性质。第四章的实验表明，使用 Pastry 默认的节点 ID 分配算法，在任意给 Scribe 中的节点指派一个 QoS 能力的情况下，所有端到端的路径中只有不到 40%的路径满足上述提到的性质。本文将提出一种新的 ID 分配算法，使得 QCast 中能够构建“QoS 感知”（QoS-aware）的组播树。这种新的 ID 分配算法主要包括：

(1) 节点 QoS 能力的计算。在 QCast 模型中，节点的 QoS 能力是由节点的可用带宽、处理能力、在线时间以及节点的需求带宽共同决定的。计算节点 p 的 QoS 能力的函数如下：

$$QoSCapabilities(p) = q1 * Capabilities(p) + q2 * availBandwidth(p) + q3 * TTL(p) + q4 * reqBandwidth(p) \quad (3\text{-}3)$$

其中：QoSCapabilities(p)表示节点 p 的 QoS 能力值；Capabilities(p)表示节点 p 的处理能力，是对节点的 CPU 和存储能力等的度量；availBandwidth(p)表示节点 p 的可用带宽，它的计算方法和约束条件将在 3.3.2 小节中介绍；TTL(p)表示节点 p 的在线时间，通常，刚加入系统的节点的 TTL 值为 1，以后每增加一段时间（例如 10 分钟），TTL 值就增加 1，我们用 TTL 值来间接反映该节点的稳定性；reqBandwidth(p)表示节点 p 的需求带宽，p 的上级节点以此带宽向 p 转发数据，它直接反映了节点 p 自身的 QoS 需求；q1、q2、q3 和 q4 表示权重，它们的值要根据系统的实际情况来设定，但必须满足 0<q1<1，0<q2<1，0<q3<1，0<q4<1，q1+q2+q3+q4=1。通常，节点在第一次加入系统时我们只能得到一个大概的 QoS 能力值，随着节点在线时间的增加，节点的 QoS 能力值的计算就越准确。每次节点正常离开后重新加入系统时，我们就把根据历史数据和式(3-2)计算出来的值作为节点新的 QoS 能力值。

(2) 节点根据 QoS 能力值被映射到对应的 QoS 类。对于 QoS 类的划分，在考虑尽量利用节点异构性的同时也要注意到 QoS 类的划分层次会影响到组播树的深度，组播树深度的增加会导致从根节点到叶子节点的传输延迟增加。在 QCast 中，一共划分了五个 QoS 类，分别是：root class、QoS class 1、QoS class 2、QoS class 3 和 best effort class。从 root class 到 best effort class 所代表的 QoS 能力依次递减。正常情况下，源服务器节点将属于 root class，而从源服务器节点直接接收媒体内容的第一层节点将属于 root class 或者更小的 QoS

类，从第一层节点接收媒体内容的第二层节点将属于等于或小于它的父节点的 QoS 类，以此类推。在树中，大部分的叶子节点将属于 best effort class。

(3) Pastry 网络的节点 ID 空间也将对应划分成五个不同的区间，每个区间对应一个 QoS 类。QoS 类的大小顺序决定了区间值的大小顺序。例如 best effort class 将对应 ID 值最小的区间，root class 将对应 ID 值最大的区间。当节点加入系统时，将根据它的 QoS 能力从对应的区间中随机选择一个 ID 赋予它。QCast 中 Pastry 网络的节点 ID 空间的划分如图 3-3 所示。

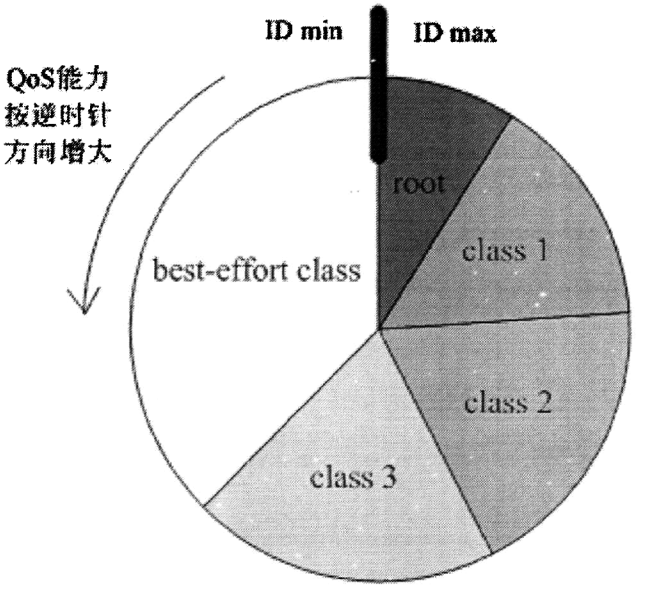

图 3-3 QCast 中 Pastry 网络的节点 ID 空间划分

在 QCast 中，规定源服务器节点的 QoS 能力是最大的，通过给源服务器节点赋予最大的 ID 值，所有节点在加入系统时都必须给源服务器节点发送请求消息，根据 Pastry 路由机制中最长前缀匹配的原则，很容易保证端到端路径中从叶子节点到根节点的 ID 值是逐步靠近的。此外，还有一个关于区间大小分配的问题。一般来说，从 root class 到 best effort class 的区间大小应该是依次增大的。因为随着组播树中节点的增加，大部分节点的 QoS 能力是属于后面几个 QoS 类的。但是如何找到一个最优的区间划分算法是目前还没解决的问题，它与系统的实际运行情况相关，区间划分的均匀程度将影响分布到每个 Peer 节点上的负载。

在组播树的构建过程考虑节点的 QoS 能力，并以此为根据给节点分配对应的 ID，这样一方面考虑了 P2P 网络中节点在处理能力、网络带宽、QoS 需求等

存在异构性的特征，另一方面也利用了 Peer 节点之间的这种异构性，把资源较“好”且比较稳定的节点放在组播树中靠近根节点的地方，从而尽量避免了在流媒体的分发过程中出现转发节点资源不足而影响服务质量的现象，保证了流媒体传输的连续性和稳定性。

3.3.2 节点的加入

虽然 QCast 模型中引入了网状拓扑协议的多邻居节点结构，它所构造的是一棵具有特殊结构的组播树，但是在本质上它仍然是基于组播树模型的。在组播树模型中，构建组播树时所有节点要满足一定的约束条件[44]。

首先我们介绍节点的带宽衡量指标。对于一个节点 p，它的带宽衡量指标除了上节中提及的可用带宽 availBandwidth(p)和需求带宽 reqBandwidth(p)外，还有最大带宽 capBandwidth(p)和已使用带宽 usedBandwidth(p)。因此，对于 P2P 网络中的任意一个节点 p，可以表示成：

$p = (capBandwidth(p), usedBandwidth(p), reqBandwidth(p), availBandwidth(p))$ [44]

最大带宽代表节点 p 的最大转发能力，它和已使用带宽、可用带宽存在以下关系：

$$availBandwidth(p) = capBandwidth(p) - usedBandwidth(p) \tag{3-4}$$

而节点的已使用带宽又和节点以及它的子节点的需求带宽之间存在一定关系。我们用 Children(p)表示节点 p 的子节点集合，用 root 表示根节点，则有：

$$usedBandwidth(p) = \begin{cases} \sum_{c \in Children(p)} reqBandwidth(c) & p = root \\ \sum_{c \in Children(p)} reqBandwidth(c) + reqBandwidth(p) & p \neq root \end{cases} \tag{3-5}$$

式(3-5)表示的意思是，对于源服务器节点，它的已使用带宽是其所有孩子节点所需带宽的总和；对于非源服务器节点，它的已使用带宽除了其所有孩子节点所需带宽的总和外，还包括父节点向它发送数据而消耗的带宽。

在 QCast 模型中，构建组播树的过程中所有节点必须满足以下约束条件：

①$capBandwidth(p) \geq usedBandwidth(p)$ (3-6)

②对于任意节点 $c \in Children(p)$，有：

$availBandwidth(p) \geq reqBandwidth(c)$且$reqBandwidth(p) \geq reqBandwidth(c)$ (3-7)

③对于任意节点 $c \in Children(p)$，c 和 p 的 QoS 类之间的关系满足：

$$QoSClass(p) \geq QoSClass(c) \tag{3-8}$$

节点必须满足式(3-6)是显而易见的，满足式(3-7)，是希望父节点在接纳当前加入者后仍然有较大的可用带宽，同时希望父节点与子节点之间的需求带宽尽可能接近，这样做的目的是为了达到 P2P 流媒体分发系统的一个重要目标——在组播树中容纳尽可能多的节点。至于式(3-8)，它是一个弱约束条件，在挑选父节点和邻居节点时，将首先从满足式(3-8)的节点中挑选，只有当所有候选节点都不满足式(3-8)时，为了在组播树中容纳尽可能多的节点，我们才不考虑式(3-8)的约束。

在 QCast 中，新节点的加入仍然是一个从源服务器节点出发沿组播树进行搜索，直到发现非饱和节点的过程。但与这种典型的基于单组播树模型的加入算法不同的是，QCast 构建的是具有多邻居节点的组播树模型，同时，在组播树的构建过程中，我们综合考虑了多种能够提供 QoS 保证的因素。下面给出 QCast 中的节点加入算法。设 N 代表待加入节点，Root 代表源服务器节点，算法步骤为：

(1) N 向 Root 发出 JOIN 消息，请求加入系统；

(2) Root 根据式(3-7)判断自己是否有足够的能力接纳 N 为直接子节点，如果可以，则直接接纳 N 作为它的子节点（第一层节点），同时向 N 发送 ACCEPT 消息，算法转向第 6 步；如果不可以，则 Root 向它的所有直接子节点转发该 JOIN 消息，试图为 N 找到一个可能的数据发送节点集合（包括父节点和邻居节点），这里称为集合 PP（Possible Parents）；

(3) 对于收到转发消息的节点，同样根据式(3-7)判断能否接纳 N，如果可以，则响应加入请求，向 N 发送 ACCEPT 消息，否则继续向该节点的所有子节点转发该 JOIN 消息。以此类推，这个过程一直持续到某个路径上 JOIN 消息的转发次数达到某个值 i 或者查询返回的 PP 集合中节点数达到某个值 j（i 和 j 都是系统参数，可根据系统实际情况设定），对于第一种情况，节点不再转发该 JOIN 消息；对于第二种情况，节点 N 直接丢弃后面接收到的 ACCEPT 消息；

(4) 节点 N 会陆续接收到来自 PP 集合中的节点发送的 ACCEPT 消息，为了减少组播树的传输延迟，N 接收到 PP 集合中的节点发送的 ACCEPT 消息后，向该节点发送一个 PROBE（探测）消息，探测两者之间的传输延迟；

