摘 要

随着计算机网络的不断发展，互联网已经成为了人类社会主流的一个重要组成部分。人们希望互联网能够不断地提供应用所需的各种网络服务。特别是，以视频会议、视频点播、远程教育等为代表的新型多媒体组播应用的大量涌现，对组播通信服务提出了迫切的需求。

近年来 P2P 技术的快速发展，基于应用层组播 P2P 流媒体传输，引起了许多大学和公司的重视并纷纷开展研究。与 IP 组播相比，应用层组播具有灵活和易实施的特点，但是因为终端主机可以自由地退出组播树，应用层组播也存在数据传递易中断的缺点，这对实时性要求严格的视频直播应用的影响尤为严重。

本论文在对现有应用层组播协议进行详细研究的基础上，针对现有系统中存在的缺陷提出一种具有多层次应用层结构的应用层组播协议。该协议具有较高的效率和良好的可扩展性，主要面向实时的多媒体应用，减少分组延迟，保证流媒体数据传输的实时性。论文的具体研究和实现工作主要包括以下几个方面：

●应用层组播系统结构的研究。对对等型，代理型，服务型三种应用层组播系统结构进行分析总结，提出了一种适用的应用层组播系统协议栈模型。

●应用层组播协议的研究。仔细分析现有的应用层组播系统的优势和不足，设计新的组播协议。协议考虑到底层网络拓扑特征，避免数据包在代价昂贵的链路上传输，从而减少延迟。同时采用基于斐波那契序列的组播算法进行群内组播。另一方面引入动态组管理策略，具有较高的扩展性。

●对所设计的协议进行仿真，通过两个不同的仿真实验来分析该协议在 AED，ALS 和 ACS 性能上的优劣。第一个仿真实验考察的是单源情况，第二个仿真实验考察的是多源情况，在每种情形下变化组播组的成员数目。

关键词：应用层组播； 覆盖网； 仿真；NS2

Abstract

With the development of computer network, Internet has been becoming an important part of the mean streaming of human being. People hope Internet could provide all kinds of network services the applications need. Especially, the emergence of many new multimedia group applications, such as video conferencing, video-on-demand and distance learning, require multicast communication service imminently.

Recently as the development of P2P techniques, Application level multicast (ALM) based on P2P also known as end system multicast or overlay multicast has become a very popular research in many college and corporation. Compared with IP multicast, ALM is more flexible and deployable. But data delivery in ALM tree can be easily interrupted by departure of end hosts, which may lead to degradation of QOS in time sensitive applications such as live streaming.

On the basis of the serious study on the existing application layer multicast protocol, this paper proposed a multi-level structure of the application layer multicast protocol as for the deficiencies in currently systems. Such protocol has a high efficiency and excellent scalability, mainly for real-time multimedia applications, to reduce the packet delay, ensuring that real-time data transmission of streaming media. The exactly research and specific work of thesis include the following:

● The studies of application layer multicast system architecture. Analyzed and summarized of three popular application layer multicast system architecture, a stack model of application layer multicast system is proposed.

● The research of application layer multicast protocol. We designed a new multicast protocol under careful analysis of the existing application layer multicast systems strengths and weaknesses. The protocol considerate the characteristics of the underlying network topology to avoid costly packets in the transmission link, thereby reducing the delay. At the same time, it adopts the multicast algorithm based

on the Fibonacci sequence to realize group multicast. On the other hand a dynamic group management strategy is introduced, with a higher scalability.

● The simulation of the protocol. Analyze the advantages of AED, ALS, ACS performance through the two different simulation experiments. The first simulation is about single-source case. The second simulation is about multi-source situation, in each case changes the number of multicast group members.

