基于对等网络的流媒体传输系统技术研究

Theoretical Analysis and System Study on Peer-to-Peer Media Streaming

（申请清华大学工学硕士学位论文）

培 养 单 位 ： 计算机科学技术系

学　　科 ： 计算机科学技术

研 究 生 ： 董 海 韬

指 导 教 师 ： 郑 纬 民 教 授

二〇〇五年五月

基于对等网络的流媒体系统技术研究

董海韬

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

摘　要

P2P 技术的迅速发展使其成为构建广域网中大型分布式系统的有力工具。P2P 流媒体系统作为 P2P 技术的重要应用之一也成为当前分布式系统领域的一个研究热点。本文针对 P2P 流媒体研究中的一些基本问题进行了深入研究，包括 P2P 覆盖网络设计、根据底层网络状况进行拓扑优化、结点信息收集算法、根据传输带宽选择合适数据发送节点及动态进行数据传输调度来得到最小延迟等。

本文的主要研究内容和贡献如下：

（1）提出了流媒体应用底层平台的非结构化 P2P 覆盖网协议 CAPU。这个系统解决了如何利用节点异构性来实现拓扑结构维护，使得 P2P 系统的结点可以根据各自能力以较低的网络开销信息收集量，并实现了高效的信息路由查找策略。

（2）设计了一个根据节点能力自适应的半结构化拓扑结构维护算法。这个算法保证覆盖网维持一个高吞吐的、负载平衡的、低网络直径并且具有高容错能力的半结构化拓扑结构。

（3）提出了主动信息发布扩散算法加两阶段搜索算法。根据节点级别决定节点掌握系统中文件信息的多少，并在此基础上设计了一个第一阶段为随机走步，第二阶段为无冗余广播搜索算法。这个搜索算法结合了非结构化 P2P 路由中两种基本模式的优点，在非结构化结构中实现了可收敛的高效信息查找机制。

（4）提出了根据底层网路状况自调整的拓扑优化算法。这个拓扑优化算法使得每个节点能根据从这些节点列表中选择网络状况较好的节点建立连接，从而为上层流媒体应用提供较好的基于覆盖网的网络连接。

（5）提出了数据发送节点动态选择算法和数据调度算法。根据网络状况和数据传输速率等因素来动态选择出“活动数据发送节点集合”，从这个集合中的节点请求媒体数据来获取最优的数据传输速率。并在数据发送节点间，根据对方服务能力，数据带宽等因素来分派每个数据发送节点所发送数据的次序，从而获得最小的数据接受延迟。

关键词：覆盖网　非结构化 P2P　分布式多媒体系统　流媒体组播　调度优化算法

Abstract

The rapid development of P2P technology makes it as one of the most disruptive tools for the construction of large-scale distributed system over Internet. Therefore, as one of the most perspective applications, P2P streaming system has become one of the hotspot of P2P research. This paper conducts in-depth research on several basic issues of P2P streaming system, including how to design of P2P overlay network, how to optimize the topology based on underlay network, how to collect node information, how to select data sending node set, and how to schedule dynamic data set between data sending nodes to obtain minimum delay.

The main issues and contributions of this paper is as following:

(1) A new P2P overlay protocol CAPU as the infrastructure of streaming application, which harness the heterogeneity of the nodes to maintain topology, and make it possible for nodes in P2P system to collect information according their capacity.

(2) A quasi-hierarchical capacity-aware topology protocol that ensure a approximately optimal overlay topology in terms of high system throughput, load balance, low diameter, and high resilience to failure.

(3) Proactive file index propagation exploit the capacity-aware topology, thus make full use of the quasi-hierarchical topology to pro-vide cache for query in advance, and shorten length of query path.

(4) Two-stage search algorithm take advantage of the capacity-topology and file index strategy, the first stage bias random search message to high capacity node dynamic-adaptive to query load, the second stage introduces Multi-Point Relays (MPRs) to multicast query request among highest level nodes, reducing the number of re-transmission packet compared with flooding.

(5) An algorithm to dynamic select active data sending nodes based on network condition and an optimal media data assignment algorithm, which results in minimum buffering delay in the consequent streaming session.

Key words: Overlay Network Unstructured P2P System Distributed Multimedia System Streaming Multicast Scheduling Optimization Algorithm

目录

第一章　引言

近年来，随着大规模存储、高性能工作站以及宽带网络等技术突飞猛进的进步，互联网上传输的信息不再只是文本、图像，各种各样的多媒体通信（Multimedia Communication）服务从技术上和经济上成为可能。多媒体通信可以集成视频、音频、文本及图像为一体，为用户提供更为丰富的使用体验，因此得到了越来越为广泛的应用。特别是流媒体技术的出现和普及，使得声音、影像或动画等时基媒体可以由音视频服务器向用户计算机的连续、实时传送，用户不必等到整个文件全部下载完毕就可观看到媒体，从而满足了实时交互的需要。然而，目前的互联网上的基于服务器/客户机模型的流媒体服务还远没有达到可以与传统的 WWW、FTP 等应用相比拟的服务质量，尤其在可扩展性（Scalability）和容错性（Fault-tolerance）等方面还远远不能满足应用的需求：

1. **可扩展性**（Scalability）是一个多方面的概念集合，包括了对计算资源的可扩展、对应用规模的可扩展以及对技术换代的可扩展。这里我们主要考虑的是对应用规模，尤其是对多媒体资源和用户数量的可扩展性。到目前为止，流媒体文件的数量的增长速度有逐年加剧的趋势，这样要求提供多媒体服务的计算机系统提供巨大的存储空间。其次，流媒体服务比普通的文本和图片服务需要更高的计算资源和带宽资源，这使得为了能够支持大量的同时在线用户，而绝大多数现有的多媒体服务器为终端用户提供媒体服务的带宽上限都不超过 50 Mbps，离支持大规模用户的应用目标差距甚远。
2. **容错性**（Fault-tolerance）是指计算机系统在遇到系统硬件或软件错误的时候仍能以可接受的服务质量继续工作的能力。计算机系统主要通过复制（Replication）、冗余（Redundancy）和自我修复（Self-stabilization）等方法来实现容错。显而易见，中心的多媒体服务器是整个多媒体服务系统中的单点故障（Single Point of Failure）部件，这种服务器/客户机模型决定了系统的容错能力不高，并易于遭到攻击。
3. **鲁棒性（Robustness）**是指当一个控制系统中的参数发生摄动时系统能否保持正常工作的一种特性或属性。研究者们发现，包括 WWW 在内的 Internet 访问具有自相似的特性 [7]，自相似序列的一个重要特

性是多个序列的叠加与原序列具有类似的分布，而不像泊松序列，多个序列叠加的结果趋于平滑。换句话说，服务并发数的增加并不意味着服务总量是相对平稳的，实际上，短时的突发访问量高峰是 Internet 访问的基本特征。另一方面，某些突发的事件可能造成使用服务的人数的短时指数级上升，这往往被称为 Flash crowd 现象，原词出现在 1971 年的一部科幻小说[8]中，意指数之千计的人们回到过去去目睹历史事件的发生。例如 2001 年 9 月对美国的恐怖袭击发生后，www.cnn.com 等新闻站点的访问量骤然上升，以致很多站点无法访问。因此，成功的 Internet 服务必须能够在过量访问下，实现得体的性能下降(Graceful performance degradation)，而不是立即崩溃。

通过以上分析，我们可以看到，目前流媒体服务所广泛采用的服务器/客户机模型不能提供高可扩展能力和高可用的服务。为了解决这些问题，近年来研究界和工业界提出了多种解决方案，比较重要的有内容分发网络（Content Delivery Network CDN）和 IP 广播（IP Multicast）等。但是，这些解决方案的共同特点是需要有专门的硬件支持，比如 CDN 需要在全球各地部署多个 CDN 服务器，通过服务器之间协同工作，分发多媒体数据；而 IP 广播更是需要修改目前 Internet 的路由机制，广泛部署复杂的支持广播功能的路由器。这样不仅耗资巨大，而且并不能从根本上解决我们上面提出的问题。

本文从研究在对等网（Peer-to-Peer Network）上架构多媒体系统。最近几年，对等网络（Peer-to-Peer，简称 P2P）引起了包括普通网络用户和科研领域人员在内的越来越多的人的广泛关注。Peer-to-Peer 模式（对等计算，亦简称 P2P）的核心思想是通过参与系统的节点之间的直接交互来实现信息资源和服务的共享。一方面 P2P 突破了传统的客户端/服务器模式，强调结点之间的“对等性”，即 P2P 系统中每一参与结点兼有服务器和客户端两种身份，在利用其他结点上的资源之同时也为其他结点提供服务。这就使 P2P 系统的服务能力能够随需求的增长而自然增长，具有“与生俱来”的可扩展性，能够解决传统客户端/服务器结构中服务器过载和资源瓶颈的问题。另一方面，P2P 系统采用结点自组织的方式工作，强调无中心的结构，并且很好地适应了结点随机加入和退出的动态性，因而在容错性、数据高可用性和抵抗攻击方面具有不可替代的优势。因此，基于 P2P 网络构建多媒体系统，能够从根本上

解决基于服务器/客户机模型的系统的不可扩展性、低容错性和低鲁棒性。

在本章以下的部分中，我们首先介绍当前基于互联网流媒体服务的体系结构和传输协议，接着讨论构造基于对等网络的流媒体系统的关键技术和主要挑战，最后介绍本文的组织结构和主要贡献。

1.1 流媒体应用介绍

流媒体（Streaming Media）是指视频、声音和数据通过实时传输协议以连续流方式顺序从源端向目的地传输，目的地只需接收到一定数据缓存后就可以立即播放的多媒体应用。在采用流式传输的系统中, 用户不必等到整个 A/V 文件全部下载完毕, 而只须经过几秒或十数秒的启动延时, 即可进行观看。当声音等时基媒体在客户机上播放时, 文件的剩余部分将在后台从服务器内继续下载。与传统的“先下载、再播放”机制相比, 流媒体技术不仅使启动延时成十倍、百倍地缩短, 而且不需要很大的缓存容量。

互联网上视频流媒体技术应用近年来增长迅速，2000 年网上访问流媒体的人数增加 65%，流媒体技术在世界范围内得到应用。流媒体技术可广泛用于网上新闻发布、在线直播、网络广告、远程教育、实时视频会议等，目前应用最直接的是网上直播。商业网站利用流媒体播放新闻，开展音乐直播和点播服务，企业和机构采用点播和流媒体进行员工培训、信息发布、公司介绍等，从而提高效率，节约开支。基于流媒体的应用转变了传统互联网呆板的内容表现形式，具有强视觉冲击力的视频节目成了人们进入宽带网络的最重要的应用之一，多媒体互动成了人们对宽带网络未来发展的寄托。传统影视媒体市场与宽带网络的应用服务相融合，产生了宽带 VOD、在线音乐、远程教育、宽带收费电视等新的应用。

目前应用的流媒体系统大体都可以分成四部分：媒体编码器、媒体文件存储器、媒体服务器和媒体播放器。各部分功能如下：

1. 媒体编码器：将原始的媒体文件或摄像头采集进来的实时媒体数据制作成适合网络传输的文件格式（流格式），然后将流文件存储在媒体文件存储器中，或直接送到流媒体服务器。
2. 媒体文件存储器：存储流格式的媒体文件，一般采用 SCSI 硬盘或

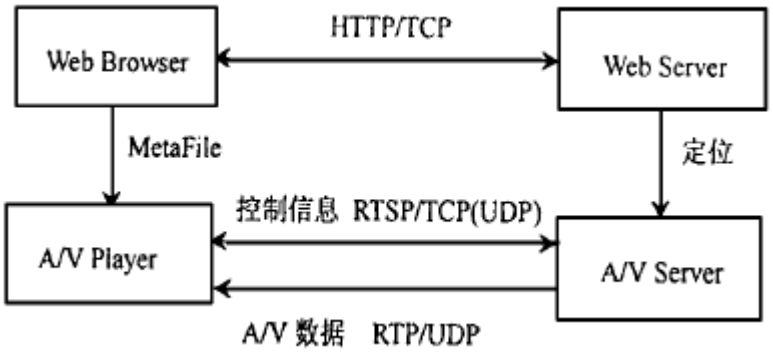

图 1.1 W eb 上的流式传输原理示意图

磁盘阵列。

3. 媒体服务器：响应调度服务器从 WEB 服务器转过来的用户请求，通过网络传输协议将流格式的文件传到用户桌面。
4. 媒体播放器：接收网络媒体数据，并在本地播放。

提供流媒体的服务器成为流媒体服务器，又称连续媒体（Continuous Media CM）服务器，广泛应用在流媒体新闻点播、远程教育、电子商务以及商业培训等方面。实际应用中的流媒体服务器，多采用 Web 服务器+媒体服务器的形式。图 1.1 是一个简单的单服务器示意图，用户通过客户端的 Web Browser 访问 Web Server 上的连接，发出控制信息，然后再由 Web Server 将此控制信息发给媒体服务器。在流式传输中, 一般采用 HTTP/TCP 来传输控制信息, 而用 RTP/U DP 来传输实时数据。Web 服务器和 Web 浏览器通过 M IM E 标记媒体类型, 浏览器通过 M IM E 识别出流媒体类型后, 再调用 P lug- in 或助手应用程序(Helper) 进行处理。

流媒体文件的传输和播放具有实时性的限定条件，如果在传输过程中不满足这个实时性的条件，客户端的播放就会出现中断、延迟或抖动的现象。另外，媒体文件多数都比较大，一个播放时间两小时播放带宽 4Mb/s 的 MPEG-2 视频文件有 3.6GB（Gigabytes）。因此，单机流媒体服务器很难支撑大规模的服务。因此出于负载分担和支持更多用户考虑，出现了集群流媒体服务器，通过一台管理服务器来根据各服务器负载状况决定将用户请求发往当前负载最低的服务器，管理服务器还负责流媒体文件管理，数字版权管理等。流媒体服务的门户一般仍然为 Web 服务器。图二是一个多机服务器模型。

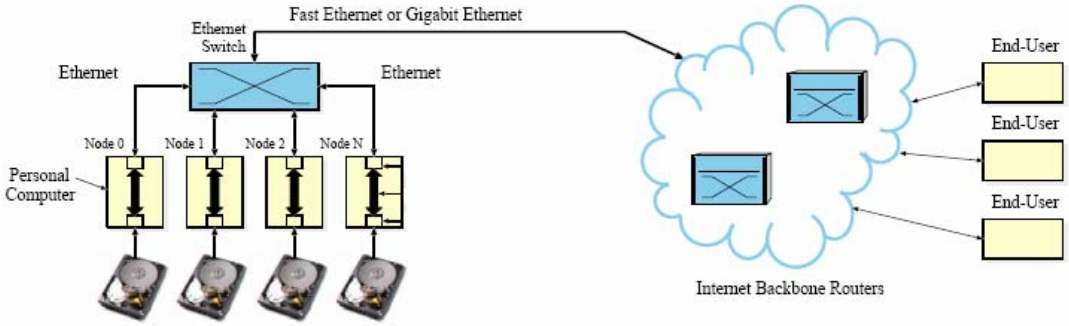

图 1.2 流媒体服务集群示意图

1.2 基于对等网络技术构建流媒体系统

一般认为 P2P 系统具备的特征包括：大规模、无中心（decentralization）、自组织、可扩展性以及高度动态性和异构性的环境。P2P 系统由网络中为数众多的参与结点组成，其数量一般在十万以上，甚至具有百万或百万以上的量级。每个参与结点具有一定的服务能力，单个结点的资源和能力可能非常弱小，譬如仅是一台普通桌面电脑的空闲资源。P2P 系统设计的关键在于让单个能力弱小但总数巨大的结点群体通过精巧的协作方式，提供强大的聚合服务能力。由于结点数量众多，整个系统的资源总量（如计算能力、存储空间、聚合带宽等）是相当可观的，而且能随着结点不断加入而自我扩展。通过合理的结构和协作算法，P2P 系统能够均衡地利用各个结点的资源，保证每个结点的负载都不超过其有限的服务能力，同时整体达到相当巨大的服务能力。随着 Napster，Gnutella，KaZaa，和 BitTorrent 等软件得到日益广泛的应用，对等网络技术在文件共享方面已经取得巨大的成功。然而，而对于本文所集中讨论的基于对等网络的实时流媒体技术而言，还没有得到研究界应有的关注。