(5) PP 集合中每个接收到 PROBE 消息的节点都向 N 返回 REPLY（回应）消息，在 REPLY 消息中还包含根据式(3-3)计算得到的该节点的 QoSCapabilities 值。节点 N 根据优先级函数 Priority(p)为 PP 集合中的每一个节点计算 Priority 值，并从满足式(3-8)的节点中先挑选 Priority 值最高的节点作为它的父节点，再按 Priority 值由高到低挑选 m 个节点作为它的邻居节点，当满足式(3-8)的候选节点不足时，则从其他节点中按 Priority 值由高到低挑选，其中 m 为系统参数，Priority(p)函数和系统参数 m 将在下面介绍；

(6) N 向父节点发送 ACK 消息，表示选择该节点为父节点并成功加入系统，同时将父节点的信息加入到自己的叶子节点集中；父节点接收到 ACK 消息后，也会将 N 的信息加入到它的叶子节点集中。如果 N 选择的父节点不是源服务器节点，N 还需要向它选择的邻居节点发送 CONNECT 消息，表示选择该节点作为 N 的邻居节点；邻居节点接收到 CONNECT 消息后，与 N 建立连接并把 N 加入到自己的叶子节点集中。

对算法的说明如下：

(1) 在 3.1 节中我们介绍过，在 QCast 模型中，源服务器节点的直接子节点（即组播树的第一层节点）只从源服务器节点接收数据，因此不需要为它们挑选邻居节点；

(2) QCast 中，所选择的邻居节点的信息是通过存放在 Pastry 架构中的邻居节点集来维护的，因此加入算法中的系统参数 m 的取值必须满足约束条件：$0<m<\max(2*2^b,n)$，其中 b 为 3.2 节中介绍的 Pastry 网络的系统参数，n 表示最终返回查询的 PP 集合的节点数；

(3) 设 c 为待加入节点，计算其候选数据发送节点 p 的优先级函数定义如下：

$$Priority(p) = q5 * Delay(p,c) + q6 * QoSCapabilities(p) \qquad (3\text{-}9)$$

其中，Delay(p,c)表示候选数据发送节点 p 和待加入节点 c 之间的数据传输延迟；QoSCapabilities(p)表示节点 p 的 QoS 能力，它的定义见式(3-3)，通过 QoSCapabilities(p)，我们在挑选父节点和邻居节点时，虽然可能出现不满足约束条件式(3-8)的情况，但是已经隐含地把式(3-8)作为挑选数据发送节点的一个优先条件；q5 和 q6 表示权重，以区别式(3-3)中的 q1、q2、q3、q4，它们的取值满足：$0<q5<1$，$0<q6<1$，$q5+q6=1$；

(4) 现在已经有很多有效的方法可用来准确地探测两个节点之间的传输延迟和带宽，但是这些方法会加重网络的负载，同时可能消耗其他连接的带宽，本文采用的探测方法是一种“轻量级”的方法，它得到的延迟是一种相对延迟，虽然不如那些“重量级”方法准确，但是避免了上述问题。这种探测传输延迟的方法非常重要，它有利于节点自组织成一个“拓扑感知”（Topologically-aware）的网络，使得地理上相邻的节点在覆盖网上有很大机会也是相邻的。例如，对于局域网中的节点，它的数据发送节点最好也能来自于同一个局域网，这将大大提高数据传输效率。

从整个加入算法可以看出，我们在挑选待加入节点的父节点和邻居节点时，综合考虑了传输延迟、可用带宽、节点的稳定性和处理能力等几个因素。这是为了保证流媒体数据传输的实时性和连续性。为了保证流媒体数据传输的实时性，应该尽量减少节点之间的数据传输延迟；为了保证流媒体数据传输的连续性，所选择的数据发送节点应该尽量提供较大的可用带宽，同时，数据发送节点应该是网络中较稳定、资源较好的节点。

值得一提的是，思科的Overcast[22]为了防止非叶节点的意外断开而导致数据分发树被分割，最先提出了选取若干非父节点作为“备用”父节点的思想。但是，它的目的仅仅是为了防止节点失效，而且，“备用”父节点是随机选择的，故障恢复的能力相对较低。这与QCast模型中使用的方法还是有很大不同的。

3.3.3 节点的退出

在P2P网络中，节点是动态变化的，随时都可能退出系统。节点的退出一般分为正常退出和非正常退出两种情况。正常退出是指程序按照正常的操作步骤结束；非正常退出则是指程序出现异常，未按照正常操作步骤退出，常见的情况如程序崩溃、进程被强制结束、网络中断等。下面详细介绍正常退出和非正常退出两种情况下的恢复算法。

1. 节点正常退出

在Pastry网络中，节点正常退出时需要向相关节点发送LEAVE消息。QCast模型中完全采用这一机制，当节点正常离开系统时，需要向它的父节点、直接子节点和与之建立连接的邻居节点发送 LEAVE 消息。当父节点和邻居节点收到

LEAVE 消息时，停止向该节点发送数据，并从它的叶子节点集中删除该节点的信息。对于处于数据传输接收者的子节点来说，当它收到 LEAVE 消息时，则需要判断消息是来自父节点还是来自邻居节点，并以此为根据进行不同的处理：

(1) 若 LEAVE 消息来自父节点，子节点从它的邻居节点集中寻找一个满足式(3-8)而 Priority 值又较高的节点，若存在这样的节点，则选择它作为该子节点新的父节点，并把它的信息从邻居节点集中移到叶子节点集中，此时子节点需要重新计算剩余邻居节点的个数，若小于某个值 t（t 为系统参数，典型取值如 t = m/2，m 的意义同 3.3.2 节），则需要再寻找邻居节点，直到邻居节点的个数大于 t；若不存在这样的节点，即邻居节点集中找不到满足式(3-8)的节点，则该子节点一边保持同邻居节点的连接关系以保证数据传输不中断，一边向源服务器节点发送 JOIN 消息，重新发起加入过程。由于我们在挑选邻居节点时已经把式(3-8)作为一个优先选择条件，因此发生后一种情况的概率很小，只要 m 不至于太小，通常情况下节点都可以很快地从邻居节点集中找到新的父节点，从而保证了组播树中数据传输的连续性；

(2) 若 LEAVE 消息来自子节点的某个邻居节点，子节点直接把该节点的信息从邻居节点集中删除掉，同时判断剩余邻居节点的个数是否小于值 t，若小于，则需要再寻找邻居节点，直到邻居节点的个数大于 t。

图 3-4 给出了一个节点正常退出后组播树恢复的例子。

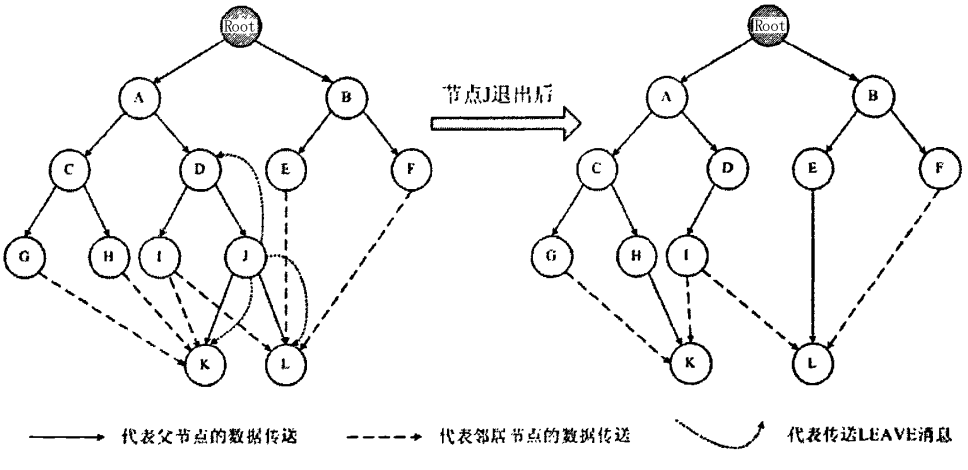

图 3-4 节点退出后组播树恢复示意图

2. 节点非正常退出

如果节点是非正常退出的，就不会向它的父节点、直接子节点和邻居节点发送 LEAVE 消息，为了侦察节点的这种失效退出的行为，Pastry 为我们提供了一种容错机制：每个节点每隔一个周期 T（T 为系统参数）就要向它的父节点、所有直接子节点和所有邻居节点发送 HEARTBEAT（心跳）消息，如果在 T 时间内都没有接收到某个节点的 HEARTBEAT 消息，其他节点就认为该节点已经失效了。相关节点侦察到该节点失效后，则可以按照节点正常退出的处理算法对组播树进行修复。

在节点的退出恢复算法上，我们充分利用了 Pastry 提供的容错机制，只是在此基础上增加了父节点的重新选择和叶子节点集、邻居节点集的维护操作。通过采用多邻居节点结构，在父节点退出时，子节点可以很快地找到新的父节点，从而能够有效地防止因父节点的退出而导致部分子节点数据传输被中断的影响。

3.4 QCast 的数据分发调度框架

在 QCast 模型中，通过引入网状拓扑协议的多邻居节点结构，当节点的父节点退出或者父节点的传输速率不能满足子节点的播放速率时，子节点可以向邻居节点请求数据，这就构成了一个由父节点（或新的父节点）和多个邻居节点共同组成的数据发送节点集合。对于这种存在多个数据发送者（multi-senders）的情况，需要建立一种调度机制来协同多个 Peer 服务节点向单个 Peer 接收节点传输流媒体数据。这种调度可以由 Peer 服务节点驱动，也可以由 Peer 接收节点驱动。在 QCast 模型中，我们选择了以接收方为驱动，这主要是考虑到了下面两个因素：(1)Peer 接收节点在数据传输调度过程中统一掌握了诸如哪些数据包已经成功到达、哪些 Peer 服务节点当前还可用、从每个 Peer 服务节点到自身的可用带宽和传输延迟等关键信息；(2)在 P2P 网络中，存在着为数不少的只接收而不转发数据的节点，因此，以接收方为驱动，可以把调度负载集中到 Peer 接收节点上面，大大减轻了 Peer 服务节点的调度负载，从而有利于 Peer 服务节点同时为多个 Peer 接收节点提供数据服务。在 QCast 的数据传输方式上，我们采用了推拉结合的策略。顾名思义，推拉结合的策略包含了推和拉两种工作模式，这两种模式配合使

用，可以克服单独采用推模式或者单独采用拉模式的缺陷，提高了流媒体传输的服务质量。此外，为了实现这种以接收方为驱动、推拉结合的数据分发策略，还必须设计相应的缓存机制，这也是应对 P2P 网络动态性特征的需要。

3.4.1 缓存机制（Buffering）

在网络中传输的流媒体文件会被分解成多个数据包，并且每个数据包所选择的路由是不尽相同的，因此每个数据包到达目的地的时间延迟也是不一样的；而 P2P 网络更是“先天”具有动态性，节点随时都可能加入或离开系统，在这种情况下，为了保证流媒体传输的连续性和稳定性，要求基于 P2P 的流媒体分发系统都要具有一定的数据缓冲区，即利用缓存机制来弥补流媒体数据传输过程中延迟和抖动的影响，同时保证正确的数据包播放顺序，这样就可以克服由于网络拥塞或者组播树修复而导致系统播放出现停顿的问题。同时，缓存机制也是 QCast 模型中采用推拉结合的流传输策略的必要条件。