通过本章前面部分的论述，我们已经看到，流媒体系统的核心是如何实现一个经济、高效的媒体组播通信系统。下面我们先介绍一下什么是组播通信。在某些时候下，我们会遇到一点到多点或多点到多点通信的情况，即群组通信。在群组通信中，许多结点与同一群组中的其它结点同时交换信息。例如一个有许多人参加的网络电话会议，系统需要把发言者的声音

同时发送给其他与会者。其它应用包括在线流媒体和有效的大量数据的散发。在以上这些群组通信的应用中，组播是一个实现数据从一个结点到多个结点的高效散发的关键组件。但是，尽管网络层的 IP 组播已经被提出了将近 10 年（例如[23][24][25][26][27]），由于没有广泛的部署和跟踪组成员的难题，使用 IP 组播技术的应用很少。为了解决这个问题，近年来提出了一些替代方案，统称为应用层组播或端到端组播。这些组播使用网络层的单播，通过建立一个覆盖网络，在应用层实现组播。

应用层组播具有许多网络层组播不具备的优点。首先，由于不需要路由器的支持，应用层组播可以在现有的基础上渐进部署。其次，应用层组播相对与网络层组播来说更灵活，能够适应上层应用不同的散发要求。[28]认为把组播的功能放置在系统的底层（例如网络层）可能是冗余的或者，和导致的成本比较，是没什么价值的。这部分是因为底层完成这些功能的低效，部分是因为底层缺乏关于上层的知识。例如，为了解决散发树中的链路具有不同的带宽的问题，一个常用的方法就是降低低带宽链路的承载的内容的质量。然而，尽管这种方法对流媒体是适当的和有利的，但对需要每一比特都正确的传输来说（例如传输软件）却是不可接受的。显然，面向应用的组播应当使用覆盖层，使用应用层的组播而不是网络层的组播。

因此，应用层的组播成为当前的一个研究热点，提出了许多可扩展的组管理和可扩展的、可靠的消息传播[29][30]。对这些系统来说，仍存的挑战依然是建立一个支持可扩展和容错的基础结构，同时保证低延迟和网络资源的有效利用。换句话来说，为了实现高效的基于覆盖层的组播系统，必须解决以下三个关键问题：

什么是适当的组播结构？另外，如何在覆盖网络上，特别是以一种分布式的方法，构建出这样一个结构？

覆盖网络如何管理数量众多的参与结点，特别是参与结点在能力和行为方面存在巨大的差别的时候？

如何让组播算法和覆盖网络自适应多变的互联网环境，例如变化的链路、拥塞、和出错？

第一个问题常常被归结为在覆盖网络上构造和维护一个高效容错的生成树，因为生成树是一个天然的适合组播的结构。超越单棵树的结构，

近来的对等网络方面的研究进一步利用树中兄弟之间的“垂直链路”来克服构造树的难题。第二个问题在 IP 的组播中称为“成员管理”。由于系统的巨大规模和分布式的特点，这个问题在覆盖网络和对等网络中变得越发复杂。事实上，覆盖网络有时候仅仅由志愿结点组成，我们无法依赖一个可靠的、强大的结点来管理组成员。这时，使用前几章中的分布式的、自组织的结构来管理组结构是适当的。第三个问题可以通过不停地测量覆盖网络的拓扑，并且采用自适应于拓扑变化的算法来解决。覆盖网络常常认为覆盖网络中两个结点之间的链路是直接连接的、独立的，不管它们在实际上是否共享物理链路或路由器。因此通过测量底层的“黑盒”（组要是覆盖网络中结点间的可联通性和带宽），覆盖网络能够感知底层拓扑的变化，进一步提高性能。

1.3 技术挑战和研究目标

P2P 流媒体系统与 P2P 文件共享系统最大的不同在于对等节点间数据共享的模式：传统文件共享系统是“下载后运行”（Play-after-Downloading）模式，而流媒体系统采用“边下载边运行”（Play-While-Downloading）。具体来说，在一个 P2P 流媒体系统中，一个对等节点的子集拥有一个特定的媒体文件（或文件的一部分），并未对此文件感兴趣的其他节点提供媒体数据。与此同时，请求数据的节点在下载媒体数据的过程中回方并存储这个媒体的数据，并成为可以为其他节点提供流媒体数据上载的节点。因此，与传统的 P2P 文件共享系统相比，P2P 实时流媒体系统需要为流媒体数据的实时传输提供要求更为严格的资源管理和控制功能。

要设计实现一个高效的 P2P 实时流媒体系统，一个挑战是由于广泛存在的节点上载带宽和下载带宽的不对称性[9]（如采用 ADSL、cable modems 或 V.90 protocol 接入的节点），提供流媒体数据服务的节点所提供的上载带宽可能小于媒体数据的回放速率。解决这个问题的方法是，是用多个数据发送节点为接受节点的每个实时流媒体会话提供服务。之前的一些相关工作[10][11]为每个接受节点只安排一个数据发送节点。

另一个挑战是：在一个高度动态异构的对等网络中，如何为每个流媒

体服务会话选择和监控适当的数据发送节点，从而得到满足应用要求的流媒体服务质量。上面提到的动态性和异构性来自于节点能力和节点间的网络连接[9]：

1. 对等网系统中的自主节点可能随时加入或退出系统；
2. 流媒体数据发送节点随时可能加入或停止流媒体服务会话；
3. 节点的网络连接可能随时断开；
4. 流媒体数据发送节点的上载带宽可能随时改变；
5. 与多个数据发送节点间的连接具有不同的端到端带宽、丢包率和失效率；
6. 就丢包率和失效率而言，底层网络拓扑结构决定了数据接受节点和多个数据发送节点间的连接存在相关性。

基于以上分析，本文认为一个理想的基于对等网技术的实时流媒体系统应具有以下几方面特性：

1. 自动扩容能力。通过让每个使用服务的节点同时成为提供服务的节点，整个系统的服务能力随着加入节点的增多而自动放大，从而从根本上解决传统服务器/客户机模型的可扩展性问题。
2. 无中心服务器。对于系统中的节点来说，地位平等，大量的并发访问涌向一小部分节点，从而使服务负载得到平衡；除此之外，避免系统中出现单点故障部件。
3. 媒体内容的高效查找。一个理想的系统应该能够并发的支持多个媒体内容的服务，这样就引发一个问题，如何在一个无中心的由大量自主动态节点组成的系统中高效查找媒体内容？
4. 处理异构性。实测工作[9]发现，节点能力的异构性，尤其是节点带宽的异构性在对等网系统中广泛的存在。这种异构性可能来源于不同节点不同的网络接入条件，也可能来源于节点为特定的 P2P 系统提供的不同的应用带宽。
5. 处理动态性。能够动态的根据节点状况和网络状况，为每个流媒体会话选择合适的一组数据发送节点，并实时监控数据传输情况，动态的进行优化调整和出错处理。

1.4 研究内容

本文研究如何基于P2P网络在动态多样的底层网络基础上构建可以可扩展、高效率、高可用的流媒体服务，解决现有基于服务器/客户机模型的互联网流媒体服务存在的性能问题，从而为P2P技术开辟更广阔的应用领域。

本文研究内容如下：

首先，本文研究支持流媒体应用的底层 P2P 覆盖网协议，及与之相关的数据信息搜索算法。从基础平台来看，P2P分为结构化P2P和非结构化P2P，这两种系统在数据存放、搜索条件、路由和一致性维护方面差别非常大，但具有很强的互补性。什么样的 P2P 底层协议能够更好的支持上层流媒体应用，这个重要的研究课题并没有得到研究界应有的重视。结合流媒体应用的特点和需求，本文提出了一种P2P底层网络平台，非结构化P2P协议CAPU，并讨论了CAPU协议在支撑流媒体上层应用时的在节点成员维护（Membership Maintenance），数据存放（Data Placement）、路由策略（Routing Protocol）和搜索算法（Search Algorithm）等核心算法和相应的性能评价（Performance Evaluation）。

其次，本文研究在动态的环境下，如何利用底层P2P覆盖网（Overlay）的拓扑结构和性能信息来合理选择数据发送节点，并利用底层覆盖网协议的相关功能调用监控邻居节点及与邻居节点间网络连接的状态，并根据状态的变化来动态调度并调整邻居节点的选择。从而合理利用底层覆盖网协议的功能，在一个节点和网络状况不可靠的环境中构造高可用性的流媒体服务。

最后，由于本文采用多个数据发送方提供流媒体数据，多个数据发送方无法保证媒体数据按照播放时的顺序依次到达，因此，播放延迟成为影响P2P流媒体可用性的重要因素。为了解决这个问题，本文设计了一个优化的数据分派调度算法，解决给定一个媒体数据接收节点和一组媒体上载带宽各异的数据发送节点，如何为每个数据发送节点分派媒体数据子集，从而保证在多个数据发送节点间合理调度数据的发送，尽量减少媒体数据的传输延迟。

1.5 各章内容简介

本文共分为六章：

第一章（即本章）在对互联网上流媒体系统的发展过程、应用、系统特点、主要研究问题等简单介绍之后，阐述了基于对等网技术构建流媒体系统中实现可扩展、高效率、高可用的流媒体服务的必要性和面临的挑战。最后介绍了本文的主要研究内容和贡献。

第二章综述了与 P2P 和 P2P 流媒体系统相关的研究结果，分为三部分：首先介绍 P2P 基础设施，及在 P2P 研究领域内的一些核心问题和关键技术。其次介绍了传统流媒体系统中可供借鉴的技术，主要是传输质量控制、穿越防火墙、传输协议。最后分类叙述并讨论了现有 P2P 流媒体的研究结果。

第三章研究作为流媒体应用底层平台的 P2P 覆盖网协议。本章介绍了一个非结构化系统，CAPU，这个系统解决了如何利用节点异构性来实现拓扑结构维护，使得 P2P 系统的结点可以根据各自能力以较低的网络开销信息收集量，并实现了高效的信息路由查找策略。。

第四章研究描述了本文提出的 P2P 流媒体系统的总体设计和核心算法，包括邻居节点动态选择算法和获得最小播放延迟的数据分派算法。

第五章描述系统架构并给出性能评价结果。包括本文的测试平台设计，刘媒体系统的体系结构、实现概要以及功能和性能的测试结果。

最后一章对论文内容进行了总结，提出了进一步的研究方向。

1.6 论文贡献

本文的主要贡献在于：

1. 本文提出了一个利用互联网异构性的半结构化 P2P 覆盖网协议 CAPU，CAPU 系统能够合理利用 P2P 系统中节点的异构性和底层网络的状况自适应自调整拓扑结构，并能提供高效率的信息搜索服务，相对传统的非结构化 P2P 搜索算法，带宽占用降低了 3-4 个数量级。
2. 本文提出了一个搭建流媒体系统的邻居节点动态选择算法和数据分派算法，能够动态环境中维护和选择出满足流媒体播放质量要求的邻居节点，并在多个数据发送节点间优化数据发送的选择，来获得最小的数据传输延迟。

第二章　研究背景和相关工作

本章综述与 P2P 和 P2P 流媒体系统相关的研究结果，分为两部分：首先介绍 P2P 基础设施，及在 P2P 研究领域与构建流媒体系统相关的一些核心问题和关键技术。然后分类叙述并讨论了现有 P2P 流媒体的研究结果。

2.1　对等网络基础设施（P2P Overlay Infrastructure）

P2P 基础设施是 P2P 结点得以相互协作的基础，一般指结点互联的拓扑结构和结点在与相邻结点保持连接时的行为规范。P2P 基础设施保证结点形成连通的图结构，并在其上建立了特定的结点逻辑组织。所谓路由（搜索）算法是指从一个结点出发，沿着结点之间的连接进行消息转发，最终到达目标结点或实现路由目标（如搜索到所需数据）的过程。基础设施与路由算法一般是一一对应的，特定的基础设施决定了其上的路由特性和搜索性能。

对于 P2P 流媒体系统而言，P2P 基础设施决定了节点间互联的基本规则，进而决定了搜索媒体数据和数据发送节点的方式和性能，因此，对于 P2P 流媒体系统有着至关重要的作用。本文下面的部分综述了 P2P 基础设施的研究成果。

2.1.1　非结构化 P2P 系统（Unstructured P2P）

P2P 系统中，每个结点保持了一些到其他结点的连接，与这些结点形成了关联关系，而整个系统靠结点之间关联而构成一定的拓扑图结构。非结构化 P2P 中，结点关联而成的拓扑结构具有很大的随意性，不具备特定的结构，因此整个拓扑结构比较松散。由于拓扑结构接近无规则的随机图，非结构化 P2P 中路由是不收敛的，从一个结点出发的路由消息不一定能够成功抵达目的地。非结构化 P2P 的拓扑随意性使系统具有较低的维护开销并且适用于高度动态的环境；其代价是路由的效率很低，用于搜索时无法保证找到所需数据。

非结构化 P2P 基础设施广泛用于 Internet 上交换共享文件。结点自主地加入 P2P 系统，向系统中其他结点共享一些自己的文件或存储空间，并使用路由或搜

索算法寻找所需要的文件。本节介绍 Gnutella[2]和 KaZaA[5]这两个应用最广的非结构化 P2P 系统，其他非结构化 P2P 系统可参考[4] [6]。

Gnutella：在 Napster 系统之后，Gnutella 是第一个真正采用无中心结构的 P2P 文件共享系统。Gnutella 中每个结点维护了一个邻居表，记录了与之相关联的结点的 IP 地址。Gnutella 主要支持三类操作：拓扑维护、文件搜索和文件下载。

拓扑维护通过在相邻结点之间彼此交换邻居结点信息来保持拓扑图的连通性，并替换因结点离线而实效的连接。结点定期向邻居结点发送 PING 消息，收到 PING 消息的结点则回应一个 PONG 消息并附带了当前所拥有的邻居信息。收到邻居列表后结点按照一定规则进行邻居替换，保证自身具有一定数量的有效邻居。当新的结点加入系统时（它需要起码知道系统中另一个结点的 IP 地址），它向系统已有的结点发送 PING 消息而获得足够的邻居结点，从而加入系统。

每个结点共享一些文件，并提供基于文件名的本地查询操作。Gnutella 使用消息洪泛（flooding）的方式搜索其他结点上的文件。发起搜索操作的结点向所有邻居结点发送 QUERY 消息，而接到 QUERY 消息的结点进行本地查询，并把查询进一步转发给自己的所有邻居。这一消息广播的过程重复进行，直到满足一定的结束条件。为避免无穷递归，每个搜索消息都有一个 TTL（time-to-live）域，它随着转发的进行而递减，TTL 为 0 的消息则不再被转发。另外结点对近期接收到的消息进行缓存，以避免重复处理同样的消息。搜索操作结束后，发起搜索的结点会收到一些查询结果，记录了满足条件的文件及其存放的结点 IP。结点可从中选择一些结点来下载所需文件。

由于路由的不收敛性，Gnutella 式的搜索又被称作盲目搜索（blind search）或随机搜索（random search）。这种搜索一方面无法保证在所需文件存在时必然搜索到，另一方面则产生了大量无价值的转发消息，造成了严重的网络负担。如何解决随机搜索的低效率和高带宽开销的问题一直是 P2P 研究工作的热点。