在 QCast 模型中，采用一个特殊的队列来实现缓冲区，如图 3-5 所示，整个缓冲区分成等大小的两部分，播放区和回收区，缓冲区中有三个指针来标识关键的位置，其中 Seq_Min 指向整个缓冲区中序号最小的数据片段，Seq_Play 指向当前准备播放的数据片段，Seq_Max 指向整个缓冲区中序号最大的数据片段，当数据开始播放后，播放区的数据片段向后滑动一格进入回收区，回收区每接收到一个数据也向后滑动一格，当回收区满时，则删除最先进入的那个数据片段。

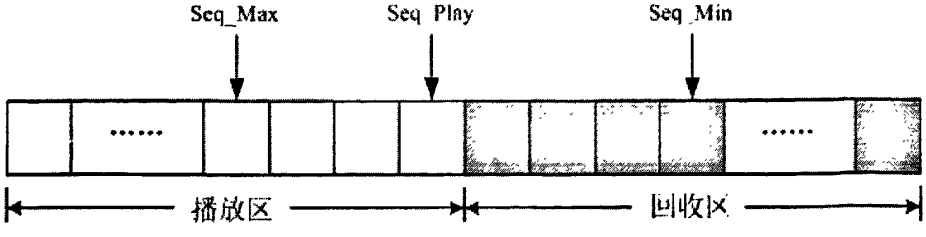

图 3-5 QCast 的缓冲区设计

为了实现缓存机制，流媒体数据需要被划分成块。通常有两种数据分块的方法：按固定大小分块和按固定时间分块。由于很难预知流媒体数据的大小，而且对流数据一般都采用恒定比特率（CBR）的方式编码，因此 QCast 模型中采用按

固定时间分块的方法，把每1秒钟的数据流划分成一块，并为每个数据块赋予一个序号。整个缓冲区分为60块，每个缓存块存放1秒钟的流数据，即最多可以缓存60秒钟的流数据。

其中播放区有30个缓存块，每个缓存块按照等时间间隔存放流数据。进入播放区的流数据，在播放前必须保证正确的播放顺序，而网络中传输的数据包由于选择的路由不同，因此不一定按照预定的顺序到达，这就要求对进入播放区的数据块按照序号进行排序。通常情况下，网络中大部分的数据包到达时是按照预定的顺序的，只有少部分是乱序的，因此QCast中采用了插入排序对播放区中的数据块进行排序，这种排序方法在大部分元素已经有序的情况下是最具效率的排序方法。为了方便推拉结合的数据传输调度，还采用了一个长度完全一样的数组BM（Buffer Map）来表示缓冲区中数据块的映射。BM 中存放的元素是二元组<SegNo，SegTag>，它们和缓冲区中的数据块是一一映射的关系。其中，SegNo表示数据片段的序号；SegTag表示该SegNo所对应的数据片段的有无，用1表示具有该数据片段，0表示不具有该数据片段。QCast模型中，每个节点可以通过BM向其它节点提供自己的数据状态信息。

回收区的设计和播放区是一样的，它们具有同样的大小、具有同样的BM数组（其实是同一个BM数组），区别只是回收区中存放的是刚刚播放后的数据片段。引入回收区的目的有两个：(1)为了完成节点加入时的虚拟修补，减少节点加入时的延迟等待。“虚拟修补”的思想[45]最早出现在流媒体点播系统中，即节点首次加入系统时，在接收父节点正常数据传输的同时，在系统缓冲时间为t的情况下把父节点之前所播放的t/2的数据也发送给该节点；(2)为了防止出现节点刚刚向子节点发出有数据的通知但数据立刻被淘汰的情况。

最后一个问题是数据块的 SeqNo 的表示问题，由于流媒体数据播放时间可能很长，因此 SeqNo 按照每秒数据块序号递增的方式是不可行的。其实在一个流媒体分发系统中，节点之间接收数据包的延迟是有限制的，因此我们只要找到一个较大的整数m，作一个简单的哈希运算mod(m)就可以实现对SeqNo的表示。例如取m=600，缓冲区的数据块一共有60秒，因此只要任意两个节点间接收数据包的延迟不超过10分钟（即不出现节点A已经接收到第601秒的数据块，节点B才接收到第1秒的数据块），就不会出现同一个SeqNo在不同节点间表示不

同数据块的情况。

3.4.2 数据传输调度策略

1. 推拉结合的数据传输调度算法的提出

通常，基于树状模型的 P2P 流媒体分发系统在数据的传输方式上采用推的机制，即由父节点负责为子节点传送数据，这种机制最大的好处就是平均传输延迟和系统控制开销都很小，因为它不需要频繁转发各种状态信息和控制信息。然而，在纯推策略中，当父节点离开或失效时，子结点在找到新的父节点之前将暂时接收不到数据而只能回放缓冲区中的数据。更为常见的问题是，由于这种方式下是一个父节点为多个子节点传送数据，由于自身处理能力不足或者网络拥塞容易造成父节点的数据传输速率下降，导致子节点接收数据的速率也下降，经过一段时间后，子节点由于接收不到足够的数据也只能回放缓冲区中的数据。这两种情况在 P2P 流媒体分发服务中称节点进入了 Consuming 状态。一旦节点的缓冲耗尽，节点将进入 Freezing 状态。

为了克服推的机制带来的弊端，基于网状模型的 P2P 流媒体分发系统在数据的传输方式上采用了拉的机制，节点可以在任何时候从多个邻居节点处同时请求数据。在纯拉策略中，节点 A 要从节点 B 处获取一个数据包 p 需要经过三个步骤：(1) B 发送 BM（Buffer Map）给 A，声明自己的缓存中拥有数据包 p；(2) 如果 A 需要数据包 p，则向 B 发送请求消息；(3)B 接收到请求消息后，将数据包 p 发送给 A。整个过程至少需要节点 A 和 B 进行 3 次通信。另外，考虑到网络传输中的效率问题，节点并不是针对每个数据包发送一次 BM 和请求，而是将一组数据包的信息合并起来发送，这样就导致数据包在节点之间的平均传输延迟进一步增大。另外，节点之间还需要周期性地发送 BM 信息和请求消息，使得网络流量中控制信息的比重较高，系统的控制开销也会增大。

为了解决纯推策略的不足，QCast 模型在单组播树拓扑的基础上引入了网状拓扑的多邻居节点结构；同时，为了避免纯拉策略在传输延迟上表现不佳以及控制开销过大的问题，QCast 模型采用了一种推拉结合的数据调度传输策略，在大部分情况下，节点之间的数据传输采用推的机制；当出现数据发送节点退出或者数据发送节点的数据传输速率明显下降时，即在节点进入 Consuming 状态时，从

邻居节点集中选取一些节点加入到数据发送节点集合，采用拉的机制从多个数据发送节点处获取数据，以避免缓冲区中的数据被耗尽而进入 Freezing 状态；此后节点之间的数据传输又恢复成推的机制，如此反复。在这整个过程中，数据传输调度是由数据接收节点驱动的，即由接收方驱动的。

2. 推拉结合的数据传输调度算法的主要设计思想

整个推拉结合的数据传输调度算法可以分为两个阶段，第一个阶段是“单发送者”（single-sender）下推的阶段，第二个阶段是“多发送者”（multi-senders）下推拉结合的阶段。

在第一阶段，待加入节点选择了组播树中的一个父节点加入系统后，此时子节点只从父节点处接收数据，即父子节点之间的数据传输是典型的推机制。在节点刚加入系统时，父节点除了转发从上层节点传输来的数据外，还将其回收缓冲区中的部分数据发送给该节点，这就是虚拟修补的方法。具体做法是：假设节点 N 在 t 时刻加入系统，那么其父节点可以把回收缓冲区中在 t 时刻之前的 k 秒数据发送给 N，与此同时 N 还继续从父节点处下载从 t 时刻开始之后的数据。采用虚拟修补的方法主要是为了减少节点加入系统时的启动延迟，即减少用户加入系统时的等待时间。并且，在父节点性能较好的情况下，这种方法有助于满足父节点向子节点发送数据的传输速率应该大于或等于子节点流媒体数据的播放速率这个约束条件。

当子节点发现父节点退出系统或者子节点发现父节点的数据传输速率小于自身流媒体数据的播放速率时，算法进入第二阶段。此时子节点将同时向其多个邻居节点请求数据，挑选邻居节点到“活动节点”（这里我们把节点的数据发送节点称为“活动节点”，active node）集合中，直到“活动节点”集合中所有节点的数据传输速率之和大于或等于子节点流媒体数据的播放速率。在这个过程中，子节点向“活动节点”发送请求消息以获取“活动节点”的 BM 信息，然后从“活动节点”处把相关的数据“拉”过来，同时充分评估“活动节点”集合中每个节点的工作状态以及它们与自己的端到端链路状况。

在多个数据发送者的情况下，当节点之间又满足“活动节点”的数据传输速率之和大于或等于子节点流媒体数据的播放速率时，子节点就根据上一时间片内对各个“活动节点”的评估结果，主动向各个“活动节点”定制自己需要的数据。

这样，每个“活动节点”在接收到新的数据包后，就不需要向子节点发送 BM 信息，而是直接根据子节点的定制情况决定是否转发该数据包。当“活动节点”集合中的任一节点退出系统或者节点之间的数据传输又无法满足约束条件需要往“活动节点”集合中加入新的邻居节点时，节点之间的数据传输方式又转换成上面描述的拉机制，稳定后又恢复成推机制，如此反复。

3. 算法的具体描述

在第一阶段，节点只是处于被动接收数据的状态。在节点刚加入系统后，节点只从父节点处以推的方式获取数据，即不向其他邻居节点请求数据。因此，第一阶段只有数据传输，不涉及数据调度。

在第二阶段，无论是工作在推模式下还是拉模式下，节点都是从多个数据发送节点中获取数据，这时就出现了数据调度问题。具体来说，就是给定了一个数据请求节点和一个数据发送节点集合，数据发送节点集合中不同节点的上载（upload）带宽可能各不相同，要求在数据请求节点和每个数据发送节点之间进行数据调度以获得最小的数据传输延迟。这个问题在 QCast 模型中可以形式化地描述为：

设 N 为数据请求节点。集合 $P=\{p_1, p_2, \cdots, p_i\}$表示 i 个与 N 建立连接的节点，称为种子节点集合。显然，种子节点集合 P 中包含了一个父节点和 i-1 个邻居节点，其中 i 满足 $0<i\leq m+1$，m 为邻居节点总数。在 N 向 P 中的节点请求数据之前，N 先根据式(3-9)计算 P 中除父节点外所有 p_i 的 $Priority(p_i)$值，然后把父节点和若干个 $Priority(p_i)$值较大的节点选进“活动节点”集合，其余的作为后备种子。我们用 P_a 表示“活动节点”集合，R_N 表示节点 N 的流媒体数据播放速率，R_p 表示节点 $p(p\in P_a)$的数据传输速率，则 P_a 的选择必须满足：

$$\sum_{p\in P_a} R_p \geq R_N \tag{3-10}$$

即“活动节点”集合中所有节点的数据传输速率之和要大于或等于流媒体数据的播放速率。此外，系统还需要维持一定数量的后备种子，当正在传输的“活动节点”失效或者出现网络拥塞等原因造成“活动节点”数据传输速率之和小于流媒体数据的播放速率时，可以启用后备种子进行数据传输。

在第二阶段中采用推拉结合的策略就是为了满足式(3-10)，使节点能够尽快获取所需数据，保证流媒体播放的连续性。为了便于数据的传输调度，数据接收

节点将时间分成连续的时间片，在数据接收节点发现不能满足式(3-10)的下一个时间片内，节点将工作在拉模式下，在其他的时间片内，节点工作在推模式下。在每个时间片内，采用以下方法来处理数据的传输调度：

(1) 滑动窗口机制

无论是工作在拉模式下还是工作在推模式下，数据传输前都需要数据接收节点向“活动节点”发送请求消息，区别只是推模式下“活动节点”不需要转发 BM 信息。为此，我们可以通过在数据接收节点中建立滑动窗口的机制来对各个“活动节点”的传输行为进行协调，通过结合缓冲区中缓存块的情况，还可以确定下一个时间片Δ内允许请求调度的数据块范围。滑动窗口的工作过程如图 3-6 所示，其中，tp 表示数据接收节点当前的播放时刻，tw 表示滑动窗口处的时间。开始时 tp 与 tw 相距为 0，滑动窗口向前滑动Δ，数据接收节点启动一次调度过程，并在后续的Δ时间内保持滑动窗口的位置不变，当 tp 与 tw 相距再次为 0 时，重复上述过程。

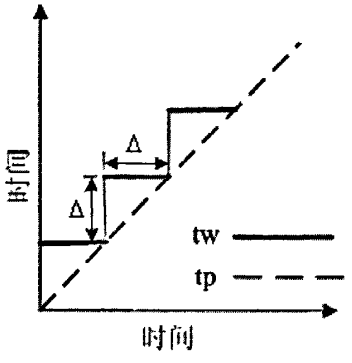

图 3-6 滑动窗口的工作过程

(2) 拉模式下的数据获取

同 CoolStreaming 一样，数据接收节点在拉模式下工作时必须满足一个约束条件：每个缓存块必须在播放时间 Deadline 内获得，否则作为过期数据没有任何意义。其中 Deadline 是一个系统参数，它取决于播放缓冲区的大小，如在 QCast 中，Deadline 为 30 秒。为了满足这个约束条件，QCast 中采用文献[46]中描述的方法：在向“活动节点”集合获取某个缓存块之前，数据接收节点会根据当前系

统时间（CurrentSysTime）、该缓存块的时间戳（PacketTime）以及节点之间的网络传输延时（Round-Trip Time，RTT）来计算该缓存块的生存时间（SegmentTTL），并根据 SegmentTTL 是否小于 Deadline 来决定是否获取该缓存块。即满足：

$$SegmenetTTL = CurrentSysTime - PacketTime + RTT < Deadline \quad [46] \qquad (3\text{-}11)$$

时，才有必要获取该缓存块。若该缓存块只有一个提供者满足式(3-11)，那么就只有从这个提供者处获取数据，若存在多个提供者，则可以优先选择从网络传输延迟最小的那个节点处获取数据，若延迟相近，则可以选择从可用带宽最大的节点处获取数据。

(3) 推模式下的数据获取

当工作在推模式下时，数据接收节点在时间片Δ内向各个“活动节点”定制的数据不应该出现重复的情况，否则将会浪费网络带宽。这个问题可以采用 3.4.1 小节介绍的方法来解决，节点在向各个“活动节点”定制数据时，可以通过对数据块序号作一个简单的哈希运算 mod(m)来实现，针对可能的结果 0，1，…，m-1，节点根据式(3-9)计算得到每个“活动节点”p 的 Priority(p)值，再以此为根据确定向各个“活动节点”定制数据的比例，这可以直接根据 Priority(p)值划分比例，但采用“轮盘赌”的算法来确定比例效果更好。

(4) 数据包丢失处理

在通常的网络环境下，无论是拉模式还是推模式都可能出现数据包延迟到达或丢失的情况。在拉模式下，一次数据请求后如果发生丢包的情况，数据接收节点可以继续向其他“活动节点”请求丢失的数据包，因此不需要对丢包进行特别处理；在推模式下，对丢失的数据包虽然也可以通过“拉”的方式再次向其他“活动节点”发送请求来获得，但在处理上则要相对复杂。这是因为在推模式下节点转发数据包的顺序和它接收数据包的顺序是一致的（注意转发时并不排序，排序是发生在播放缓冲区中），由于路由的不同不能确保转发时数据包是有序的，因此难以根据接收到的数据包顺序来判断中间是否有丢包的情况发生。例如，节点 A 向节点 B 定制了数据包 1，2，3，4，5，6，7，…，由于 B 从上层节点收到数据包的顺序很可能为 1，2，4，5，7，6，3，…，并且 B 在收到数据包的同时立即向 A 转发，因此 A 收到数据包的顺序也很可能是 1，2，4，5，7，6，3，…。这样，当 A 收到序号为 7 的数据包时，它无法判断数据包 3 和 6 是否真的丢失

了。这时如果 B 立即向其他“活动节点”重新请求数据包 3 和 6，则很可能会收到重复的数据，从而浪费了带宽。为了防止这种情况发生，可以在定制数据的时候指定一个最大落后序号差 g。转发时，如果落后数据包的序号和已发送数据包的最大序号相差 g 以上，则不再对该数据包进行转发[28]。同时，接收节点也可以根据 g 来判断是否需要重新请求未到的数据包。假如在上面例子中，如果指定 g=3，当节点 B 在发送了数据包 7 和 6 之后，收到了数据包 3，这和已经发送数据包的最大序号 7 相差大于 g，因此 B 不再转发数据包 3。对于接收节点 A，当它收到数据包 7 之后就知道 B 不会向其转发数据包 3 了，它可以立即通过“拉”的方式从其他“活动节点”处获取数据包 3，从而避免了数据包的重复传送。

3.5 小结

在 Internet 上提供高质量的流媒体分发服务是一项极具挑战性的工作。本章提出了一种 QoS 保证的 P2PStreaming 模型——QCast 模型，它从数据分发体系和数据分发调度策略出发为 P2P 流媒体分发服务提供 QoS 保证。QCast 是构建在 Pastry 基础上的应用层组播模型，通过修改 Pastry 的 ID 分配算法，根据节点的 QoS 能力来分配 ID，使得 QoS 能力越强的节点越靠近组播树的根节点，这样一方面利用了节点之间的异构性，另一方面可以“强迫”节点在加入系统时按照“QoS 感知”的方式来构建组播树。在基于单组播树拓扑的基础上，QCast 引入了网络拓扑协议的多邻居节点结构，使得发生节点离开或失效行为时，子节点可以很快找到新的父节点，并保证这一过程中数据传输的连续性和稳定性。针对这一结构中潜在的多个数据发送者，QCast 模型采用了一种接收方驱动的、推拉结合的数据传输调度策略，以最大化满足数据接收节点的 QoS 要求，保证节点的播放质量。

第4章　系统原型实现与性能评估

基于上一章中 QCast 模型的数据分发体系和数据分发调度框架的设计，本章实现了 QCast 模型的系统原型和核心算法，并在此基础上设计仿真实验，根据实验结果对 QCast 模型进行性能评估。

4.1 概述

QCast 的系统原型的实现是构建在各种开源技术基础上的。通过采用这些符合标准的开源技术，使得整个系统具有良好的通用性和可移植性，同时也使得系统的编码工作变得较为简单。

首先，由于 QCast 是构建在分布式哈希表 Pastry 基础上的，因此我们选择了著名的开源项目 FreePastry[35]作为系统的底层架构。FreePastry 是 Rice 大学发起的，微软研究院等参与的开源分布式哈希表项目，它是采用 Java 语言编写的 Pastry 协议的完整实现，为构建基于 Pastry 的 P2P 网络提供了完整的 API。在 FreePastry 的软件包中还包含了应用层组通讯系统 Scribe 和基于多树拓扑的应用层组播协议 SplitStream 等的完整实现。此外，FreePastry 还是一个被广泛使用的 P2P 仿真器[42][47]，FreePastry 的仿真器是一个离散事件仿真器（Discreet Event Simulator），可以用来仿真各种规模的 P2P 网络。在 QCast 中，我们将利用 FreePastry 的 API 来构建 P2P 网络和应用层组播树，同时借助于 FreePastry 的仿真器来设计仿真实验。

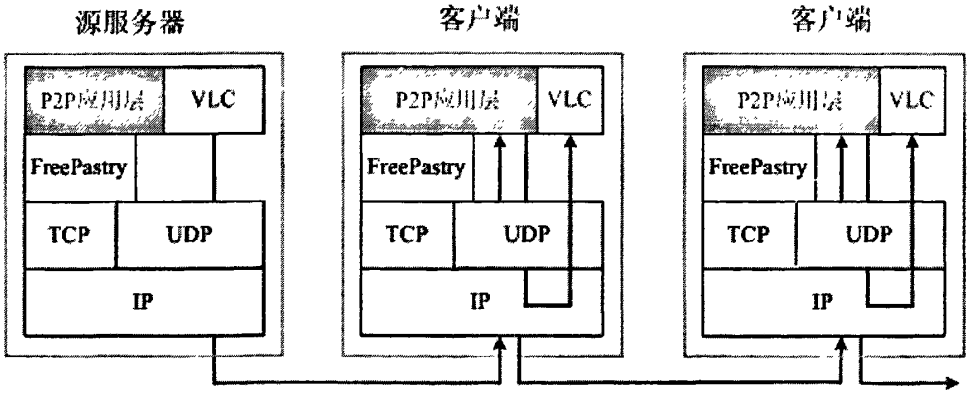

图 4-1 QCast 的技术架构图

在 QCast 中，我们研究的重点是系统的数据分发体系构成和数据分发调度算法，而对于编码器、解码器、播放器等都是采用通用技术。这里我们采用的是 VLC[48]，因为它是完全开源的并且提供了丰富的文档资源。VLC 是跨平台的，它支持所有主流的多媒体格式，并且既可以作为流媒体播放的服务器，又可以作为客户端。然而，VLC 并不是 QCast 中唯一可以使用的播放器，QCast 在设计上采用分层的思想将流媒体应用层与系统的其他部分进行分离，可以随时根据需要采用其他的播放器。