KaZaA：到本文写作的时间为止，KaZaA 系统每天的在线用户数已经超过 3,000,000，并有超过 5,000TB 的共享文件。2002 年在华盛顿大学校园网上的实验显示，KaZaA 应用消耗的网络流量占 TCP 总流量的 37%，是在相同网络上相同时间内 WWW 流量的两倍。文献 [Gummadi03]。另一项统计表明 [Sandvine03]，美国 76%的 P2P 文件共享流量由 KaZaA 程序消耗，而其中仅仅

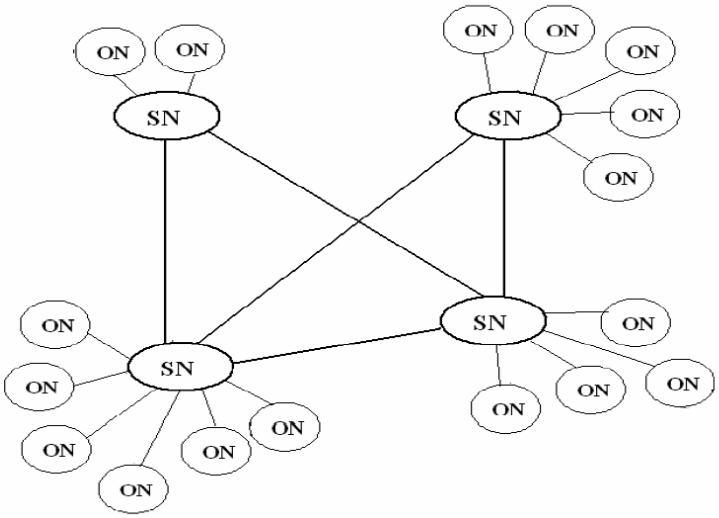

图 2.1 KaZaA 的架构示意图
SN：super node ON：original node

有 8%属于 Gnutella 应用。总之，在用户数量和带宽消耗方面，KaZaA 都已经成为 Internet 网络上最为重要的应用程序之一。

KaZaA 与 Gnutella 在文件共享方式方面非常相似，都没有使用集中式的服务器来提供文件监控和定位服务。而与 Gnutella 不同，KaZaA 把加入系统的结点按照其能力分为两类，强结点(Super Node)和普通结点(Ordinary Node)。如图 4 所示，每个普通结点都隶属于一个强结点。当一个普通结点启动 KaZaA 程序时，它首先与某一强结点建立 TCP 连接，然后向这个强结点传送它所共享文件的原数据。这样就允许强结点维护了一个关于所有隶属于他的普通结点文件的数据库，包括了其从属的普通结点的文件标示和对应的 IP 地址。于是，每个强结点就成为了一个局部的类似于 Napster 中心文件目录服务期的中心结点。这样，KaZaA 较好的利用了大规模 P2P 系统中的结点异构性，按照一个两级的分层结构，让连接能力（带宽），计算能力（CPU）和在线时间长的结点处于较高的层次，在系统维护和文件搜索中承担更多的任务。每个强结点也通过 TCP 连接一定数量的其他强结点。存在连接的两个结点间周期性的交换强结点列表。KaZaA 的典型设置中，每个普通结点维护一个包括 200 个强结点的列表，而每个强结点维护包括上千强结点的列表。在这个列表中，每个结点对应的表项都包括一个时间戳，当结点受到新的结点列表时，会把时间戳较为靠前的结点替换掉。

普通结点向其归属的强结点汇报的文件元数据包括：文件名，文件大小，内容杂凑值（Hash value），以及其他附属信息（这些附属信息可以用于关键字查询）。文件内容杂凑值是一个文件的唯一标示，如果对于某文件的下载任务失败，KaZaA 客户端会根据此内容杂凑值自动搜索相同的文件。

KaZaA 的文件查询过程可以分为三步。

搜索符合用户指定关键字的文件元数据。当用户希望找到某个文件的时候，用户所在的普通结点将会把用户指定的关键字发送给其对应的强结点。

强结点将在本地目录数据库中查找相关文件，若成功找到，则返回文件的原数据，包括文件所在结点的 IP 地址。若不能找到，则向与之存在 TCP 连接的其他强结点广播这个查询消息。收到这个消息的强结点做本地数据库查询，如果能找到，则向提交请求的普通结点返回查询结果，否则，继续广播查询消息。

用户结点从接收到的返回结果中选择最合适的表项进行下载。

2.1.2 结构化 P2P 系统（Structured P2P）

本节将介绍一些前文提到的典型的覆盖网路由协议。这些协议都是根据接收到的标识（一般是一定长度的数字串），把信息路由到相应的结点。每个结点也具有一个标识符(ID)，而且，这个标示符通常是和他对应的文件的标识相同（当一个结点对应一个区间内的文件标识时，结点标识符一定位于这个区间之内，一般是区间的中点）。同时，每一个结点都维护一张路由表，记录其他一些结点的信息。当一个结点收到一个查询操作时，如果它发现所查询的标识不在自己关联的区间内，那么该结点将会把该查询发送给其路由表中它认为最靠近目标的邻居。

Plaxton et al.: Plaxton et al.[36]是第一个能够被 DHT 系统大规模使用的路由算法，虽然该算法的初衷是针对静态网络设计，而非 P2P 系统。该算法采用的是“前串匹配”的路由模式，每一步路由，都会把信息传送到前串更加匹配目标的结点去。例如 ID 为 47532 的结点收到目标 ID 为 47190 的查询请求，它与目标 ID 前串有两位是匹配的，那么它将会把该请求发送到与目标 ID 前串有 3 位匹配的结点去（比如 47103）。这样，覆盖网中的所有结点都要根据自己的 ID 记录一张路由表。具体记法是如果 ID a 具有前串 b 而且 b 的下一位是 c 那么，该结点针对前串 b 需要根据 b 的每一种可能的下一位记录一个邻居结点。例如，

Node ID 10233102

Leaf set

10233021	10233033	10233120	10233122

Suffix set

-0-2212102	**1**	-2-2303203	-3-1203203
0	1-1-301233	1-2-230203	1-2-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

图 2.2 Pastry 系统中结点的路有表（采用的是 4 进制）。Leaf set 记录着在标识空间中与结点距离最近的若干（图中为 4）结点，Suffix set 的第一行，记录的是第一位与该结点 nodeId 不同的点，第二行记录着 nodeId 第一位与该结点相同，而第二位不同的点，以此类推。空格代表系统中不存在符合要求的点。

ID 采用 4 进制，ID 为 1234 的结点针对其前串 12 需要记录 121？、122？和 124？各一个作为自己的邻居。这个算法的另一个优点是，当 O(n2)的结点间距离已知时，结点可以通过选择路由表记录的邻居做到对路由路径的优化。

Pastry: 在 Pastry[15] 中，每一个结点都被分配了一个 128 位的结点标识（nodeId）。这个标识符被用于确定该结点在环状标识空间中的位置。每一个结点的 nodeId 是在结点加入系统是随机分配的，这里假设 ID 的分配是均匀的（实际上，现在的哈希算法能够保证这一点），因为这种随机性，所以在标识空间中位置相邻（nodeId 接近）的任意两个结点在权限，所处的网络拓扑，所有权和网络配属等等方面都可能有很大差异，换句话说，在标识空间相邻的结点，在实际网络中往往差异很大。

在 Pastry 中，结点所记录的路由表如图 1 所示，当系统规模为 n 时，只要不出现标识空间中连续的 L/2（L 是一个系统配置参数，通常取 16 或 32）个结点同时失效，那么，系统一定可以在 O(log n)步之内，把消息路由到 nodeId 与目标标识最近的结点。为了进行路由 nodeId 和目标标识被想象为以 2b 进制的数字串，每一次路由，结点都是在路由表中找到 nodeId 距离目标标识最近的点，具体做法是，如果本结点与目标标识前串匹配长度为 b 位，那么，系统将在该结点路由表中找到 nodeId 至少与目标匹配 b+1 位的结点，如果没有找到这样的点，

那么就去寻找同样匹配 b 位，而且距离目标比本结点近（nodeId 更接近）的点，如果仍未找到，则本结点为目标终点。

Chord: Chord[16]也是一个使用环状标识空间的系统。同样，针对一个标识的路由的目标就是具有在数值上最接近该标识的 nodeId 的结点，并称为针对该标识的承接点。在 Chord 中，每一个结点都维护着两套邻居，其中一套在标识空间中紧接着该结点的 k 个结点，另一套是指向在整个标识环中以该结点为基准依次折半的点的指针。其中，第一套邻居是确保路由正确的关键。Chord 可以确保在路由在标识空间中单向靠近目标接点而且不会越过，并且可以保证路由在 O(log n)步内完成。

CAN: CAN[35]采用的标识是从一个多（假设为 d）维环状空间中选择的。每一个结点都与其对应的标识周围的一个立方区域相关联，同时，它的邻居就是拥有和它相邻的立方区域的结点。路由时，消息一直是从一个结点传输到它的拥有与目标标识最接近的 nodeId 的邻居。与前面几种算法不同，CAN 中每个结点维护大小为 d 的路由表，同时，每一次路由都在 O(dn1/d)步内完成，特别的当 d=O(log n)时，CAN 的路由表和路由长度将于其它系统相似。

2.2 P2P流媒体技术综述

在以下的小节中，我们将从流媒体组播结构构造、组成员管理以及基于覆盖网和测量的组播调整等三个方面介绍目前在 P2P 流媒体系统领域内的研究成果。最后介绍了一个实用的系统，由 CMU 大学开发的 Narada 系统。

2.2.1 组播结构的构建

为了实现覆盖网络的组播，主要的问题是构造组播的结构，即数据流动的路径。考虑一个简单的情形（大多数群组通信可以简化为这样一个模型）：只涉及一个源结点和许多目的结点，我们的目的是尽快地把数据从源结点散发到所有的目的结点。为了解决目的结点数目巨大的问题，一些目的结点充当数据传送路径上的接力结点，把自己收到的数据转发给其它目的结点。因此，许多应用把组播树作为从一到多应用的基本结构。给定覆盖网络的带权连接图 G=(V, E)－－－其中 V 代表所有的参与结点，E 表

示覆盖网络的连接，理论上，这个问题转变成构造一棵以源结点为根的最小生成树（MST）的问题。尽管有构造最小生成树的有效算法，在实际上构造这样的一棵树却不简单：在互联网上取得完整的覆盖网络的连接图不容易；同时，构造出的最小生成树常常包含高出度的结点，这些结点很容易过载，成为系统的瓶颈。因此，在实践中许多组播算法不寻求构造最优的生成树，而是根据部分连接图构造出度受限的近似最优生成树。组播结构主要考虑的问题：1)减少延迟和提高带宽；2)容错；3)构造的难度，特别是用分布的方式。

Overcast[37]试图解决利用互联网发送带宽密集型应用－－－内容提供商面临的一个难题。Overcast 由一个中心的源结点、散布于网络的许多 Overcast 内部结点（具有持续存储的标准的个人电脑）、以及许多具有标准 HTTP 连接的客户机组成。Overcast 利用一个简单的树构建协议，把内部结点组织成一棵以源结点为根的散发树。Overcast 的树构建算法的目标是使每个结点到根结点的带宽最大化。这通过把一个新的结点放置在一个离根结点最远，但又不牺牲到根结点带宽来获得。一个结点一旦初始化，它开始一个从根开始的与其它结点一起的自我组织的过程。在每一轮中，这个新结点考虑它直接到当前结点的带宽和通过当前结点的孩子结点到当前结点的间接带宽。如果任一间接带宽大于或等于直接带宽，与这一间接带宽对应的孩子结点成为当前结点，新的一轮开始。通过这个方法，这个新的结点可以离根结点尽可能的远，同时保证可获得的带宽。因此，树构建算法倾向于把一个新的结点挂在位于同一个网络内的父结点身上。这样，Overcast 通过组播树利用了结点的转发能力，使得带宽最大化。

Bayeux[38]利用了它从应用层路由协议，例如 Tapestry，继承的前缀匹配的路由算法。这样，它结合了负载平衡、可扩展到任意规模的接收者的可扩展性、近邻特性、以及网络容错等。Bayeux 追随了伴随了基于 DHT 的路由带来的“天然的组播树”。既然我们可以很容易地从大多数基于 DHT 的算法中派生出一个树结构，那么利用这样一棵树来进行覆盖网络的组播，而把故障恢复和近邻问题留给覆盖网络的路由层就是很有好处的。

然而，基于树的组播协议存在一些固有的缺点，并且有时难以充分利用协作环境中的可使用的资源。造成这个问题的原因是：在任何的组播树中，承担复制和转发组播数据的结点只是组播树中的内部结点，这些内部

结点只占所有参与结点的一小部分。大部分结点是叶子结点，没有贡献任何资源。这个结果和希望所有结点都承担转发负载的初衷相违背。这个问题在高带宽的应用中进一步恶化，例如视频或大文件发送：在这些应用中，许多接受者可能没有担当传统组播树中的内部结点的能力。另外在树的结构中，从上到下，带宽一般是单调递减的。较高层的损失都会减少低层能够取得的带宽。尽管许多技术被提出来去恢复丢失的数据，由此提高可取得得带宽(例如[29])，然而，一个结点获得的带宽却是只由它在树中的唯一的一个父亲决定的。

为了克服树的天然的缺陷，进来的研究提出使用“垂直链路”来增大通过树获得的带宽(Bullet[39])，或者使用多棵树来取代单一的树(SplitStrem[40])。基本上，这些技术具有一个共同点：我们应当利用覆盖网络中的每一个结点来增强数据的传输，而不是仅仅利用单棵树中的内部结点。

SplitStream 的关键思想是把内容分成 k 个带 (stripe)，并且每个带利用一棵独立的树来广播。结点根据它们愿意接收的带的个数加入同样多的组播树中，每个结点同时还给定一个它们愿意转发的带的上限。问题转变为构造一个组播树的森林：在这个森林中，一个在某棵树中是内部结点的结点在加入的其它的树中都是叶子结点，并且满足结点的出度限制（即转发的带的个数）。这确保转发负载散布到所有参与的结点。例如，如果所有的结点都希望接收 k 个带和转发 k 带，SplitStream 将会构造一个转发负载均衡的森林，这个森林还是低延迟和低链路负载的。图 3 描绘了 Splitstream 中的带和组播森林。

设计 Splitstream 面临的关键的挑战是以一种分布式的、可扩展的、高效的、并且自组织的方式构造一个由内部结点正交的组播树构成的森林。一组树是内部结点正交的，意思是说一个结点最多在一棵树中担当内部结点，而在其它树中是为叶子结点。Splitstream 利用 Pastry 的路由和 Scribe[41]的组成员管理来辅助内部结点正交树集的构建。Pastry 通常把一个消息转发给结点标识和消息标识共享最长前缀的结点。由于一棵 Scribe 树是通过从组成员到组标识的路由来形成的，因此所有的内部成员与组标识共享某些前缀。由此，只要组播树的组标识的最显著的位不同，就可以保证 k 棵 Scribe 树有正交的内部结点集。图 1 演示构造。一个 Stripe 组的组标识也

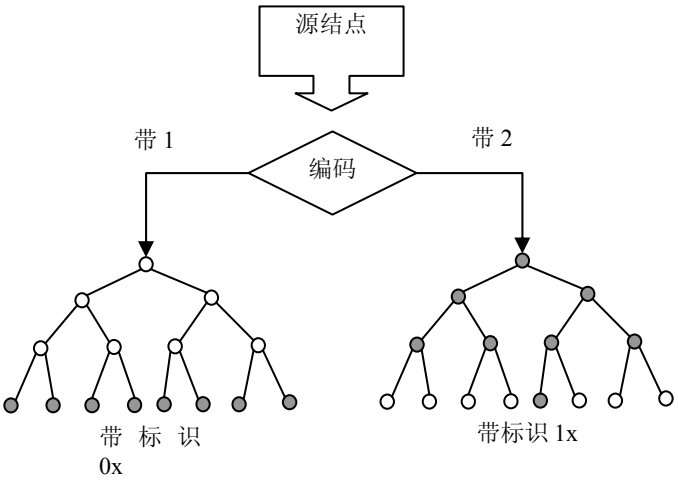

图 4.3 带和 Splitstream 的组播森林

称为这个带的带标识。为了限制结点的出度，还应用了 Scribe 的内建的”push-down”方法。当一个已达到最大出度的结点收到另外一个结点要求接纳为前者的孩子结点的时候，前者把当前孩子结点的列表给后者；后者接着寻求一个有最小延迟的孩子成为它的父亲结点。这个过程在树上从上到下递归进行，直到遇到一个可以接纳后者成为它的孩子的结点为止。而且，为了充分利用一些强结点的空闲带宽，这些强结点被组织成 Scribe 的一个独立的组，称为能力空闲组。所有的 Splitstream 的结点，只要它的孩子的数目少于它的能转发的带的数目都属于这个组。当一个转发请求不能提供时，Splitstream 将在能力空闲群中找一个能满足要求的结点。

Bullet 面向高带宽的组播数据发送，例如把巨量的内容通过互联网发送到多个接受者。不把相同的数据流发送给所有的结点，Bullet 提出在组播覆盖网络中的结点协作传送正交的数据集。结点仍然从父结点那里接收到一组数据，但他们还必须寻找能给它们提供它们缺的数据的结点。Bullet 有个分布式的算法使得数据可以均匀地散布在所有参与的结点当中。使用这个方法，它避免了定位最后一个数据块的问题，取得持续的高带宽的数据发送。

2.2.2 组成员管理 (Scribe)

除了组播的结构，还需要管理参与的结点，特别是把结点根据它们的特性组织成不同的组，和把消息发送到一些特定组的所有成员。组成员管

理是覆盖网络的一个基本服务，也常常是许多组播程序的一个必要的组成部分。

Scribe 提出了一个面向大规模组的成员管理的分布式的应用层组播结构。Scribe 建立在 Pastry 之上，利用了 Pastry 的网络近邻、容错、分布式等显著的特点来同时支持巨量的组。任何 Scribe 结点都可以创建一个组；其它结点可以加入这个组，或者把消息广播给这个组的成员（假定它们有恰当的证书）。每个 Scribe 的组都有一个唯一的组标识；结点标识和这个组标识最接近的结点充当这个组的会合点(rendezvous)；这个会合点是为这个组创建的组播树的根。如果一个 Scribe 的结点希望加入一个组，例如 g，它要求 Pastry 路由一个以 g 的组标识作为消息标识的消息 JOIN。这个消息向 g 的会合点路由过去。路由路径上的每个结点查看它自己的组列表，看自己是否是 g 的转发器。如果是，它把这个结点接纳为孩子结点，把它加入到自己的孩子表中。否则，它为 g 创建一个条目，把这个结点加入自己的孩子表中，还通过把一个 JOIN 消息发送给这条从源结点到会合点的路由路径的下一个结点而成为 g 的一个转发器。因此，组管理机制对于组规模大小不同的组来说都是高效的。Pastry 的随机的特性确保树是平衡的，并且转发的负载均匀散布在所有的结点中。这种平衡使得 Scribe 可以支持许多有非常多成员组。更进一步来说，加入请求以一种分布的方式当地给吸纳了。特别是，会合点没有处理所有的加入请求。因此，Scribe 可以作为基于覆盖网络的组播的基本组成部件，一个例子就是前面介绍的 Splitstream。

CoopNet[42]面向现场直播的流媒体的应用。这些应用把一个服务器的内容发送给许多可能高度动态的接收结点。CoopNet 使用客户端的组播来减轻流媒体服务器的负载，并且帮助服务器克服瞬间冲击的问题。CoopNet 没有使用纯粹的 P2P 方式。因此，CoopNet 受益于一个比分布方式更高效的中心的管理。CoopNet 有一个指定的工作站负责管理结点的加入和离开。工作站把组播树的整个结构存储在内存中。当一个结点开始接收现场直播的流媒体时，这个结点与工作站接洽加入的操作。工作站从保存在内存的组播树中找到一个合适的位置，把这个结点的父结点返回给这个结点。这个方式是非常高效的：它只需要一轮消息。尽管集中管理不是自我可扩展的，研究表明 CoopNet 的管理任务是非常“轻”的：一个有 2 GHz Mobile

Pentium 4 处理器的笔记本电脑可以支持每秒 400 个加入和离开消息。因此，集中管理也是可行的，在实际应用中可以扩展到大规模系统中。

2.2.3 基于覆盖网络的测量和组播调整

除了组播的构建和组成员管理外，一个覆盖网络应用程序还必须不停地调整自身来适应多变的互联网环境。众所周知，互联网是一个高度动态的环境，很容易发生不可预测的分割、拥塞、和突发冲击。因此，早些时候构建好的组播结构过了一段时间后可能变得效率低下。所以，覆盖网络应当通过反复的测量感知底层的变化，相适应地调整自身的结构。

现在已经提出了很多基于检测的技术来测量覆盖网络的连接。这些技术使用轻量的消息检测来估计结点间的延迟与带宽。在这些技术中，RTT(round-trip time)和 10KB 的 TCP 检测是常用的方法。另外，一些方法[43]试图去估计瓶颈带宽。尽管检测消息和轻量的估计不能完全勾画连接的特性，但它们在故障检测和路径选择方面是很有用的。通常，覆盖网络把结点间的连接看作“黑盒”，而对黑盒的测量对基于覆盖网络的组播系统来说已经足够了。

检测出变化之后，覆盖网络需要调整自身和组播结构来提高性能。许多覆盖网络的设计采用了组播结构的局部调整：允许结点动态地选择能够提供更好服务的结点。在 Overcast 中，为了获得移动数据时观察到的近似带宽，协议测量下载 10KB 花费的时间。这个方法给出的结果好于使用底层带宽测量，例如 ping，得出的结果。除此，结点通过周期性地测量到当前兄弟结点、父亲结点、和祖父结点的带宽来估计自身的位置。结点直接测量到祖父结点的带宽，作为对先前作出的做当前父亲结点的孩子结点是否正确的一个测试。如果必要的话，结点向上移动一层，成为它原来父亲结点的兄弟结点。由此，Overcast 内在地可以容纳除了根结点外的结点失效和网络拥塞。

2.2.4 Narada：实用的视频转播系统

Narada[11]是一个由美国 CMU 大学研制的实用的组播系统。这个系统采用了一种称为终端系统组播（End System Multicast）的技术。并且在它之上，开发了实用的视频转播系统。使用这个视频转播系统，转播了

SIGCOMM，SOSP 等国际会议。以下介绍 Narada 以及在其之上的视频转播系统。Narada 采用了被称作终端系统组播（End System Multicast）。在这个结构里，终端系统实现了所有的组播相关的功能，包括成员管理（membership management），包复制（packet replication）等。这种把由路由器支持组播，转变到由终端系统支持组播的结构，能够解决 IP 组播的大部分问题。

这种终端组播系统具有以下优点：

1. 可扩展性：相对于 IP 组播，路由器不需要维护每个组播组的状态。由终端系统来维护组播组的状态，但是它们只参与到很少的组当中去，所以具有可扩展性。
2. 容易部署：由于 ISP 不愿意部署 IP 组播，而终端系统组播更容易部署。
3. 能够简化对更高层功能的支持：终端系统组播可以利用终端系统的计算和存储能力，比如说缓存包，改变编码等。所以更加容易对更高层功能进行支持。

Narada 创建组播树有两个步骤：第一，它首先创建一个连通图，称为网（Mesh）。并保证使其具有一些特性。第二步，Narada 在 Mesh 上面创建生成树（spanning tree）。每棵树以对应的源结点为根。

Narada 可以对不同的应用做不同的优化。视频会议应用需要对覆盖树同时进行基于延迟和可用带宽的优化。视频会议应用可以容忍由于丢包而造成应用程序质量的下降。这就允许使用 TFRC[44]作为底层的传输协议。为了使得覆盖树的创建能够同时针对带宽和延迟做优化，使用[45]中的关于多个路由尺度的工作。

第三章　P2P 基础平台设计

P2P 基础设施是 P2P 结点得以相互协作的基础，一般指结点互联的拓扑结构和结点在与相邻结点保持连接时的行为规范。一般将 P2P 的基础设施称为覆盖网（Overlay），覆盖网可以用一个图结构来表示，结点由图中的顶点表示，如果两个节点间有网络连接（包括软状态 Soft State 连接），则在图中对应的顶点间连边，这两个节点互相称为邻居节点。那么覆盖网的设计目标就是，为 P2P 网络中的顶点构造一个连通的图结构，并在其上建立了特定的结点逻辑组织，这个逻辑组织决定了节点间的互连方式。除此以外，覆盖网协议决定路由算法。所谓路由（搜索）算法是指从一个结点出发，沿着结点之间的连接进行消息转发，最终到达目标结点或实现路由目标（如搜索到所需数据）的过程。基础设施与路由算法一般是一一对应的，特定的基础设施决定了其上的路由特性和搜索性能。

通过前面的章节我们已经知道，对于上层流媒体系统而言，需要底层作为基础设施的覆盖网提供一个可以连接的节点列表从中选择邻居节点，并实时更新这个列表来根据数据传输情况和网络状况来动态作出邻居节点的调整和选择。此外，我们所设计的目标流媒体系统中可以并行的运行多个媒体会话，这也就意味着节点在加入某个媒体会话之前需要实现查找到这个媒体会话。因此，底层覆盖网还需要提供高效的信息查找服务。最后，加入同一媒体会话的节点组成一个动态的节点组，因此，底层覆盖网还需要提供动态的组成员管理服务。

综上，我们的覆盖网的设计目标是：在一个动态、异构的网络环境中，为节点提供高效的节点收集、信息查询和动态组管理的服务。在本章中，我们研究什么样的底层覆盖网络能够为上层的流媒体服务提供更好的支持，并提出了基于非结构化 P2P 覆盖网的 CAPU。

3.1 节介绍 CAPU 系统的基本框架和系统设计算法设计以及性能分析，3.2 节介绍节点能力自适应拓扑结构维护算法，3.3 节描述节点信息发布机

制，3.4 节给出两阶段信息搜索算法，3.5 节讨论如何根据底层网络状况优化拓扑结构。

3.1 CAPU：基于非结构化P2P的覆盖网算法

相对于结构化覆盖网而言，非结构化 P2P 覆盖网的节点间形成的拓扑结构具有很大的随意性，接近于无规则的随机图。在这样的随机化的拓扑结构上，路由一般采用洪泛式算法或者随机走步算法，这些算法搜索效率低下。其中洪泛式路由会造成造成不必要的大量的冗余消息，随机走步式路由又无法保证找到所需的数据。但是，随机拓扑结构给了系统中节点选择邻居节点较大的自由度，也就为特定上层应用而优化网络连接提供了较大的设计空间。

结合结构化 P2P 系统的特性和流媒体应用的需求，我们希望设计的非结构化 P2P 覆盖网协议具有如下三方面的特点：

1. 整个系统能支撑大规模节点同时在线提供并接受流媒体服务。
2. 用户能够高效的搜索到感兴趣的媒体文件信息。
3. 系统的拓扑结构能够有效的根据实际网络状况进行调整，从而为用户提供高质量的基于覆盖网的网络连接。

基于以上考虑，我们提出并设计了 CAPU 系统，一个非结构化 P2P 覆盖网协议，这个协议有如下部分组成：

1. 根据节点能力自适应的半结构化拓扑结构维护算法。这个算法保证覆盖网维持一个高吞吐的、负载平衡的、低网络直径并且具有高容错能力的半结构化拓扑结构。
2. 主动信息发布扩散算法。在上述的拓扑结构之上，每个节点被赋予一个与节点能力相关的级别，根据这个级别，每个节点将自身的数据信息及级别比它低的邻居汇报给它的信息汇报给级别比他高的邻居节点，而且这个过程是递归的。
3. 两阶段的信息搜索算法。根据上面的拓扑结构及主动信息发布机制，我们设计了一个第一阶段为随机走步，第二阶段为基于 MPR（Multi-Point Relays）的无冗余广播搜索算法。这个搜索算法结

合了非结构化 P2P 路由中两种基本模式的优点，并较好的利用了 CAPU 的拓扑结构和主动信息发布机制，在非结构化结构中实现了可收敛的高效信息查找机制。

4. 根据底层网路状况自调整的拓扑优化算法。CAPU 系统提供给每个节点大量的其它节点的信息，这个拓扑优化算法使得每个节点能根据从这些节点列表中选择网络状况较好的节点建立连接，从而为上层应用提供较好的基于覆盖网的网络连接。

本节由以下几个部分组成，3.2.1 到 3.2.4 分别介绍协议的四个主要功能模块，分别是拓扑结构维护算法、节点信息发布机制、两阶段信息搜索算法和根据底层网络进行自调整的拓扑优化算法。最后 3.2.5 节给出 CAPU 系统的性能分析。

实验证明，CAPU 系统能够合理利用 P2P 系统中节点的异构性和底层网络的状况自适应自调整拓扑结构，并能提供高效率的信息搜索服务，相对传统的非结构化 P2P 搜索算法，带宽占用降低了 3-4 个数量级。

3.2 节点能力自适应拓扑结构维护算法

对于 CAPU 系统而言，节点能力或者说异构性级别（Capacity or Heterogeneity Level）的概念至关重要。[12][17]等工作中将节点能力的分布视为连续分布，与之不同，我们将不同能力的节点分成有限离散的数个级别。为节点赋予离散的异构性级别使得构建半结构化的拓扑结构成为可能。我们将节点的能力概括为一个包括网络访问带宽、处理器主频、内存容量及外存访问延迟等参数的向量。这些参数都描述与节点与参与 P2P 系统能力高度相关的资源能力。对于我们特定的流媒体应用而言，与服务质量关系最为密切的参数就是节点带宽。因此，在我们的模型中，我们把节点的网络访问带宽作为最关键的参数，并根据这个参数来为节点定级。文献[19]指出，在 Gnutella 系统中的节点带宽差异非常大，带宽的最大值（Cable 联接、高速 LAN 联接）与最小值相差几个数量级。因此，带宽还能很好的描述节点的异构性。

CAPU 的节点能力自适应拓扑调整协议主要处理节点如何选择、连接、断开和维护与邻居节点间的连接。下面的讨论都是把 CAPU 的拓扑结构视为有向图，图上的顶点代表系统中的节点，若节点 A 在另一个节点 B 的邻居列表中，则在图上从对应的顶点 B 向节点 A 引一条有向边。通过这个协议，CAPU 系统能够：

1. 负载平衡。保证每个节点连接度（所连接的邻居节点的个数）与节点能力近似成正比，级别相同的节点彼此间度数平衡。
2. 可达性。任给系统中两个节点都存在有向路径使得两点彼此可达。此外，连接两点的最短路径所经过的节点级别介于两个端点的级别之间。
3. 连通性。任给系统中一个节点，若其级别在整个系统中不是最高的，那么它至少有一个邻居比它的级别高。

通过以上三个目标，系统能够充分利用加入节点的能力，根据节点的能力级别来分配拓扑结构维护和信息查询的负载；同时避免一个能力较弱的节点成为影响整个系统吞吐量的瓶颈，也让一个能力较强的节点维护更多的邻居连接并能获知更多的信息数据和有更多的机会处理查询请求；系统还保证任何一个节点能够在至多 L 步之内到达系统中某个级别最高的节点。因此，从整个系统的角度来看，这个样的半结构化系统可以得到近似最优的系统吞吐率。