系统的技术架构图如图 4-1 所示。

4.2 系统原型设计与实现

4.2.1 QCast 的系统设计

QCast 采用分层的设计思想，系统结构如图 4-2 所示，整个系统可以划分为三层：P2P 网络层、流媒体控制层和流媒体应用层。其中，P2P 网络层和流媒体控制

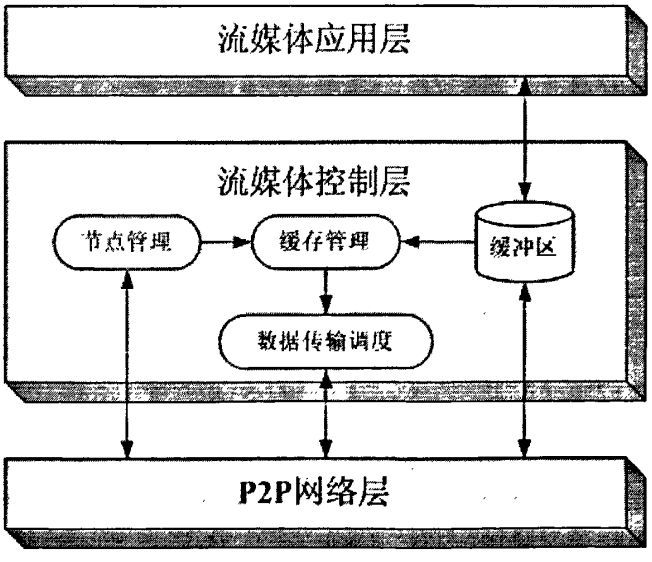

图 4-2 QCast 的分层结构图

层联系比较紧密，这两层一起封装了 Peer 节点的逻辑和底层网络；流媒体应用层比较独立，在源服务器节点中，这一层主要负责流媒体数据的输出；在普通节点中，这一层除了负责流媒体数据的输出，还负责从缓冲区中取数据，再交由本地

的播放器进行播放。由于流媒体应用层只是对 VLC 的封装，这里不再详细介绍，下面重点介绍 P2P 网络层和流媒体控制层。

1. P2P 网络层

P2P 网络层中包含的主要类如图 4-3 所示。上层应用通过调用 NodeManager 类创建一个节点对象，它是对 P2P 网络中节点的属性和逻辑的封装。如果该节点不是 P2P 网络的根节点，则执行 join 方法加入 Pastry 环。当节点对象被创建之后，紧接着就创建 StreamDataServer、MyMessage、MsgStack 和 MsgProcessor 这四个对象。其中，StreamDataServer 是用于流媒体数据传输的监听线程，它包含了一个套接字用于监听流媒体数据的请求，每收到一个新的流媒体连接请求，就启动一个新的线程来处理该请求。MyMessage、MsgStack 和 MsgProcessor 一起构成 QCast 中消息处理的核心：MyMessage 是所有控制消息的基类；MsgStack 是消息栈，用于存储 MyMessage；MsgProcessor 是消息处理线程，它会针对不同类型的消息调用不同的逻辑进行处理。

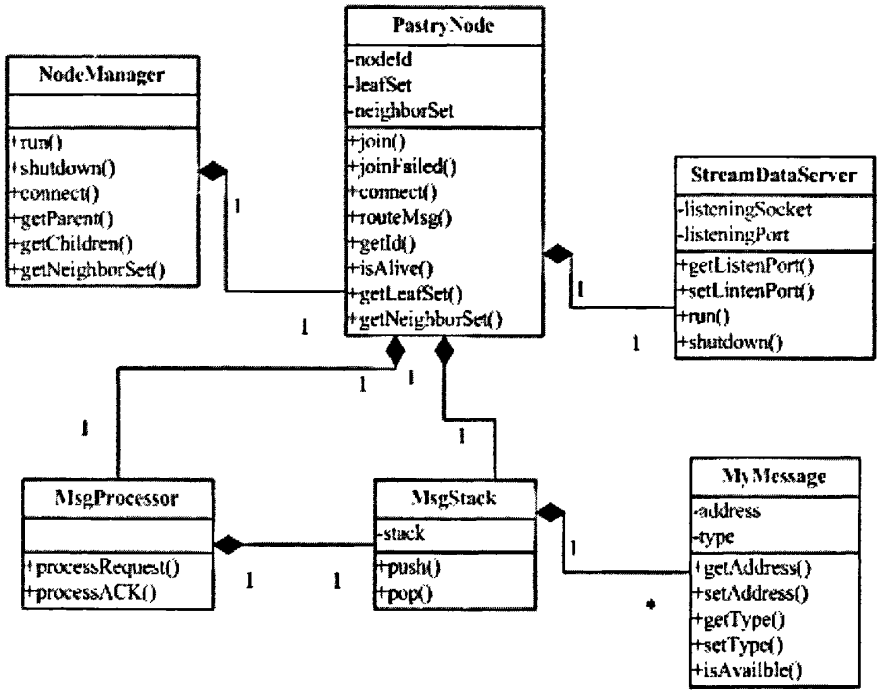

图 4-3 P2P 网络层的主要类图

2. 流媒体控制层

流媒体控制层中包含的主要类如图 4-4 所示。这一层通过 NodeManager 类和 P2P 网络层联系在一起。其中主要的类有 BufferManager、BufferMap、DataScheduler 和 StreamManager。BufferManager 和 BufferMap 一起构成了 QCast 的缓冲区管理模块，BufferMap 表示缓冲区中缓存块的映射，BufferManager 负责管理和维护缓冲区，包含了读写缓存块、清空缓冲区、设置指针变量等操作，BufferManager 是缓冲区对外的接口，每个节点只能有一个 BufferManager 实例，这可以通过单态模式来实现。DataScheduler 是 QCast 的数据调度模块，封装了数据分发调度的逻辑；StreamManager 类主要实现了对流媒体数据收、发逻辑的封装。

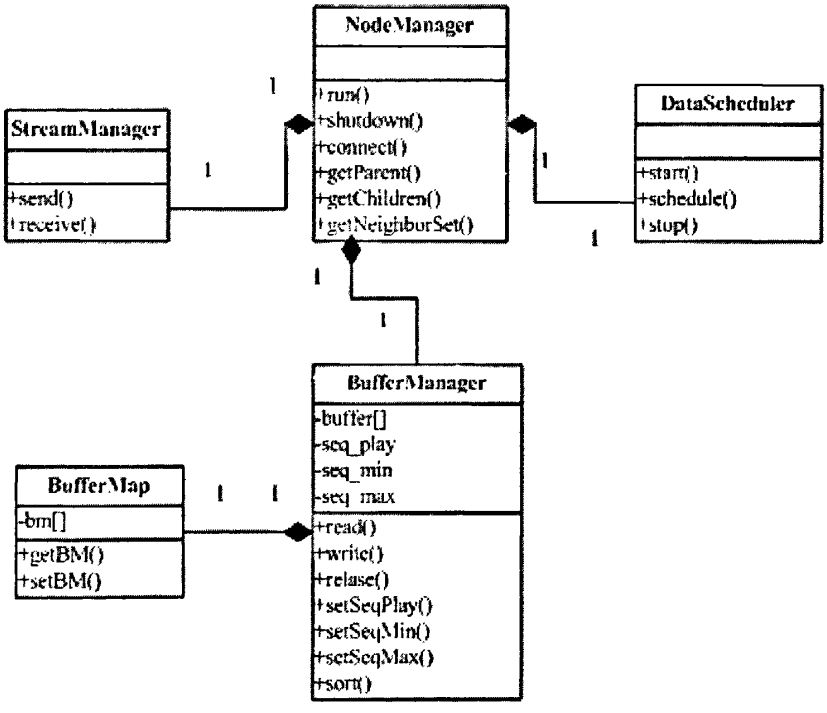

图 4-4 流媒体控制层的主要类图

4.2.2 QCast 的核心算法设计

本小节将采用 Java 伪代码的形式描述 QCast 系统中的核心算法设计，包括节点的加入算法、滑动窗口控制算法、推拉结合策略下推机制的数据获取算法和拉机制的数据获取算法等。在算法描述的过程中，我们使用了表 4-1 所描述的修改后的 FreePastry 软件包的部分 API。

表 4-1 根据 FreePastry 修改的部分重要的 API

返回值	函数原型	功能描述
void	routeMsg(Id key, Message msg)	将消息路由到 Id 最接近 key 的节点
boolean	allowJoin( )	判断节点是否可以接纳一个子节点
void	setChild(NodeHandle newNode)	接纳 newNode 为子节点
List	getChildren( )	获取节点的所有直接子节点

1. 节点加入算法

```
算法 1：join ( )
算法输入：
    newNode：待加入节点；
算法过程：
    parentNode = Root;    //从根节点开始寻找“非饱和”节点；
    routeMsg(parentNode, JOIN);      //向根节点发送 JOIN 消息；
    flag = parentNode.allowJoin();    //allowJoin 函数判断是否满足式(3-7)
    if (flag = = true){
        parentNode.setChild(newNode);
    }
    else{
        children = parentNode.getChildren( );    //获取直接孩子节点的集合
        while (children.hasNext( )&&Seed_Num != m+1){// m 为最大邻居节点数
            candidateNode = children.next( );
            flag = findPossibleParent(newNode, candidateNode);  //寻找 PP 节点
            if(flag = = true){
                Seed_Num++;
                possibleParents[Seed_Num] = candidateNode;
                priority[Seed_Num] = Priority(candidateNode);   //计算优先级
            }
        }
        sort(priority);  //按优先级从大到小将 priority 和 possibleParents 排序
        flag = true;
        for(i=0; i< Seed_Num; i++){
            if(flag&&QoSClass(possibleParent[i]) >= QoSClass(newNode)){
                parentNode = possileParents[i];      //选择父节点
                flag = false;
            }
            else newNode chooses possibleParents[i] as its Neighbor Node;
        }
        if(flag = = true) parentNode = possibleParents[0];
    }
```

2. 基于滑动窗口的数据调度算法

```
算法 2：schedule ( )
算法输入：
    tp：节点当前的播放时刻；
    tw：滑动窗口处的时间；
    segmengList：待获取的缓存块列表；
算法过程：
    if(tp = = tw) {  //只在时间片的开始考虑是否需要改变数据调度策略
        tw = tw + △;          //窗口向前滑动△，△为滑动窗口大小
        reset the segmentList;  //重设待获取的缓存块列表
        if(data download rate < data play rate){
            pull( );          //采用拉机制，pull 函数将在下面介绍
        }
        else{
            push( );          //采用推机制，push 函数将在下面介绍
        }
    }
```