我们将一个节点 X 的邻居节点列表，用 $N(X)$ 标记，分为两个组，分别成为父节点列表（ $N_p(X)$ ）和子节点列表（ $N_c(X)$ ）。节点 X 的能力级别计作 $L(X)$。父节点列表包括该节点所有级别大于或等于 X 的邻居节点。相应的，子节点列表 $N_c(X)$ 包括 X 所有比它能力级别低的邻居节点。我们将节点 X 的节点列表的大小定义为 X 的连接度数（ $D(X)$ ），并满足：

$$D(X) = D_p(X) + D_c(X)$$

而对每个节点而言，都有一个度数上限（ $Th(X)$ ），这个上限与节点度数成指数正比。

当一个节点 X 加入系统时，首先通过类似于现有 Gnutella[2]的通过联系一个节点缓存服务器（这种方式被称为“约会”机制）或者通过这个节

符号	意义及说明
$L(I)$	节点能力级别
$Th(I)$	节点邻居列表上限
$B(I)$	启动节点集合
$N(I),D(I)$	邻居节点集合及其大小
$N_p(I),D_p(I)$	父节点集合及其大小
$N_c(I),D_c(I)$	子节点集合及其大小

表 3.1 本章主要符号说明

点本机上先前所存储的节点列表来获取一个初始节点列表。X 从这个列表中选出所有级别高于 $L(X)$ 的节点组成 X 的启动节点集合，计作 $B(X)$。X 向 $B(X)$ 中的节点发送标记为 PING 的请求连接的消息，这个消息附加节点 X 的一些信息，如 X 的级别 $L(X)$ 等。当 $B(X)$ 中的节点接收到这样的 PING 消息之后，首先检查自己现有的度数是否已经达到其度数上限。如果还没有达到，那么接受 X 为邻居节点，并向 X 返回一个 PONG 消息，这个消息也会附带上一些此节点的诸如孩子邻居列表大小和级别之类的信息。

每个节点都倾向于选择级别高的节点作为自己的邻居节点；如果两个备选节点的级别相同，那么选择孩子节点列表较小的那个节点作为邻居。在这个条件限制下，节点 X 将返回 PONG 消息的所有节点进行排序，选出前 $Th(X)$ 个节点，向它们发送 CONNECT 消息，试图建立连接。如果向 X 返回 PONG 消息的节点个数小于 $Th(X)$，那么 X 继续向其他节点或系统的节点缓存服务器索取节点列表，并从这个列表中选择节点发送 PING 消息。

如果节点 Y 从节点 X 接收到了这样一个 CONNECT 消息，N 首先检查它的度数 $D(Y)$ 是否小于 $Th(Y)$。若 $D(Y)<Th(Y)$，那么 Y 将接受 X 为自己的邻居节点并将 X 加入它的邻居节点列表 $N(Y)$。若此时 $D(Y)=Th(Y)$，那么 Y 检查自己邻居列表中是否存在节点的级别小于 X 的级别。如果存在这样的节点，M，则 Y 向 M 发送消息将 M 到 Y 的连接重定向到 X，并接纳 X 作为自己的邻居节点。CAPU 系统中的节点周期性的与邻居节点发送心跳消息（Heartbeat Message）。如果一个节点在几个心跳周期内都没有收到某

个邻居发来的心跳消息，那么将把这个邻居节点的状态视为离开，并将这个邻居节点在邻居列表中的表项删除。下面的章节中，我们将进一步解释如何在这个心跳消息中附加上其它的一些数据，例如文件列表，邻居列表，及连接的权重等。算法 3.1 是这个算法的详细伪码。

算法 3.1: 能力自适应拓扑维护算法

{new node I join}
if I is new node into system **then**
Get $B(I)$ from web cache;
For each node $J \in B(I)$, such that $L(J) \geq L(I)$
Send PING msg to J
end if
{when node I receive PING message for node J }
if $D(I) < Th(I)$ **then**
Send PONG back msg to I ;
else {node I has no free degree }
if $\exists M \in N_c(I)$, such that $L(M) < L(J)$ **then**
Send PONG back msg to I ;
end if
end if
{when node I receive PING message from at least $Th(I)$ nodes}
Denote the set of all the nodes returning PONG msg as S ;
Select the top $Th(I)$ nodes with max level and free degrees;
Send CONNECT request message to all these nodes;
{when node I receive CONNECT message for node J }
if $D(I) < Th(I)$ **then**
Add J into $N_c(I)$, accept the connect request;
else
if
Select node $M \in N_c(I)$, such that $L(M)$ is minimum;
Delete M from $N_c(I)$; Redirect M 's link to J ;
Add J into $N_c(I)$, accept the connect request;
end if
end if

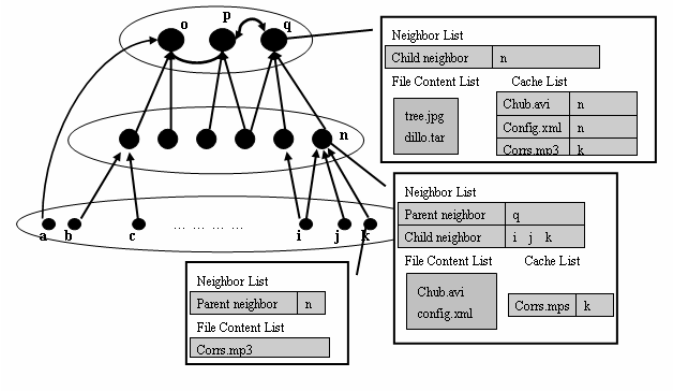

图 3.1 节点信息发布机制

3.3 节点信息发布机制

为了充分利用对等网络中节点的异构型，CAPU 系统中每个节点都要主动的将存储在它上面的文件信息加上上别的节点向它汇报的信息一起汇报给它所有的父节点。文献[12][17]中也提出了相似的方法，但在这两个系统中，每个节点只维护它直接邻居的数据信息。而在我们所设计的 CAPU 中，由于 3.2.1 所介绍的拓扑结构算法保证节点信息从弱节点向强节点逐级汇报，这样系统中的节点所掌握的节点信息将与这个节点的能力相关。在邻居节点间相互发送的心跳消息中，附加这个节点从上一次汇报信息到本次汇报之间这个节点上的文件信息所发生的变化。若与某个邻居节点相关的文件信息在给定的时间间隔之内没有更新，那么就视为陈旧数据，直接删除。这样保证在节点生命期内所存储的数据保持更新。这样，当节点接受到其它节点发来的查询请求的时候，除了检索本机所存储的文件之外还能不必发送消息而查询所有比它级别低的邻居所汇报的文件信息。这样大大减少搜索步数、反应时间并减少消息冗余。同时，由于我们设定文件消息的传播是从弱节点向强节点扩散，所以，使得掌握较多文件信息的节点是处理能力强的节点，使得这种机制在实际系统的实现中成为可能。图 3.1 给出了 CAPU 中的信息发布机制的一个例子。

3.4 两阶段信息搜索算法

CAPU 的两阶段信息搜索算法包括两个步骤：基于概率的偏向随机走步搜索和基于 MPR 的无冗余广播算法。

节点 X 试图搜索某个文件信息或者接受到其它节点发来的搜索请求的时候，首先检查本地是否存有满足这个搜索请求的数据文件。如果本地没有，那么检查它的邻居节点汇报过来的文件信息列表中是否有满足查询条件的文件。如果上面的本地查询能否找到合适的查询结果，那么直接向发起搜索的源节点返回搜索结果。否则，考虑向 X 的邻居节点发送查询消息。如果 $L(X)$ 不是整个系统中的最大值，根据我们前面介绍的拓扑构造算法，节点 X 必然存在级别高于 $L(X)$ 的邻居节点。则 X 从父节点列表 $N_p(X)$ 中选择节点 Y，向 Y 发送（或转发）查询请求。由于前面所介绍的基于节点能力的拓扑算法和主动信息扩散机制保证在邻居节点之间，级别较低的节点所获知的系统中的文件信息的集合是级别较高节点掌握文件信息集合的子集，这样保证查询请求从低级节点向高级节点扩散的过程也是从较小文件信息集合向较大文件信息集合扩散的过程。

为了避免一些节点成为系统中查询消息所访问的热点（hotspot）从而负载过重，我们在节点选择在父节点列表中选择下一跳节点的过程中引入了负载平衡的偏向随机走步机制。这个机制将在下面给出详细解释和理论模型。

如果一个查询请求到达某个顶级节点然而还不能找到满足要求的文件内容时，则查询算法进入第二个阶段。这个查询请求将在 CAPU 系统中的顶级节点间广播。与类似与 Gnutella 中的洪泛式广播策略不同，我们引入了基于 MPR（Multi-Point Relay）的广播策略来减少简单洪泛式路由中大量冗余消息造成的带宽浪费。算法 3.4 给出两阶段搜索算法的详细伪码。下面我们详细介绍两阶段搜索算法中的负载平衡的随机走步算法和基于 MPR 的无冗余广播算法。

3.4.1 负载平衡的随机走步算法

算法3.2: 链接权重更新算法

$w_{i,j}$: the weight of node I associated with parent neighbor J

$w_{j,i}$: the weight of node I associated with child neighbor J

$w_{out} = \sum_{J \in N_p(I)} w_{i,j}$; $w_{in} = \sum_{J \in N_c(I)} w_{j,i}$;

for all $J \in N_p(I)$ {update parent link weights}

$w_{i,j} = \frac{w_{i,j}}{w_{out}}$; send $w_{i,j}$ with *keep alive* to J

end for

for all $J \in N_c(I)$ {update child link weights}

$w_{j,i} = \frac{w_{j,i}}{w_{in}}$; send $w_{i,j}$ with *keep alive* to J

end for

为了避免系统中某个节点成为查询请求的热点而负载过重，我们设计了这个负载平衡的随机走步算法。节点 I 向它的父节点 J（$J \in N_p(I)$）发送查询请求消息的概率为 $w_{i,j}$，$w_{i,j}$ 的定义如下：

$$\sum_{J \in N_p(I)} w_{i,j} = 1 \qquad (1)$$

$$\sum_{I \in N_c(J)} w_{i,j} = 1 \qquad (2)$$

满足约束(1)的非负矩阵 $(w_{i,j})$ 成为随机矩阵，如果此矩阵也满足约束(2)，则称之为双随机矩阵。若一个矩阵不可约且满足双约束的约束条件，那么以这个矩阵为转移矩阵的马尔可夫链有一个唯一稳定状态分布，且此分布为均匀分布。这意味着节点间的查询负载均匀分布。

每个父邻居节点的权重 $w_{i,j}$ 通过节点间交换心跳信息来保持更新。更新的方式满足下式(3)(4)。

$$w_{out}(i) \leftarrow \sum_{J \in N_p(I)} w_{i,j}, \forall J \in N_p(I): w_{i,j} \leftarrow \frac{w_{i,j}}{w_{out}(i)} \qquad (3)$$

$$w_{in}(i) \leftarrow \sum_{j \in N_c(I)} w_{j,i}, \forall J \in N_c(I) : w_{j,i} \leftarrow \frac{w_{j,i}}{w_{in}(i)} \qquad (4)$$

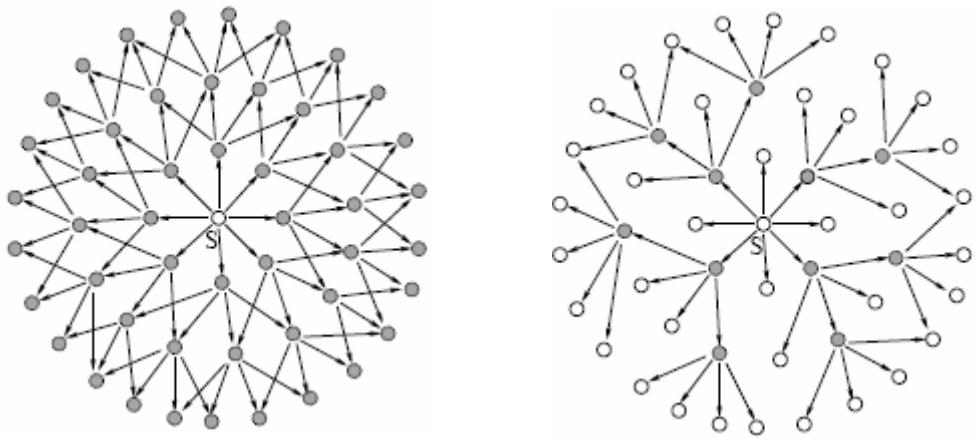

图 3.2 基于 MPR 广播的示意图。左图是 Gnutella 中洪泛式广播的示意图，可以看到，在这种模式下，为了使 S 点发出的消息让图中所有的节点接受到，总共需要发出 49 条消息。而右图所示的基于 MPR 广播中，同样的广播范围只需发送 11 条消息。

节点 I 将通过等式(3)计算更新之后的 $w_{i,j}$ 发送到父节点列表 $N_p(I)$ 中对应的节点 J，且将通过等式(4)更新之后的 $w_{j,i}$ 发送给子节点列表 $N_c(I)$ 中的节点 J。这些更新消息可以附加在前面提到的邻居节点间的心跳消息中发送。当一个新节点刚刚建立邻居节点列表的时候，它将所有邻居节点对应的权重设为均匀分布的平均值。这里描述的权重更新算法文献[20]中所提到迭代算法的一个特例，详细的描述请参考[20]。

基于 MPR 的无冗余广播搜索算法

如果查询消息到达某个顶级节点仍不能找到满足查询请求的文件，那么 CAPU 系统将启动两阶段查询算法的第二个阶段：基于 MPR 的无冗余广播搜索。

为了减少在广播消息转发过程中的冗余消息，我们在广播中引入 Multipoint Relaying（MPR）集合机制。在纯粹的洪泛式广播中，每个节点将受到的消息转发给自己所有的邻居节点，而这些邻居节点将受到的广播消息转发给它们各自的邻居节点。而 MPR 技术选择一个特殊的邻居节点集合继续转发消息，不在这个集合中的邻居节点只简单接受这个消息而不做转发。通过这个机制可以减少系统中可能产生碰撞的消息。我们把 MPR

集合定义为，节点 I 的邻居节点集合中的某个子集，这个子集合中的节点转发消息 I 发出的广播消息，这些消息在一步内能够达到的节点与 I 的所有邻居节点都转发这个消息所能达到的节点相同。满足这个条件的所有子集合中的最小集合称为这个节点的 MPR 集合。MPR 技术在减少网络中消息数的同时以很高的概率保证与消息冗余度很高的简单洪泛式广播算法相同的广播目标。图 3.2 演示了从结点 S 出发的洪泛式广播和 MPR 广播过程的对比示意图。在这个例子中，洪泛式广播需要发送 168 个消息覆盖到的节点集合在 MPR 算法中只需要发送 11 个消息就可以覆盖。

MPR 集合可以分布式来计算，每个节点独立的计算自己的 MPR 集合，而且当节点的邻居节点集合发生变化之后重新计算 MPR 集合。节点计算自己的 MPR 集合需要知道自己所有邻居节点的邻居节点列表。因此，若两个顶级节点互为邻居节点，那么它们需要在彼此的之间的心跳消息中附加自己的邻居列表。在数学上已经证明，计算最小 MPR 集合是 NP-hard 问题，因此我们利用这些信息，通过启发式算法来计算近似最优的 MPR 集合。算法 3.3 是这个启发式算法的伪码。

3.5 根据底层网络状况优化拓扑结构

我们前面已经提到，非结构化覆盖网协议作为流媒体服务底层支持结构的一个优势在于，节点有着较大的自由度来选择合适的邻居节点。本节我们将详细讨论如何通过探测底层网络状况来优化覆盖网拓扑结构，从而为上层大规模应用提供更为有效的连接支持。

对于基于覆盖网的应用，其通讯模式归纳起来有如下五类：

1. 短消息单播。从一个的源节点通过覆盖网发送到一个目标节点的短消息通讯，例如 DHT 系统中的路由通信。
2. 短消息组播。从一个源节点通过覆盖网发往多个目标节点的短消息通信，例如 Gnutella 中的洪泛消息。
3. 长消息单播。从一个的源节点通过覆盖网发送到一个目标节点的长消息通讯。相对于短消息通讯，长消息通讯传输的数据量更大，因此长消息在覆盖网上的传输路径的网络质量对于消息传输质量

算法3.3: 计算MPR

$N_s^2(X)$: the union of sibling neighbor list of all the nodes in X's sibling neighbor list

MPR(X): Multipoint relaying set of X

$MPR(X) \leftarrow \phi$

$S \leftarrow N_s^2(X)$

for all node $M \in N_s(X)$, such that $|N_s(M) \cap N_s^2(X)| = 1$

Put M into $MPR(X)$

$S \leftarrow S - N_s(M)$

end for

while ($S \neq \phi$)

For node $M \in N_s(X) - MPR(X)$,such that $|N_s(M) \cap S|$ is the maximum

Put M in $MPR(X)$

$S \leftarrow S - N_s(M)$

end while

算法 3.4: 两阶段搜索算法

$K(I)$: file content stored in node I ;

$F(I)$: file index propagated to I from its child neighbors

if X is the initial node **then**

Choose node J from $N_p(I)$ proportional to $w_{i,j}$

Send query to J

else

if $K(I)$ has matching file for the query **then**

send the matching file back to initial node; return;

else if $\exists J \in N_p(I)$, such that $L(J) > L(I)$

Choose node J from $N_p(I)$ proportional to $w_{i,j}$

Send query to J

else

MPR multicast the query

end if

end if

end if

4. 更为重要。由于底层的 IP 网络提供了基本的单播传输服务，因此，覆盖网协议应该能从多条可能的 IP 路径中选择服务质量最好的（延迟低，带宽高）的路径来提供给上层通讯请求。
5. 长消息组播。从一个源节点通过覆盖网发往多个目标节点的长消息通信，例如本文所主要研究的 P2P 流媒体服务。对于长消息组播应用，除了需要覆盖网选择 Qos 较好的覆盖网路径，还需要考虑在覆盖网上如何构造合理的组播树（网）来有效的在消息收听节点集合中组播大规模的数据。

通过上面的分析，我们知道，为上层流媒体应用所提供的拓扑结构优化应主要满足长消息的通信应用。下面我们讲讨论如何随机子集流言方式（Random Subset Gossip）获知系统节点信息进行这样的优化。

由于我们讨论的是大规模 P2P 应用，因此系统中的节点不可能获得网络中所有其它节点的信息并从中选择最优的邻居节点。因此，我们引入流言方式来在节点间交换节点信息列表，从而在有限的带宽开销和存储开销上使得节点能有效的掌握网络中节点信息。而根据 CAPU 系统前面的几个功能模块，我们并不需要付出额外的通信开销来实现这个功能，因为，在节点向父节点汇报的文件信息中已经附带了这个文件所在地节点信息，我们只需要合理的利用这些节点信息来实现网络拓扑地优化。

在 CAPU 中，每个节点从其它节点汇报上来的文件信息中抽取出这些文件所在节点的列表，作为自己所获知系统中其它节点的一个采样。在这个随机子集中，节点随机选择一些节点并探测与这些节点间的网络状况参数，如延迟、带宽。对于网络延迟，可以简单的利用 PING 来探测。带宽测量需要更为复杂的机制也需要消耗更多的网络带宽。具体的探测机制，请参考[31][32]。

从上面描述的随机节点子集中随机选择节点来探测网络延迟和带宽，并从中选择具有最小延迟和最大带宽的节点作为邻居节点。

第四章　P2P 流媒体系统设计

本章讨论 P2P 流媒体设计。P2P 流媒体系统的核心模块是对数据发送节点的动态选择和在多个数据发送节点间的数据分派算法。由于 P2P 环境地高度动态性和异构性，为了实现高质量的媒体流，如何选择最优的数据发送节点是一个很重要的问题，也是设计实现流媒体系统最大的挑战之一。在动态环境中维护和选择出满足流媒体播放质量要求的邻居节点的基础上，还要设计合理的算法在多个数据发送节点间优化数据发送的选择，来获得最小的数据传输延迟。

本章的组织结构如下：4.1 节介绍本文讨论的 P2P 流媒体系统的总体结构。4.2 描述系统的核心算法，包括数据发送节点的动态选择算法和最小传输延迟的数据分派算法。最后 4.5 对本章小节。

4.1　P2P流媒体系统总体设计

本文提出的P2P流媒体系统的基本目标是在P2P底层覆盖网提供的拓扑维护、信息查询等通讯服务的基础上，以接受节点驱动（PUSH）的方式，以网状的拓扑结构，动态选择数据发送节点，并在数据发送结点间合理分派数据来实现的高质量的媒体数据传输。表 4.1 总结了流媒体系统设计中的一些关键技术可选方案的比较。

表 4.1 流媒体系统关键技术可选方案的比较

流媒体系统关键技术		优点	缺点
体系结构	服务器/客户机	简单，易于实现，低延迟	不可扩展，不鲁棒
	P2P	可扩展，鲁棒	复杂，难于实现，延迟较高
组播方式	树状	协议简单，延迟低	容错能力差，负载不均衡
	网状	鲁棒，负载均衡	协议复杂，延迟可能较高
驱动方式	发送方（PUSH）	算法简单	不容错，数据冗余，依赖路由协议
	接受方（PULL）	不需路由协议支持,容错，数据冗余度低	算法较复杂

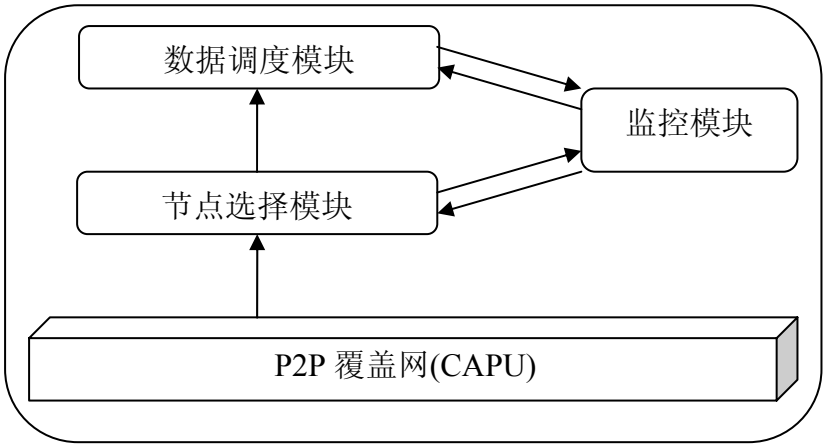

图 4.1 P2P 流媒体系统功能模块示意图

如图 4.1 所示，本文所提出的系统由四个关键功能模块组成：

1. 底层 P2P 覆盖网络模块。为流媒体应用提供基本的拓扑结构构造维护，信息路由查询和网络状况探测等功能。
2. 数据发送节点动态选择模块。根据网络状况和数据传输速率等因素来动态选择出“活动数据发送节点集合”，从这个集合中的节点请求媒体数据来获取最优的数据传输速率。
3. 数据调度模块。在数据发送节点间，根据对方服务能力，数据带宽等因素来分派每个数据发送节点所发送数据的次序，从而获得最小的数据接受延迟。
4. 监控和自调整模块。动态监控每个数据发送节点的数据传输和数据丢失状况，为节点选择模块和数据分派模块提供必要的统计信息。

下面介绍 P2P 流媒体系统的工作流程。当一个用户请求收看某个媒体文件，例如一部电影，首先系统向底层 P2P 覆盖网模块发送查询请求。P2P 模块通过调用信息查询路由算法找到一个拥有这个媒体资源的节点集合，并通过网络探测模块检查出由用户节点到这些节点中间通讯链路的延迟和带宽。然后，节点选择模块根据这些信息选择出“活动数据发送节点集合”。前面已经提到这个子集合是底层覆盖网返回的节点集合中与数据接收节点间网络状况最好的节点所构成的集合，其它节点组成备用节点集

合，当活动节点集合中有节点退出或者网络传输质量下降，则从这个集合中选出一个节点替代掉活动节点集合中质量下降的节点。

下面，系统调用数据调度模块，来根据网络情况来决定给活动数据发送集合中每个节点分派合适的数据发送速率和数据块序号。然后数据接受节点根据这些参数分别与数据发送节点建立链接请求数据。如果活动数据发送节点退出会话或是链路传输质量下降，则监控模块通知数据调度模块重新在活动节点间分派数据传输速率。如果重新分派之后仍不能满足应用的需求，则通知节点选择模块提供新的合适的节点。同时，节点选择模块还会周期性的从底层网络中提供的新节点集合来选出新节点替换现有活动节点集合中数据传输率较差的节点。

下面分别详细介绍系统中几个主要的功能模块的算法。

4.2 核心算法

4.2.1 数据发送节点选择算法

下表定义本文下面需要用到的一些参数。

符号	意义及说明
R	数据接收节点
S	数据发送节点
$B_u(S)$	数据发送节点的上载带宽
$B_p(R)$	数据接收节点的媒体播放带宽
β	冗余系数

决定一个数据发送节点S服务能力的主要参数是与数据接受节点R间的端到端带宽。因此，一个简单的数据发送节点算法就是对底层拓扑返回的节点列表按照端到端带宽进行排序，选出前n个数据发送节点$S_1,S_2.....S_n$使得每个数据发送节点的数据发送带宽之和$\sum_{j=1}^{i} B_u(S_j) \geq \beta B_p(R)$。其中，$\beta \geq 1$。更为精确的节点选择算法还需要考虑的备选结点到数据接受节点之

间端到端路径的相关性。因为，这些端到端路径可能共享某些底层网络路径，从而有同时网络涌塞（Congestion）的可能。本文暂不考虑这些情况。

4.2.2 数据调度算法

本节主要讨论如何在多个数据发送节点间进行数据调度。具体来说，就是给定一个数据请求节点和一个数据发送节点集合，而这些数据发送节点间的上载带宽各不相同，如何在与每个数据发送节点间建立的数据传输通道间调度数据传输速率和数据块次序来获得最小的数据传输延迟。

这个问题的形式化描述如下：给定一个数据接受节点 R 和 n 个可用数据发送节点，数据发送节点 S_i 的发送带宽上限为 $B_u(S_i)$，优化目标是获得使节点 R 可以连续播放的最小数据传输延迟。这个传输延迟定义为节点 R 上媒体流的开始时间与播放时间的差。这个传输延迟的产生是由于存在多个数据发送管道，而在多个管道中数据块的传输顺序与播放顺序不同。因此，对于多个数据发送节点的 P2P 流媒体系统来说，数据传输调度算法对于减少传输延迟来说至关重要。

我们定义函数 $p(t)$ 为播放开始之后时间 t 秒所播放的数据，函数 $d(t)$ 定义为从流媒体开始之后 t 秒之后数据接收节点 R 所接受到的连续数据块。我们假定数据用 CBR（恒定比特率）方式编码，因此，可以用时间来标示数据。下面我们用时刻 k 来表示播放 k 秒的流媒体所需要下载的数据。要保证节点 R 上流媒体连续的播放，必须满足下面这个条件：

$$\forall t \geq 0, d(t) \geq p(t)$$

我们定义节点 R 上的播放延迟为 $\delta = \max_{t \geq 0}\{t - d(t)\}$，为了使流媒体连续播放的条件满足，$p(t)$ 与播放延迟 δ 间的关系须满足

$$p(t) = \min\{\max\{t - \delta\}, L\}, t \geq 0$$

本文下面讨论使 δ 最小化的调度算法设计。这个算法运行在数据接收节点 R 上，根据活动数据发送节点集合中节点信息计算出每个数据发送节点所分配的数据发送序列后，节点 R 分别通知数据发送节点对应的媒体数据序列。数据发送节点根据这些序列号向 R 传输流媒体数据。

首先，我们给出对这个问题的一个模型化描述，我们定义 r 为节点 R 所接收流媒体文件的回放速率，因此，我们假定节点 R 的接受带宽为 r。对

算法 4.1：正排工序法数据调度算法

$i = 0;$

$D = 0;$

For j = 1 to m {

$d[j] = 0;$

$c[j] = class(S_j);$

End for

While ($i \le 2^n$)

For j = 1 to m

If ($d[j] = D$)

Assign segment i to S_j;

$i = i + 1;$

$d[j] = d[j] + 2^{c[j]};$

End if

End for

$D = \min(d[1], d[2], ...d[m]);$

End while

于数据发送节点 S_i，我们假定其能够提供的上载带宽 $B_u(S_i) \le r$。为了便于下面算法的讨论，我们对上载带宽进行离散化，分别为以下值之一：$r/2$，$r/2^2 \ldots\ldots r/2^n$。若某数据发送节点 S_j 的上载带宽为 $r/2^i$，则称此节点的级别为 i，记作 $class(S_j) = i$。假定有 m 个活动数据发送节点，按照上载带宽逆序排列为 S_1，$S_2 \ldots\ldots S_m$，且满足 $\sum_{i=1}^{m} B_u(S_i) = r$。

这样，我们把这个数据调度问题转化为经典的车间调度问题（Job Shop Scheduling Problem）。对于某些问题可能还有其他约束，调度问题就是决定每一作业的操作在指定机器上的可能调度起始时间和结束时间。调度的目标是，对给定作业，要么让它尽可能早完成，要么使它最靠近指定时间完成。调度过程首先是根据作业间的依赖关系决定作业的调度顺序，如管理上的优先级、结束时间先后、松弛时间、作业数量以及考虑装配关系的

算法 4.2: 倒排工序法数据调度算法

$i = 2^n - 1;$

$D = 2^n;$

For j = 1 to m {

$d[j] = 2^n;$

$c[j] = class(S_j);$

End for

While (i $\geq$ 0)

For j = 1 to m

If ($d[j] = D$)

Assign segment i to P_j;

$i = i - 1;$

$d[j] = d[j] - 2^{c[j]};$

End if

End for

$D = \max(d[1], d[2], ...d[m]);$

End while

次序等。调度方法一般存在两种形式，即正排工序法和逆排工序法。正排工序法对每一作业的调度从第 1 道工序开始考虑调度的起始时间，根据操作过程间的相互关系，每一操作被安排到可行加工时间最早的机器上。正排工序法的目的是使作业最早完成。逆排工序法对每一作业的调度从最后一道工序开始考虑调度的结束时间，根据作业结束时间要求及操作过程的约束，每一操作被安排到可行加工时间最迟的机器上。逆排工序法的目的是使作业的完成时间最接近指定的作业结束时间。

算法 4.1 和算法 4.2 分别给出用正排供需法和逆排工序法设计的调度算法。

4.2.3 监控模块

在整个流媒体会话过程中，数据发送节点可能退出会话，或者网络发生拥塞或中断。这些事件的发生会影响数据传输质量，因此我们需要引入监控模块。监控模块运行在数据接收节点上，对每一个数据发送节点实时的收集丢包率和数据传输速率信息。当发现某个数据发送节点的丢包率突然上升或数据传输速率突然下降，则认为这个数据发送节点失效或发生链路断开。这时从备选节点集合中选出一个节点加入活动数据发送节点集合，来代替失效节点。

若监控模块发现某个数据发送节点的数据传输速率发生抖动，则重新启动数据调度模块来对现有活动数据发送集合中的节点重新分配数据块序列。如果发现重新分配序列之后仍不能满足应用需求，那么从备用节点集合中选择节点加入活动数据发送节点集合。

第五章　性能评价

本章通过模拟和实测实验对本文提出的 P2P 流媒体系统作出性能评价。

5.1　试验环境设置

通用 Internet 结构生成和模拟用于在实验室环境中模拟 Internet 和广域网情况，以进行各类 Internet 上的实验和系统测试。它包含两方面的功能，其一是生成符合实际情况的 Internet 网络连接图。这方面的通用工具有 GT-ITM[33]结构生成器，它可以产生基于 Transit-Stub 结构的拓扑连接关系。其二是网络模拟，即根据连接图模拟通信消息在网络中传送时遇到的延迟、带宽限制、拥塞和丢包等问题。很多通用模拟器可以解决此类问题，如 ModelNet[34]和 NS2 等，也可以使用事件驱动的消息模拟器。系统在 Internet 上运行时这一层次被实际网络所替代，在实验室环境模拟时我们使用 GT-ITM 生成网络结构图，并利用事件驱动来模拟消息延迟的情况。