3. 推拉结合策略下的数据获取算法

```
算法 3：push ( )          //推机制的数据获取算法
算法输入：
    activeNodes：“活动节点”列表，列表大小用 ActiveNodes_Num 表示；
    segmengList：待获取的缓存块列表；
    BM：BufferMap;
算法过程：
    if（single-sender）{      //处于数据传输的第一阶段
        receive data from the active node(parentNode); //被动从父节点接收数据
    }
    else if（multi-senders）{  //处于数据传输的第二阶段
        set seg[ActiveNodes_Num] = {segments to be received from each active
                                    node};  //按比例划分请求数据
        set j = the first segment in segmentList;
        for(i=0; i< ActiveNodes_Num; i++){ //向每个“活动节点”请求数据
            set msg = DataRequestMsg from segment j to segment seg[i];
            routeMsg(activeNodes[i], msg);    //发送数据请求消息
            j = j + seg[i];
        }
    }
    for each segment i in segmentList{     //置缓存映射表状态
        if(read segmeng i from activeNodes successfully) BM[i] = 1;
        else BM[i] = 0;
    }
```

```
算法 4：pull ( )                    //拉机制的数据获取算法
算法输入：
    activeNodes：“活动节点”列表，列表大小用 ActiveNodes_Num 表示；
    segmengList：待获取的缓存块列表；
    BM：BufferMap;
算法过程：
    for each segment i in segmentList{
        for each node j in activeNodes{
            if (i exists in j && timeout<deadline){   //timeout 根据式(3-11)计算
                add j to the supplierList of i;
            }  //将满足式(3-11)的节点加入到数据块的提供者列表中
        }
        if (size of the supplierList of i = = 1){
            send data request message to the supplier;
        }
        else if (size of the supplierList of i > = 2){
            select the best supplier among the supplierList;
            send data request message to the selected supplier;
        }   //有多个数据提供者时，依次根据延迟、带宽选择最佳提供者
    }
    for each segment i in segmentList{      //置缓存映射表状态
        if(read segmeng i from activeNodes successfully) BM[i] = 1;
        else BM[i] = 0;
    }
```

4.2.3 系统原型运行效果

图 4-5 是用 QCast 的系统原型搭建一个 P2P 网络电台时的运行效果图，包括播放窗口、控制窗口和日志窗口。如图所示，节点与多个节点保持连接关系，从“活动节点”处获取流媒体数据，同时也为其他节点提供数据。

4.3 系统性能评估

4.3.1 仿真环境及配置

仿真中我们首先使用工具 GT-ITM[49][50]生成一个具有两个层次的类似 Internet 的 Transit-stub 网络拓扑，分别代表核心路由层和边缘路由层，核心路由层由一个

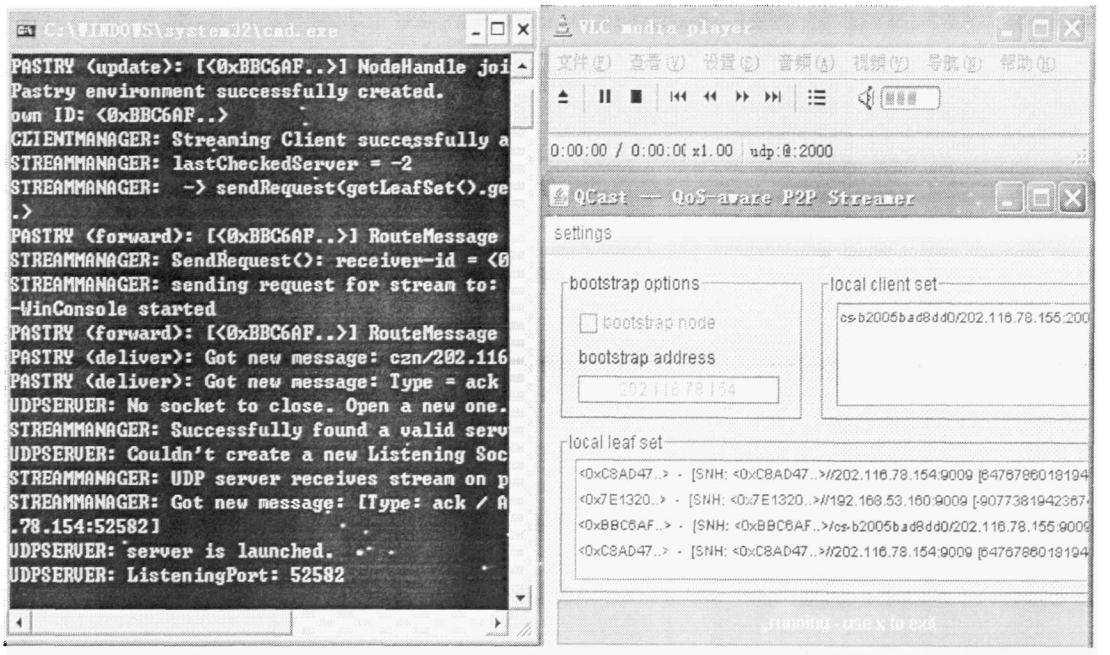

图 4-5 QCast 原型系统运行效果图

中转域（transit domain）组成，它包含 4 个核心路由节点，类似大的 Internet 服务提供商；边缘路由层由 12 个桩域（stub domain）组成，每 3 个桩域附属于一个核心路由节点，每个桩域平均包含 8 个边缘路由节点，类似校园网、一般企业网等。在每次实验中，假定源服务器均连接到一固定的核心路由节点上，其他主机节点加入系统并随机连接到某个边缘路由节点上。对每种性能指标的仿真测试，实验均进行 10 次，实验结果取其平均值。

我们用 FreePastry 作为仿真工具，FreePastry 是一个离散事件模拟器，由美国的 Rice 大学、Microsoft 研究院等共同研究开发的 P2P 网络仿真集成环境，具有开放性好、扩展性强、跨平台等特点，被广泛应用于基于 Pastry 等结构化网络的 P2P 系统的仿真实验。利用 FreePastry 进行网络模拟的整个过程如图 4-6 所示。

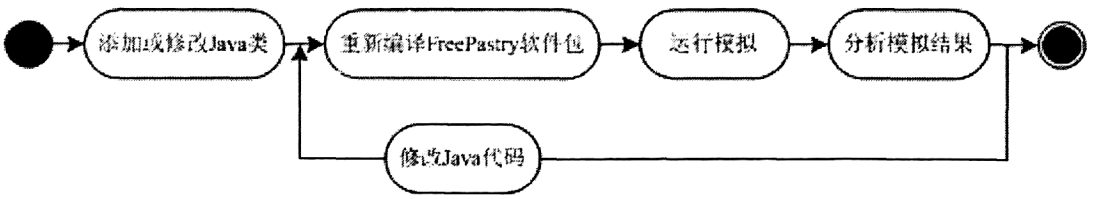

图 4-6 利用 FreePastry 进行网络模拟的过程

4.3.2 仿真实验与结果分析

1. 仿真的性能指标

(1) QoS 满足率：本文在仿真实验设计中将给每个节点随机指派一个 QoS 能力值（一个 0 到 255 之间的值），用 QoS 满足率来表示从根节点到叶子节点之间满足 QoS 能力单调递减的端到端路径所占的百分比。

(2) 数据传输延时：现实网络的数据传输延时是很难模拟的，GT-ITM 生成的网络拓扑就不模拟网络的数据传输延时。本文在仿真实验设计中采用节点的路径长度即节点到源服务器节点的路由跳数（hop number）来反映数据传输延时，这种方式简单有效，在很多文献中都可以见到。

(3) 系统的鲁棒性：本文主要通过在部分节点同时退出的情况下，系统中能够接收到数据服务的节点数目占现有节点总数的比例来衡量系统的鲁棒性。

2. 仿真实验设计

在 QCast 模型的设计中，有一些参数的取值不同可能会导致系统的性能不同。在本文的仿真实验中，节点的 QoS 能力值是随机指派的（源服务器节点除外，源服务器节点永远拥有最大的 QoS 能力值），而节点加入算法中的延时系数 q5 要根据具体的网络环境来调整，由于选择数据发送节点时传输延时是一个非常重要的指标，因此 q5 一般是一个大于或等于 0.4 的值。还有一个参数是每个节点所拥有的邻居节点的最大数目 m，它的取值也会影响到系统的性能。下面取 q5 等于 0.4，m 分别取 8、12、16 时对系统的平均传输路径长度进行仿真测试。

实验结果如图 4-7 所示。对比 3 条曲线，可以看出一个总的趋势，就是 m 值越大，平均传输路径长度越小。然而，m 值越大，节点加入系统时的首次加入时间也会越多，并且从图 4-7 可以看出，m 等于 16 时平均传输路径长度最小，但是与 m 等于 12 时相比性能提高并不大，因此，综合考虑节点加入系统时的首次加入时间，我们认为 m = 12 是比较合适的。以下实验中 m 的取值都为 12。

本文所设计的 QCast 模型是基于树状结构的，同时引入了网状拓扑协议的多邻居节点结构加以优化。同时，在设计 QCast 模型的算法时还考虑了若干 QoS 保证的因素。为了验证 QCast 模型的设计是否优化了系统的总体性能，本文选择了完全基于树状结构的 Scribe-based 系统作为比较对象，Scribe 是一个应用层组通讯系统，在 FreePastry 的软件包中有完整的实现，利用 Scribe 可以很容易地构造出完全基于树状结构的模型，下面就 QCast 和完全基于树状结构的 Scribe-based 系统

的几个主要性能指标设计仿真实验进行比较。

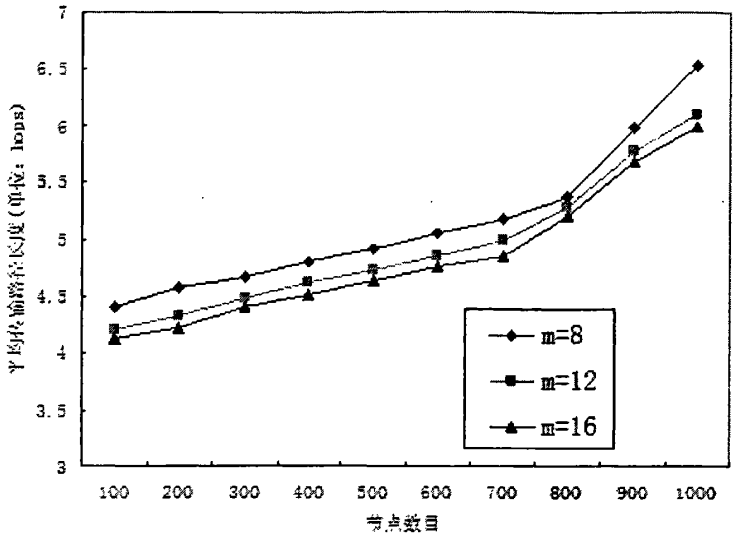

图 4-7 不同 m 取值的平均传输路径长度比较

(1) QoS 满足率

本文针对不同节点规模下系统的 QoS 满足率进行了仿真。图 4-8 显示了节点

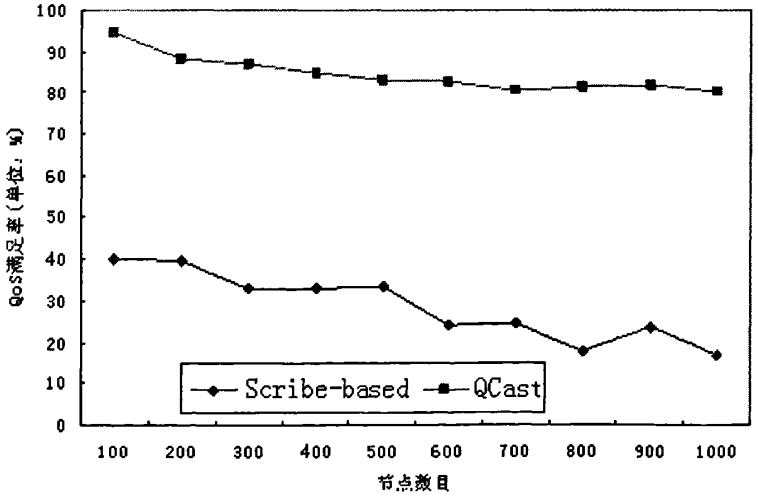

图 4-8 QoS 满足率的比较

规模从 100 到 1000 时 QCast 和 Scribe-based 系统中所有端到端路径的平均 QoS 满足率的变化情况。在计算 QoS 满足率时，只要端到端路径中有一个节点不满足 QoS 能力单调递减的性质，我们就不能保证这个节点以及它下面的所有子节点的 QoS 需求。由图 4-8 可知，Scribe-based 系统构造的组播树中平均只有 17%到 40%的路径满足 QoS 单调递减的性质，这是因为 Scribe 中使用的是 Pastry 的随机 ID 分配算法，在组播树的构建过程中并没有将节点的 QoS 需求考虑在内。而 QCast 构造的组播树的平均 QoS 满足率在 80%到 95%之间，由于在挑选父节点和邻居节点时，我们只是把满足 QoS 单调递减作为弱约束条件，因此 QCast 的组播树中存在一些端到端路径不满足式(3-2)。

(2) 数据传输延时

图 4-9 显示了节点数从 100 到 1000 变化时 QCast 和 Scribe-based 系统的平均传输路径长度（以跳为单位）的变化趋势。从图 4-9 可以看出，当节点数小于约 140 个时，QCast 的平均传输路径长度稍稍大于 Scribe-based 系统；当节点数大于约 140 个时，QCast 的平均传输路径长度则开始小于 Scribe-based 系统；当节点数到达 1000 时，QCast 的平均传输路径长度大约比 Scribe-based 系统的少 2.5 个跳（hops），并且随着节点规模的增大，QCast 的曲线变化是比较平缓的。

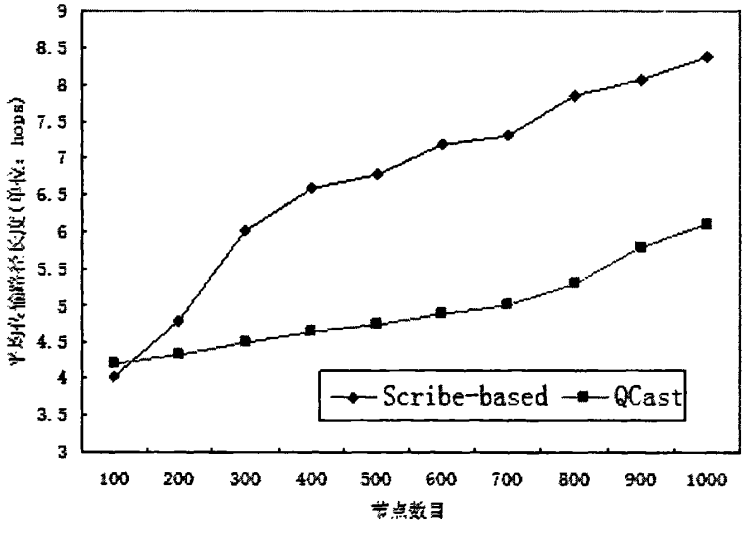

图 4-9 平均传输路径长度比较

当节点规模较小时，系统相对稳定，QCast 的多邻居节点结构作用不明显，相反，带来的控制开销比 Scribe-based 系统的要大。随着节点规模的不断增大，系统的稳定性也会受到影响，组播树深度的增大和网络拥塞会导致一些节点的数据传输速率下降。当系统中某个节点的数据传输速率下降时，在 QCast 中可以通过多个邻居节点获取数据，使得节点到源服务器节点之间存在多条传输路径，相对减少了数据传输延时；而在 Scribe-based 系统中，每个节点只有一个父节点，节点到源服务器节点之间也只有一条传输路径，父节点数据传输速率的下降会影响它下面所有的子节点的数据传输，这样就相对增加了数据传输延时。总的来说，随着节点规模的增大，QCast 的多邻居节点结构不但可以保证节点的下载带宽，同时还减少了数据传输延时，使得节点的平均数据传输延时增长比较平缓，因此 QCast 的可扩展性比 Scribe-based 系统更好。

(3) 系统的鲁棒性

为了分析两个系统的鲁棒性，本文模拟高动态的网络即同时有多个节点离开的网络，并且用系统中能够接收到数据服务的节点数目占现有节点总数的比例来衡量系统的服务质量。仿真实验模拟了 1000 个节点加入系统后有 20、40、60、80、100、120、140、160、180、200 个节点同时离开系统的场景。

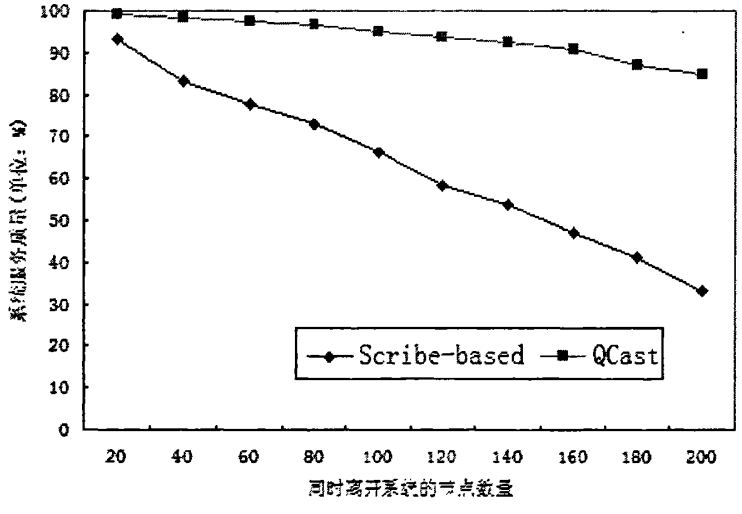

图 4-10 部分节点退出对系统服务质量的影响

从图4-10可以看出，当有20个节点同时离开系统时，QCast约有99.2%的节点可以获得数据，Scribe-based系统约有93.4%的节点可以获得数据。随着离开节点数目的增加，两个系统的服务质量都呈下降趋势，但是QCast的下降幅度比Scribe-based系统的要小得多。当同时离开系统的节点数目达到200时，QCast中仍然有约85%的节点可以获得数据，而Scribe-based系统仅有34%不到的节点可以获得数据。

究其原因，是因为Scribe-based系统中每个节点只有一个数据发送节点，父节点的离开将会影响它下面所有子节点的数据传输，子节点只有通过重新加入系统或者重新寻找父节点才能获取数据；而在QCast中，当节点离开系统时，子节点可以通过邻居节点继续获取数据，尽量避免了数据传输中断的情况。因此，QCast比Scribe-based系统具有更好的鲁棒性。

3. 仿真结果分析

下面对以上的仿真实验进行小结分析：

(1) 由于使用了QoS保证的节点ID分配算法，在组播树的构建过程中充分考虑了节点的QoS能力，QCast在服务质量保证上比Scribe-based系统要好很多。

(2) 当节点规模较小时，QCast在数据传输延时性能上比Scribe-based系统稍稍差点，但是随着节点规模的不断增大，QCast在数据传输延时性能上逐渐赶上并超过Scribe-based系统，并且在这个过程中所有节点的数据传输延时变化比较平缓，这表明了QCast在数据传输延时和可扩展性方面做出了较好的平衡，在数据传输延时的性能上明显优于Scribe-based系统。

(3) 在动态的网络环境下，当大量节点同时离开系统时，完全基于树状结构的Scribe-based系统中的很多节点会发生数据传输中断的情况，而引入了多邻居节点结构的QCast中的大部分节点依然能够获取数据，这表明了QCast比Scribe-based系统具有更好的鲁棒性。

这些结论表明了QCast模型对几个主要的性能指标都进行了优化，并在这些性能指标之间做出了较好的平衡，因此QCast模型的设计具有一定的正确性和有效性。

4.4 小结

本章首先给出了 QCast 模型的系统原型设计，然后用 Java 伪代码对其核心算法进行了描述，最后借助 FreePastry 仿真器为 QCast 模型的原型系统设计了仿真实验，并与完全基于树状结构的 Scribe-based 系统中相对应的算法进行了比较。仿真实验的结果表明了 QCast 模型对系统的服务质量保证、数据传输延时、鲁棒性等性能指标都进行了优化，并在主要的性能指标之间做出了较好的平衡，从而证明了 QCast 模型的设计具有一定的正确性和有效性。

第5章　结论与展望

本文专注于基于 P2P 技术的 QoS 保证的流媒体分发服务的研究。下面对全文工作进行总结，并展望了进一步的研究方向。

5.1 总结

利用 P2P 技术的优势在 Internet 上提供大规模的流媒体分发服务是一项很有意义的事，由于 P2P 技术和流媒体应用各自的特殊性，基于 P2P 的流媒体分发技术的研究面临着诸多挑战。针对这些挑战，在研究了若干种典型的 P2P 流媒体分发模型后，本文提出了一个 QoS 保证的 P2P 流媒体分发模型——QCast。QCast 是基于单组播树模型的，同时引入了网状拓扑协议的多邻居节点结构；它既具有单组播树结构控制简单、维护代价小的优点，又结合了网状结构中多个发送者的数据传输方式，从而有效减少了传输延迟，提高了系统的鲁棒性。此外，本文在 QCast 的分发体系和数据传输调度的设计上还研究了若干提供 QoS 保证的机制。

本文的研究工作和成果主要表现在以下几个方面：

1. 在 DHT 的基础上设计出 QCast 模型的系统框架，充分利用了 DHT 的路由机制、消息触发机制和故障恢复机制；同时，通过改造 DHT 中节点 ID 分配的随机性，提供了一种 QoS 保证的节点 ID 分配算法，这样就利用了 P2P 网络中节点的异构性，把资源较好且比较稳定的节点放在组播树中靠近根节点的地方，使得组播树的构建过程具有“QoS 感知”的性质。
2. 在设计节点的加入算法时，在采用经典的从源服务器节点出发沿组播树搜索非饱和节点的基础上，综合考虑了节点的 QoS 能力和传输延迟，设计优先级函数来选择节点的父节点和邻居节点，一方面减少了因节点退出而导致的传输延迟，另一方面保证了节点之间具有较大的可用带宽，从而可以保证流媒体传输的实时性和连续性。
3. 针对 QCast 的混合结构模型，提出了推拉结合的数据传输调度算法。在第一阶段，节点加入系统后，被动地接受父节点的数据传送，即采用推机制。在第二阶段，当节点的数据下载速率小于其数据播放速率时，节点主动地向多

个“活动节点”请求数据，采用拉机制获取数据，同时充分评估每个“活动节点”的工作状态以及端到端链路状况；当节点的数据下载速率又大于或等于其数据播放速率时，节点间的数据传输又切换到推机制。该算法能够减少节点间的传输延迟，提高节点的下载带宽，从而保证了流媒体的播放质量。