我们设计实现了一个基于事件驱动模型的可扩展对等网络模拟平台。下面简要介绍一下这个模拟平台的设计。

5.1.1　设计思路

设计实现这样一个系统需要解决如下几个各方面的挑战：

第一，如何在节点数有限的机群上模拟百万量级的节点。

第二，如何设计简洁有效的数据结构模拟异构节点的各种状态。前面已经提到，CAPU 适用于高异构性的网络环境，节点的带宽、地域、生命期等各方面的性质各不相同，另外同一个节点在不同的时间可能以不同的级别加入 CAPU 系统。这些节点状态的多样性要求我们设计出灵活高效的数据结构。

第三，如何模拟真实网络的拓扑性质。在这方面已经有一些现成的工具，如 GTITM。但是这些工具的一个共有的缺点是不能生成超大规模的

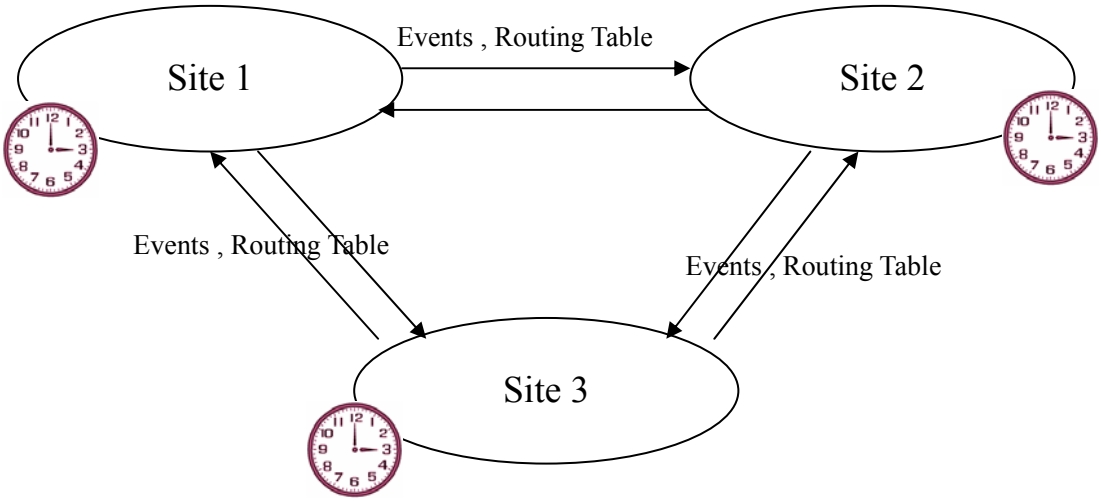

图 5.1 P2P 模拟平台的并行体系结构

网络拓扑结构，因此我们必须根据我们的系统要求，实现一个生成百万量级网络规模拓扑结构的工具。

下面提出的模拟平台的设计结构较好的解决了以上问题，能够在一个中等规模的机群上较为真实的模拟基于大规模网络系统的 CAPU 协议。

5.1.2 体系结构

我们把这个模拟平台的体系结构设计为的并行结构，可在任意多台计算机上以对等模式运行，以三个站点为例，如图 1 所示：

首先把所有的虚拟节点的相关数据（Traffic File，Network Matrix）按照一定规则分布到所有计算节点上，每个计算节点模拟这部分节点的 Tourist 协议，相当于模拟了整个网络的一个子集。节点内部的结构与前面介绍的串行结构完全相同。

但是，每个计算节点上有可能产生改变其他计算节点数据的事件，因此，每个节点还要维护一个系统时钟和一个本地时钟，系统时钟记录所有站点事件队列中的最小时间，本地时钟记录本地事件队列中的最小时间。每个节点的 Event Queue 和 Node Set 的初始化方法与前面一节中介绍的串行方式完全相同。

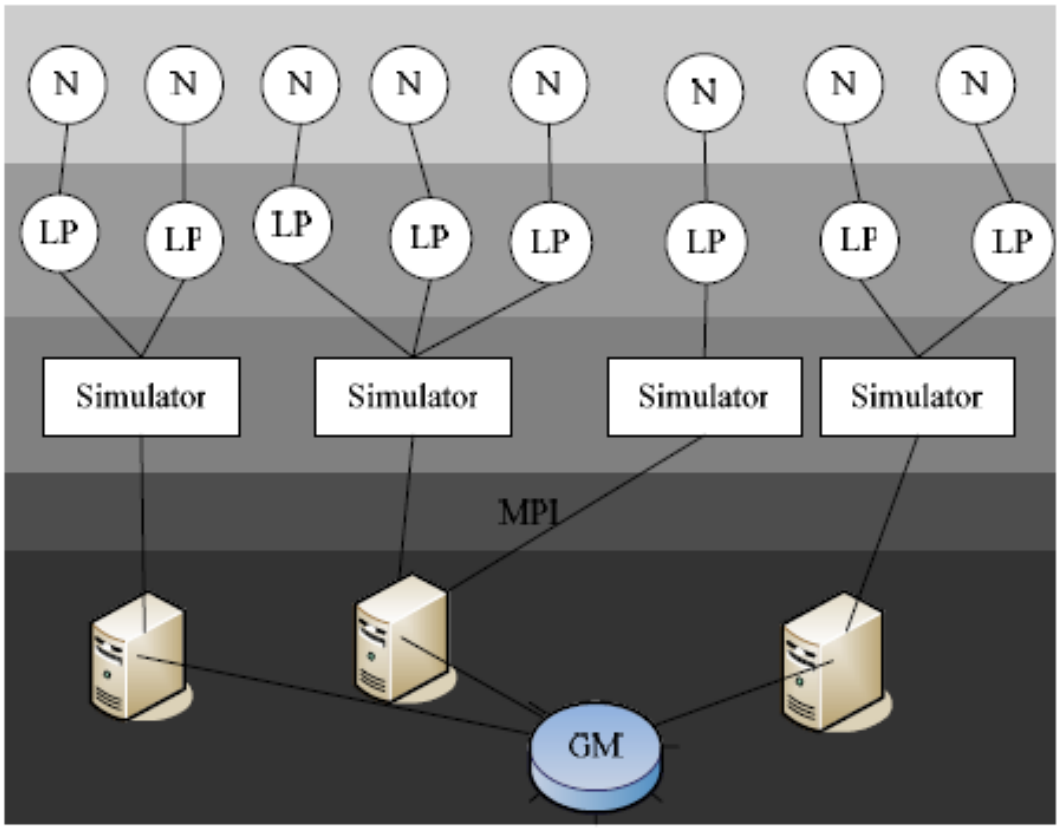

图 5.2 P2P 模拟平台结构示意图

当本地时钟等于系统时钟时，按照串行模式执行取本地事件队列中的事件执行，并生成新的事件，如果新事件只设计本地虚拟节点，则作与串行模式相同的动作；如果涉及其它计算节点上的数据，则放到缓冲区中，等待发送。直到本地时钟大于系统时钟为止，然后把新生成涉及其他计算节点的事件和数据发送给对应的计算节点，并等待其它计算节点发来的事件数据和节点信息数据。计算节点收到别的节点发来的事件和数据后，根据这些事件更新本节点的系统时钟、事件队列（Event Queue）和虚拟节点集合（Node Set）。重复以上过程直到所有的计算节点上的事件队列为空。

5.1.3 接口设计

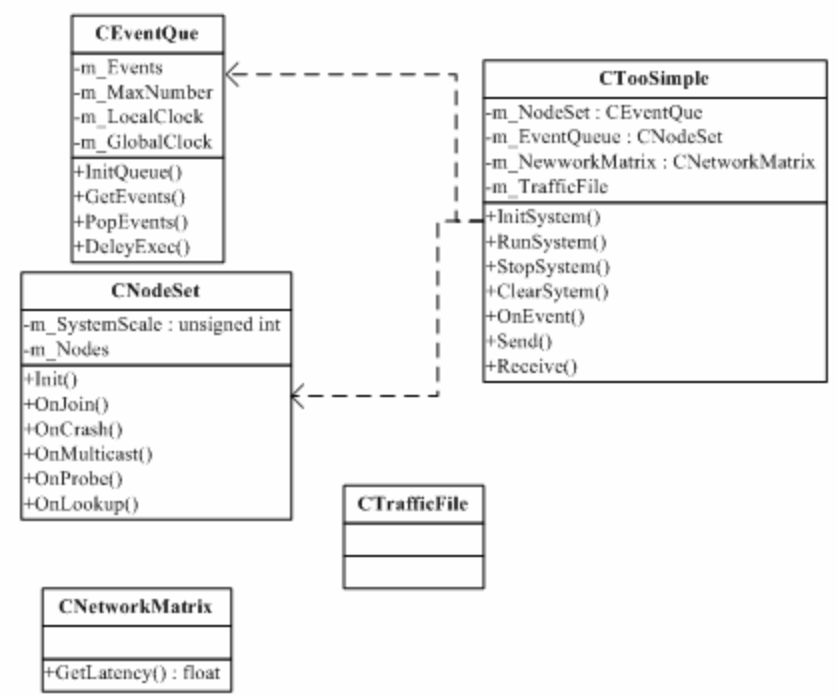

图 5.3 P2P 模拟平台接口设计示意图

图 5.3 是模拟平台接口设计的示意图。

5.2 性能测试

本节讨论的性能测试基于 5.1 节介绍的 P2P 模拟平台，运行在由 32 个处理器(Pentium IV CPU，1G 内存)的集群上，操作系统为 Linux Redhat7。0，结点之间用 2Gbps 的 Myrinet 连接。

本节试验分为两个部分：首先考察底层覆盖网 CAPU 的性能测试。网络中节点的带宽能力、生存期分布等基本参数设置采用[9]对 Gnutella 系统的实测结果。底层网络拓扑结构通过 Transit-Stub 模型[129]生成。作为对比，我们同时实现了 Gnutella 协议和[12]提出的 Gia 协议，并将这两个协议运行在与 CAPU 相同的网络设置上。本节考察的关键测试指标包括结点带宽占用、搜索步跳数、可扩展性等等。实验结果表明，相对于现有协议，CAPU 在搜索效率方面有着较大的性能改进。

接着，本文讨论流媒体系统几个核心算法的运行效率。

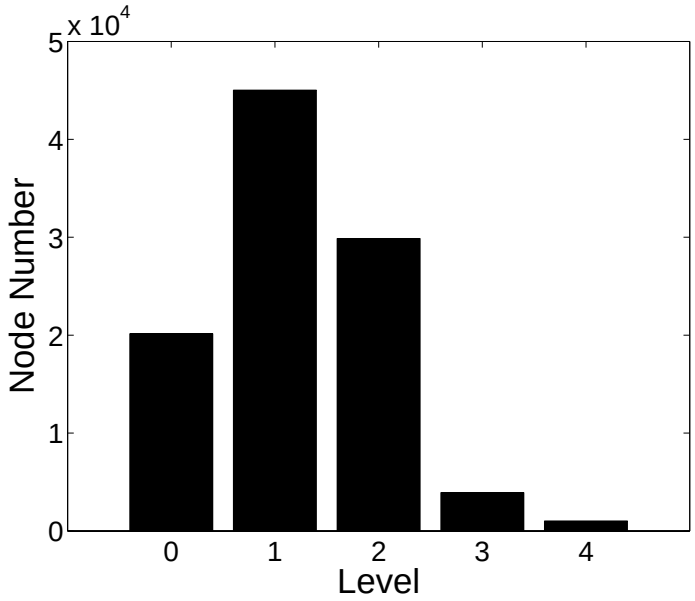

图 5.4 不同能力级别的节点数量分布

5.2.1 底层覆盖网协议性能

本节对 CAPU 协议性能的考察主要集中在整体系统性能，尤其是信息查询能力。下面列出的实验结果都是运行在 10，000 个节点的网络规模上，文件内容服从 Zipf 分布，其α系数为 0.95。

图 5.3 显示了不同能力节点的分布情况。这些数据来源于[9]中给出的 P2P 系统的实际测量数据。可以看出，除了能力最弱的零级节点外，其他级别的节点分布基本符合 Power Law 分布，也就是说，某个节点级别的节点数量与级别成负幂次分布。图 5.4 显示了不同级别节点上文件内容的分布。

图 5.4 说明了不同协议生成的拓扑结构中节点度数分布情况。作为对比的 RANDOM 图表示的是 Gnutella 协议所生成的随机图。可以看出，CAPU 协议中的根据节点能力调节拓扑的算法有效的使节点度数与节点能力正相关，有效的利用了系统中的节点异构性。

图 5.5 展示的是 CAPU 的搜索效率，作为对比的是 Gnutella[2]协议和 Gia 协议[12]。可以看出，CAPU 的搜索效率明显优于两个对比系统。

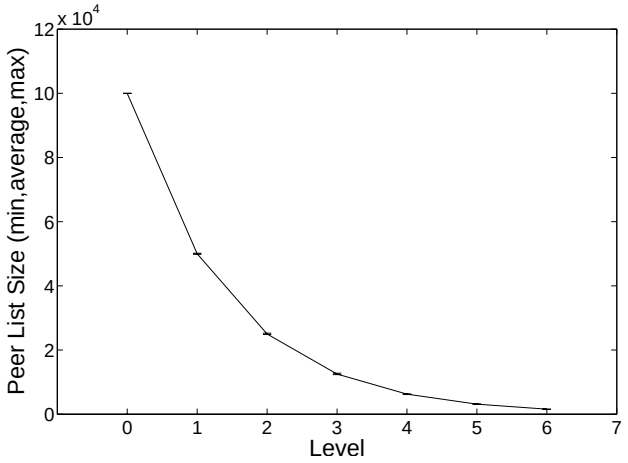

图 5.5 不同级别结点的信息列表的大小分布

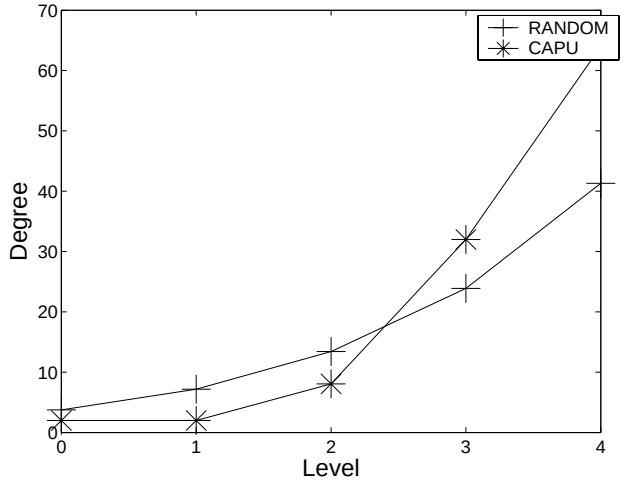

图 5.6 不同能力级别节点的度数分布

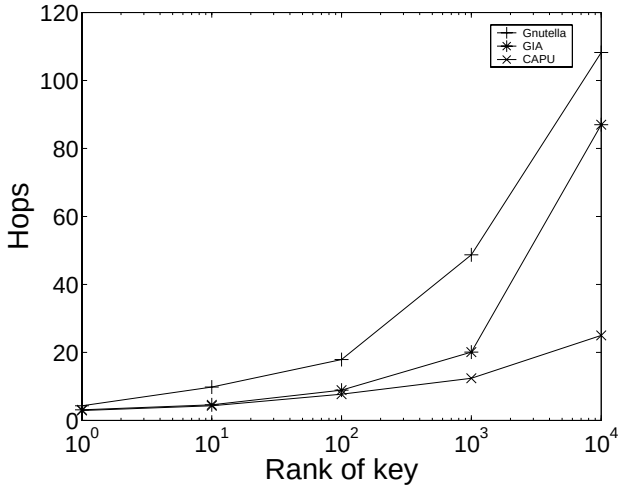

图 5.7 搜索效率，横坐标表示所搜索的内容在整个系统中流行程度

5.2.2 流媒体传输协议性能

图 5.7 给出了基于 CAPU 的流媒体系统下载流媒体文件速度的概率分布图。图 5.8 给出系统规模对于基于 CAPU 的流媒体传输系统的性能影响，作为对比的是服务器/客户机模型的流媒体系统。图 5.9 演示网络传输出现问题的时候，本文提出的流媒体系统自我恢复的能力。