4. 实现了 QCast 模型的原型系统，并针对 QCast 模型的相关性能指标设计了仿真实验，实验结果表明了 QCast 模型对系统的服务质量保证、数据传输延时、鲁棒性等性能指标都进行了优化。

5.2 未来展望

相比传统的流媒体分发技术，P2P 流媒体技术在可扩展性、容错性、经济性等方面具有“天生”的优势，能够满足 Internet 上大规模流媒体分发服务的需求，因此必将具有美好的应用前景。由于时间仓促和能力所限，本文的研究工作只是 P2P 流媒体研究领域中的一小部分，部分研究成果在深度上和广度上也有待于继续下去。在研究和实验的过程中，总结出 QCast 模型在以下几个方面还有待进一步的研究与改进：

1. 目前只是初步完成了 QCast 的原型系统，还没有办法在真实和复杂的网络环境下进行大规模的系统测试和性能评估。因此，今后的工作首先应该不断完善 QCast 的原型系统，为后续的算法研究提供一个真实的测试和调优平台。
2. NAT 穿透问题。在 Internet 的各个自治域内，不同的局域网有不同的本地安全策略，局域网和外网的通信就涉及到 NAT 穿透的问题。QCast 中采用了 FreePastry 默认的 NAT 穿透机制，能够满足基本的 NAT 穿透需求，但针对多种多样的局域网类型还有待进一步的研究和改进。
3. Internet 的异构性问题。Internet 上不同的接入商（ISP）之间的带宽存在差异，在跨 ISP 通信时，IP 包的延时和丢包率比较高，目前 QCast 模型的设计还没有考虑到不同 ISP 网络的差别，在跨网段进行流媒体分发时可能会有一定的影响。因此，在以后的研究中应该考虑这些因素加以改进。

参考文献

[1] 流媒体世界网. http://www.lmtw.com

[2] 中国互联网络信息中心. 中国互联网络发展状况统计报告(2006/1). http://www.cnnic.com.cn/images/2006/download/2006011701.pdf, 2006

[3] S. Deering, D.R. Cheriton. Multicast routing in datagram internetworks and extended LANs. ACM Transactions on Computer Systems, May 1990, 8(2):85-110

[4] S. Deering, D. Estrin, D. Farinacci, et al. An architecture for wide-area multicast routing. ACM SIGCOMM Computer Communication Review, October 1994, 24(4):126-135

[5] C. Diot, B. N. Levine, B. Lyles, H. Kassem and D. Balensiefen. Deployment issues for the IP multicast service and architecture. IEEE Network magazine special issue on Multicasting, 2000, 14(1): 78-88

[6] J. Kangasharju. Internet Content Distribution. PhD thesis, University of Nice Sophia Antipolis/Institut Eurecom, April 2002

[7] Napster Website. http://www.napster.com

[8] Gnutella Website. http://www.gnutella.com

[9] KaZaA Website. http://www.kazaa.com

[10] FreeNet Website. http://freenetproject.org

[11] Morpheus Website. http://www.morpheus.com

[12] Bittorrent Website. http://www.bittorrent.com

[13] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, November 2001, 329-350

[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In SIGCOMM'01:

Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA: ACM Press, 2001, 149-160

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In SIGCOMM'01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, New York, NY, USA: ACM Press, 2001, 161-172

[16] D. Malkhi, M. Naor and D. Ratajczak. Viceroy: A scalable and dynamic emulation of the butterfly. In Proceedings of the twenty-first annual symposium on Principles of distributed computing, New York, NY, USA: ACM Press, 2001

[17] E.Adar and B.Huberman. Free riding on Gnutella. Technical Report, Xerox PARC, 2000

[18] 龚海刚，刘明，毛莺池等. P2P 流媒体关键技术的研究进展. 计算机研究与发展，2005，第 42 卷，第 12 期，2033-2040

[19] ESM Website. http://esm.cs.cmu.edu

[20] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming Live Media over a Peer-to-Peer Network. Technical Report, Stanford University, August 2001

[21] S. Banerjee, B. Bhattacharjee and C. Kommareddy. Scalable Application Layer Multicast. In SIGCOMM'02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, Pittsburgh, Pennsylvania, August 2002

[22] J. Jannotti, D. Gifford, K. Johnson, et al. Overcast: Reliable Multicasting with an Overlay Network. In Proceedings of 4th Symposium on Operating Systems Design Implementation, 2000

[23] V.N. Padmanabhan, H.J. Wang, P.A. Chou and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. In Proceedings of International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Miami Beach, FL, USA, May 2002

[24] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh. SplitStream: High-bandwidth multicast in a cooperative environment. In

Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP), Lake Bolton, New York, October 2003

[25] J. Ganesh, A.-M. Kermarrec and L. Massoulie. Peer-to-peer membership for gossip-based protocols. IEEE Transactions on Computers, February 2003, 52(2):139-149

[26] X. Zhang, J. Liu, B. Li and T.-S. P. Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Live Media Streaming. In Proceedings of IEEE INFOCOM, Miami, FL, USA, March 2005, 2102-2111

[27] D.A. Tran, K.A. Hua and S. Sheu. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In Proceedings of IEEE INFOCOM, San Francisco, CA, USA, March 2003, 1283-1292

[28] M. Zhang, Y. Tang, L. Zhao, J.G. Luo and S.Q. Yang. GRIDMEDIA: a multi-sender based peer-to-peer multicast system for video streaming. IEEE International Conference on Multimedia and Expo, 2005, 614-617

[29] D. Kostic, A. Rodriguez, J. Albrecht, et al. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In Proceedings of the 19th ACM Symposium on Operating System Principles, Bolton Landing, NY, USA, 2003, 282-297

[30] M. Hefeeda, A. Habib, B. Botev, D. Xu and B. Bhargava. PROMISE: peer-to-peer media streaming using collectcast. In Proceedings of the 11th ACM International Conference on Multimedia, 2003

[31] R. Rejaie and S. Stafford. A framework for architecting peer-to-peer receiver-driven overlays. In Proceedings of NOSSDAV 2004, NewYork, NY, USA: ACM Press, June 2004, 42-47

[32] Y Guo, K. Suh, J. Kurose and D. Towsley. P2Cast: Peer-to-peer Patching Scheme for VoD Service. In Proceedings of 12th World Wide Web Conference (WWW), Budapest, Hungary, May 2003

[33] S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In Proceedings of IEEE INFOCOM, San Francisco, CA, USA,

March 2003, 1521-1531

[34] D. Xu, H.-K. Chai, C. Rosenberg and S. Kulkarni. Analysis of a Hybrid Architecture for Cost-Effective Streaming Media Distribution. In Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), Santa Clara, CA, January 2003

[35] FreePastry Website. http://freepastry.rice.edu/FreePastry

[36] M. Castro, P. Druschel, A.-M. Kermarrec and A. Rowstron. SCRIBE: a large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications (Special issue on Network Support for Multicast Communications), 2002, 20(8): 1489-1499

[37] Z. Li and P. Mohapatra. Qron: QoS-aware routing in overlay networks. IEEE Journal on Selected Areas in Communications, 2004, 22(1): 29-40

[38] B.G. Rocha, V. Almeida, and D. Guedes. Increasing QoS in selfish overlay networks. IEEE IC, 2006, 10(3): 24–31

[39] Z. Li. Resiliency and quality-of-service (qos) support in multicasting and overlay networks. Ph.D. dissertation, Davis, CA, USA, 2005, adviser-Prasant Mohaptra

[40] S. Chen, K. Nahrstedt and Y. Shavitt. A QoS-aware multicast routing protocol. IEEE Journal on Selected Areas in Communications, 2000, 18(12): 2580-2592

[41] T. Guang and S.A. Jarvis. Stochastic Analysis and Improvement of the Reliability of DHT-Based Multicast. In Proceedings of IEEE INFOCOM, Anchorage, Alaska, USA, May 2007, 2198-2206

[42] M. Brogle, D. Milic and T. Braun. QoS Enabled Multicast for Structured P2P Networks. In Proceedings of workshop on Peer-to-Peer Multicasting 2007, January 2007

[43] EuQoS Website. http://www.euqos.org

[44] D. Pendarakis, S. Shi, D. Verma, et al. ALMI: An Application Level Multicast Infrastructure. In Proceedings of 3rd USENIX Symposium on Internet Technologies and Systems, 2001

[45] 陈勇．基于应用层组播的流媒体主动修补技术研究．硕士论文，郑州大学，2006

[46] Tu Yicheng, Lei Shan. Towards cost-effective on-demand continuous media service: A peer-to-peer approach. Tech.Rep:2002-14, Purdue University, 2002

[47] M.H. Firooz, K. Ronasi, M.R. Pakravan, et al. A multi-sender multicast algorithm for media streaming on peer-to-peer networks. Computer Communications, July 2007, 30(10): 2191-2200

[48] VLC Website. http://www.videolan.org

[49] K. Calvert, M. Doar and E. Zegura. Modeling Internet Topology. IEEE Communication Magazine, 1997, 3: 160-163

[50] GT-ITM Website. http://www-static.cc.gatech.edu/projects/gtitm

附录

攻读学位期间发表的与学位论文相关的学术论文：

1. Zhinuan Cai and Xiaola Lin. QCast: A QoS-aware Peer-to-Peer Streaming System with DHT-based Multicast. In Proceedings of the 3rd International Conference on Grid and Pervasive Computing. Lecture Notes in Computer Science. May 2008. （第一作者，已录用，EI Index）

攻读学位期间主要参与的科研项目：

1. 国家自然科学基金项目，片上网络系统的拓扑和路由关键问题研究，编号 60773199

致谢

首先，我要衷心感谢我的导师林小拉教授。在两年硕士研究生的学习与生活中，林老师不仅给我创造了一个良好的学习和科研环境，还给予了我无微不至的关心与指导。无论在学习上还是在生活上，林老师总是不遗余力地帮助我、指导我。不仅如此，林老师开明的思想、严谨的治学作风、深厚的学术水平和令人钦佩的师德，让我钦佩不已，值得我用一生去学习。在此，我衷心感谢林老师给予我的谆谆教诲和大力支持，让我得以顺利完成硕士研究生课题。

其次，我要感谢实验室的龚亚东博士、余士元师兄、李江山师兄、谭理发师兄、潘健和其他同仁，无论在学习研究还是在工作生活中，他们都给予了我宝贵的建议与帮助，他们对本文的写作给出相当宝贵的指导意见。他们热忱的工作态度、渊博的知识、活跃的思维，也是我学习的榜样。

同时，也向所有关心和帮助过我的信息科学与技术学院的老师和工作人员表示由衷的感谢。感谢我的宿友，感谢班上的同学们，谢谢他们给了我两年共处的美好时光。

在此，谨对我的父母及家人表示最崇高的敬意，感谢他们这么多年来对我无私的照顾与支持、信任与鼓励，他们永远是我前进的不竭动力。

最后，再次感谢所有关心、帮助和支持过我的老师、同学和朋友们。