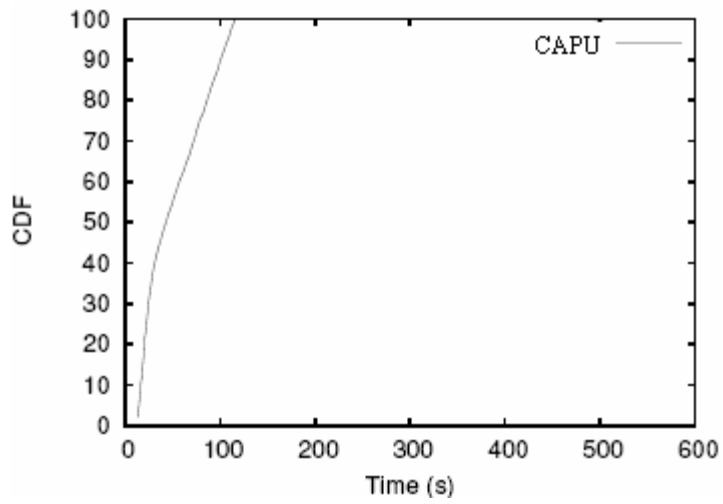

图 5.8 下载速度概率分布(CDF)

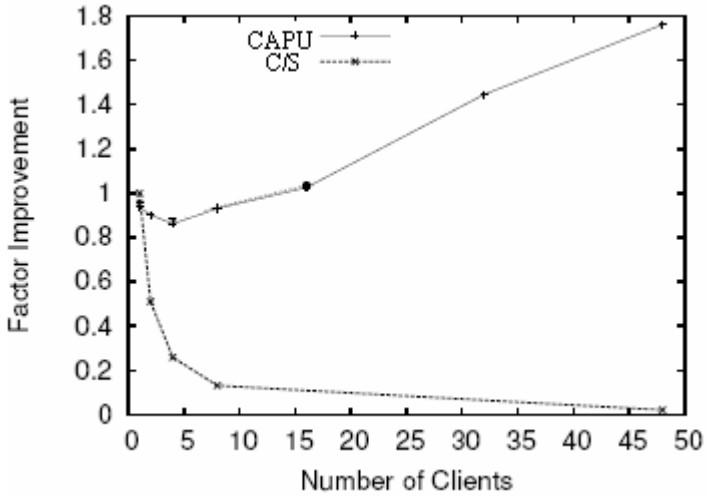

图 5.9 系统规模对下载速率的影响

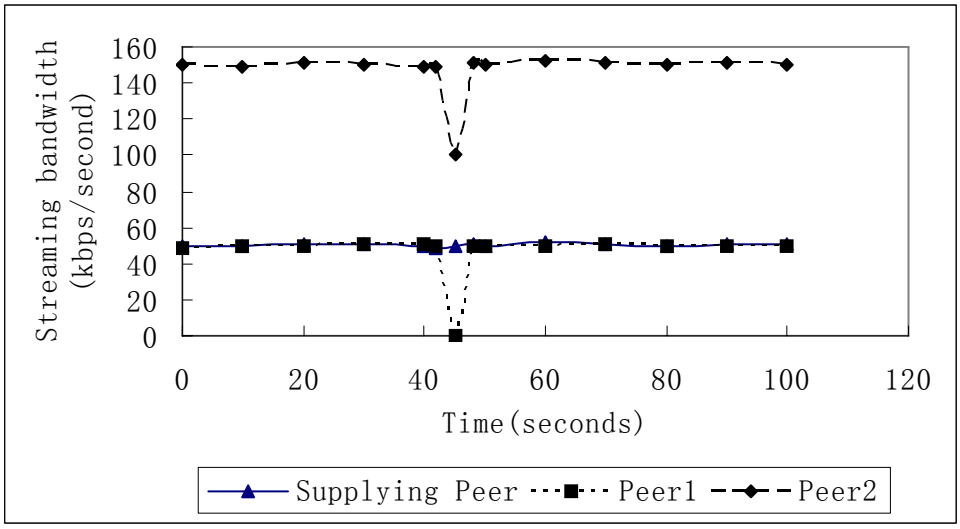

图 5.10 节点失效检测和恢复

第六章 结论

P2P 技术的迅速发展使其成为构建广域网中大型分布式系统的有力工具。P2P 流媒体系统作为 P2P 技术的重要应用之一也成为当前分布式系统领域的一个研究热点。本文针对 P2P 流媒体研究中的一些基本问题进行了深入研究，包括 P2P 覆盖网络设计、根据底层网络状况进行拓扑优化、结点信息收集算法、根据传输带宽选择合适数据发送节点及动态进行数据传输调度来得到最小延迟等。

本文的主要研究内容和贡献如下：

（1）提出了流媒体应用底层平台的非结构化 P2P 覆盖网协议 CAPU。这个系统解决了如何利用节点异构性来实现拓扑结构维护，使得 P2P 系统的结点可以根据各自能力以较低的网络开销信息收集量，并实现了高效的信息路由查找策略。

（2）设计了一个根据节点能力自适应的半结构化拓扑结构维护算法。这个算法保证覆盖网维持一个高吞吐的、负载平衡的、低网络直径并且具有高容错能力的半结构化拓扑结构。

（3）提出了主动信息发布扩散算法加两阶段搜索算法。根据节点级别决定节点掌握系统中文件信息的多少，并在此基础上设计了一个第一阶段为随机走步，第二阶段为无冗余广播搜索算法。这个搜索算法结合了非结构化 P2P 路由中两种基本模式的优点，在非结构化结构中实现了可收敛的高效信息查找机制。

（4）提出了根据底层网路状况自调整的拓扑优化算法。这个拓扑优化算法使得每个节点能根据从这些节点列表中选择网络状况较好的节点建立连接，从而为上层流媒体应用提供较好的基于覆盖网的网络连接。

（5）提出了数据发送节点动态选择算法和数据调度算法。根据网络状况和数据传输速率等因素来动态选择出“活动数据发送节点集合”，从这个集合中的节点请求媒体数据来获取最优的数据传输速率。并在数据发送节点间，根据对方服务能力，数据带宽等因素来分派每个数据发送节点所发送数据的次序，从而获得最小的数据接受延迟。

参考文献

[1] Napster. http://www.napster.com

[2] Gnutella. http://gnutella.wego.com.

[3] Clarke, I. A distributed decentralised information storage and retrieval system. Master's thesis, University of Edinburgh, 1999.

[4] Clarke, I., Sandberg, O., Wiley, B., and Hong, T.W. Freenet: A distributed anonymous information storage and retrieval system. In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability (Berkeley, California, June 2000). http://freenet.sourceforge.net.

[5] KaZaA. http://kazaa.com.

[6] BitTorrent. http://bitconjurer.org/BitTorrent/

[7] http://www.skype.com. December 2004.

[8] C—NET NEWS. Napster among fastest-growing Net technologies, Oct. 2000. http://news.com.com/2100-1023-246648.html.

[9] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. Proceedings of SPIE/ACM MMCN 2002, Jan. 2002.

[10] M. Bawa, H. Deshpande, and H. Garcia-Molina. Transience of peers and streaming media. First Workshop on Hot Topics in Networks (HotNets 2002), October 2002.

[11] Y. Chu, S. Rao, S. Seshan, and H. Zhang. A case for end system multicast. IEEE Journal on Selected Areas in Communications (JSAC), 20(8):1456–1471, October 2002.

[12] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003 .

[13] B. T. Loo, R. Huebsch, I. Stoica, and J. Hellerstein. The Case for a Hyrid P2P Search Infrastructure. In IPTPS 2004.

[14] Ben Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley. April 2001.

[15] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. (Middleware 2001). November 2001.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proceedings of ACM SIGCOMM 2001 . August 2001.

[17] Lv, Q., Ratnasamy, S., and Shenker, S. Can Heterogeneity Make Gnutella Scalable. In Proceedings of IPTPS '02. Cambridge, MA, Mar. 2002

[18] A. Qayyum, L. Viennot, A. Laouiti. "Multipoint relaying: An efficient technique for flooding in mobile wireless networks". INRIA research report RR-3898, 2000

[19] M. Russopoulos, and M. Baker. "CUP: Controlled Update Propagation in Peer-to-Peer Networks." USENIX 2003 Annual Technical Conference, San Antonio TX, Jun 2003

[20] M. Naor, U. Wieder, Know thy Neighbor's Neighbor: Better Routing for Skip-Graphs and Small Worlds, In Proceedings of IPTPS'04, San Diego, USA, Feb 2004

[21] I. Csisza´r, "Information Theoretic Methods in Probability and Statistics," Information Theory Soc. Rev. articles

[22] Mudhakar S., Bugra G. and Ling L. "Scaling Unstructured Peer-to-Peer Networks With Multi-Tier Capacity-Aware Overlay Topologies" Proceeding of ICPADS 2004..

[23] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," ACM Transactions on Computer Systems, vol. 8, no. 2, May 1990.

[24] S. E. Deering, Multicast Routing in a Datagram Internetwork, Ph.D. thesis, Stanford University, Dec 1991.

[25] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L.Wei, "The PIM Architecture for Wide-Area Multicast Routing," IEEE/ACM Transactions on Networking, vol. 4, no. 2, April 1996.

[26] S. Floyd, V. Jacobson, C.G. liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," IEEE/ACM Transaction on networking, vol. 5, no. 4, pp. 784–803, Dec. 1997.

[27] J.C. Lin and S. Paul, "A reliable multicast transport protocol," in Proc. Of IEEE INFOCOM'96, 1996, pp. 1414–1424.

[28] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," ACM Transactions on Computer Systems, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[29] K.P. Birman, M. Hayden, O.Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, “Bimodal multicast,” ACM Transactions on Computer Systems, vol. 17, no. 2, pp. 41–88, May 1999.

[30] Patrick Eugster, Sidath Handurukande, Rachid Guerraoui, Anne-Marie Kermarrec, and Petr Kouznetsov, “Lightweight probabilistic broadcast,” in Proceedings of The International Conference on Dependable Systems and Networks (DSN 2001), Gothenburg, Sweden, July 2001.

[31] R. L. Carter and M. E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report BUCS96-006, Computer Science Department, Boston University, March 1996.

[32] V. Paxson. End-to-End Internet Packet Dynamics. In Proc. of the ACM SIGCOMM, Cannes, France, September 1997.

[33] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In Proc. of IEEE Infocom'96, CA, May 1996.

[34] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan,Dejan Kostic, Jeff Chase, and David Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In ACM OSDI’02, 2002

[35] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. Annual Conference of the Special Interest Group on Data Communication (SIGCOMM 2001). August 2001.

[36] C. Plaxton, R. Rajaraman, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. In Proceedings of the ACM SPAA (Newport, Rhode Island, June 1997), pp. 311–320.

[37] J. Jannotti, D. K. Gifford, and K. L. Johnson, "Overcast: Reliable multicasting with an overlay network," in USENIX Symposium on Operating System Design and Implementation, San Diego, CA, October 2000.

[38] Shelly Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John Kubiatowicz, “Bayeux: An Architecture for Scalable and Faulttolerant Wide-Area Data Dissemination,” in Proc. of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), Port Jefferson, NY, June 2001.

[39] Dejan Kostic, Adolfo Rodriguez, Jeannie R. Albrecht, Amin Vahdat: Bullet: high bandwidth data dissemination using an overlay mesh. In Proceedings of the 19th ACM Symposium on Operating System Principles, October 2003. 282-297

[40] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: High-bandwidth Content Distribution in Cooperative Environments. In Proceedings of the 19th ACM Symposium on Operating System Principles, October 2003.

[41] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE JSAC, 20(8), Oct. 2002.

[42] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou. Resilient Peer-to-Peer Streaming. In 11th IEEE International Conference on Network Protocols (ICNP'03) November 04 - 07, 2003 Atlanta, Georgia

[43] K. Lai and M. Baker, “Nettimer: A tool for measuring bottleneck link bandwidth,” in Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, Mar. 2001.

[44] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based Congestion Control for Unicast Applications. In Proceedings of the ACM SIGCOMM, August 2000.

[45] Z. Wang and J. Crowcroft. Bandwidth-delay Based Routing Algorithms. In IEEE GlobeCom, November 1995.

致　谢

衷心感谢导师郑纬民教授、汪东升教授对本人的精心指导，他们的言传身教将使我终生受益。

在清华大学计算机系高性能计算技术研究所的三年学习和研究生涯当中，承蒙郑纬民教授、沈美明教授、汤志忠教授、石纯一教授、汪东升教授、杨广文教授、温东婵教授、舒继武副教授、陈文光副教授、张悠慧老师、鞠大鹏老师、武永卫老师、毛希平老师以及实验室其它老师的热心指导与帮助，不胜感激。

还要感谢黎明、胡进锋、马永泉、宁宁、高崇南、孙竞等同我在一起工作的同学们。

感谢陈明、史树明、刘学铮、李卢、顾瑜、嵩天、杨济、王庆、曲绍刚、姜建锦、陈永健、陈康等同学的帮助与支持。

最后，特别的感谢要献给我的家人，是他们给了我奋斗的动力与勇气。

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成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____日　期：____

个人简历、在学期间的研究成果及发表的论文

个人简历

姓　名：董海韬

出生年月：1980 年 2 月 11 日

出　生　地：河北省保定市

学习经历：

1997 年 9 月－2002 年 7 月

就读于中国科技大学计算机系，计算机科学与技术专业，于 2002 年 7 月获工学学士学位。

2002 年 9 月－2005 年 7 月

就读于清华大学计算机科学与技术系，计算机科学技术专业至今

研究成果

攻读硕士学位期间参加的研究项目

2002－2003　863 高科技计划项目“集群服务器功能软件“

2003－2005　国家自然科学基金项目“对等计算及广域网虚拟平台”

在国际和国内学术刊物上发表的论文

[1] **Haitao Dong**, Jinfeng Hu, Weimin Zheng, Dongsheng Wang, Ming Li “Janus: Build Gnutella-like file sharing system over structured overlay”, In Proc. Grid Cooperative Computing, GCC2004. Published by Springer Varlag Lecture Notes on Computer Science. Vol. 3251.

[2] Jinfeng Hu, **Haitao Dong**, Weimin Zheng, Dongsheng Wang, and Ming Li. Twins: 2-hop Structured Overlay with High Scalability. International Conference on

Computer Science (ICCS 2004). June 2004. Published by Springer Varlag Lecture Notes in Computer Science, Volume 3036/2004, page 174-183. SCI 检索号：BAG03.

[3] Jinfeng Hu, Ming Li, Hongliang Yu, **Haitao Dong**, and Weimin Zheng. PeerWindow: An Efficient, Heterogeneous, and Autonomic Node Collection Protocol. 2005 International Conference on Parellel Processing (ICPP-05). June 2005.

[4] Jinfeng Hu, Ming Li, Weimin Zheng, Dongsheng Wang, Ning Ning, and **Haitao Dong**. SmartBoa: Constructing P2P Overlay Network in the Heterogeneous Internet Using Irregular Routing Tables. 3rd International Workshop on Peer-to-Peer Systems (IPTPS 2004). February 2004. Published by Springer Varlag Lecture Notes in Computer Science, Volume 3279/2005, page 278–287.

[5] **Haitao Dong**, Lu Li, Weimin Zheng, Dongsheng Wang, “CAPU: Enhancing P2P File Sharing System with Capacity Aware Topology”, under submission.