Keywords: Application Layer Multicast; Overlay Network; Simulation; NS2

厦门大学学位论文原创性声明

本人呈交的学位论文是本人在导师指导下，独立完成的研究成果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合法律规范和《厦门大学研究生学术活动规范（试行）》。

另外，该学位论文为（　　）课题（组）的研究成果，获得（　　）课题（组）经费或实验室的资助，在（　　）实验室完成。（请在以上括号内填写课题或课题组负责人或实验室名称，未有此项声明内容的，可以不作特别声明。）

声明人（签名）：郭鑫

2009年6月5日

厦门大学学位论文著作权使用声明

本人同意厦门大学根据《中华人民共和国学位条例暂行实施办法》等规定保留和使用此学位论文，并向主管部门或其指定机构送交学位论文（包括纸质版和电子版），允许学位论文进入厦门大学图书馆及其数据库被查阅、借阅。本人同意厦门大学将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于：

（　　）1.经厦门大学保密委员会审查核定的保密学位论文，于　年　月　日解密，解密后适用上述授权。

（　　）2.不保密，适用上述授权。

（请在以上相应括号内打“√”或填上相应内容。保密学位论文应是已经厦门大学保密委员会审定过的学位论文，未经厦门大学保密委员会审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权。）

声明人（签名）：郭鑫

2009年6月5日

第一章 绪论

1.1 组播

在 Internet 上，流媒体技术如视频会议和视频点播等应用日益广泛。点对点传输的单播方式已经不能适应这一类业务的传输特性—“单点发送，多点接收”，因为服务器必须为每一个接收者提供一份相同内容的 IP 报文拷贝，使得网络上重复传输大量相同内容的报文，占用了大量资源。在这种情况下组播[1](multicast)应运而生，它的出现解决了网络数据冗余的问题，尤其对于音频，视频数据，可以节省大量网络资源，解决一个主机向多个接收者发送数据的问题。组播是指同时将数据分组高效地发送给网络上的一组目标主机，而不关心这些主机位于网络的什么位置。

在真正的组播技术产生以前，可以使用两种不同的方式来实现组播功能。第一种是采用广播方式，将分组广播给网络上的所有主机，所有的目标主机都能收到分组；第二种方式是假设源节点知道所有目标主机，通过顺序单播的方式将分组依次发送给目标主机。这两种方式效率都比较低。

在现有的网络服务中，组播（Multicast）通信服务一直占有重要的地位。不同于单播（Unicast）通信发送者需要向各个接收者单独发送数据报文，在组播通信中发送者只需要向所有接受者发送一个原始数据报文，该报文的拷贝由具有组播功能的中间系统完成，并最终转发给各个接收者。因此，组播通信能节省大量的网络通信资源，并提高通信效率，这使其成为了支持以实时多媒体应用为代表的下一代新型组播应用（如：视频会议、视频点播、远程教育、网络电视等）的关键技术之一。

然而，新型组播应用的种类非常之多，它们在组播组的规模、组内数据发送源的数量、组成员的动态性、以及对带宽和延时要求等方面存在着很大的差异。表 1.1 给出了几种典型组播应用的特点比较[2][3]。由此可见，如何适应不同应用的差异，满足它们的需求，是组播通信面临的新课题。

表 1.1 几种典型组播应用的特点与比较

应用	组播数据源	组规模	带宽	延时	组成员动态性
视频会议	多	小	中	小	低
远程教育	单，少	中	中	小	低
网络电视(IPTV)	单	大	大	小	高
视频点播(VoD)	单	大	大	小	高
多方游戏	多	中、大	中	小	高
在线股票/新闻	单	大	小	小	高
分布式仿真	单，少	中	大	视情况	低
分布式 Web 缓存更新	单，少	中	大	大	低

1.2 IP 组播

在 Internet 体系结构中，Deering 首先提出 IP 组播体系结构[4]，组播功能在网络层实现，组播分组的复制和转发都在网络的路由器上进行。IP 组播是实现组播分组转发的有效方式，因为它可以使在全网范围内分组复制的数量达到最少。

1.2.1 IP 组播地址

IP 组播通信必须依赖于组播地址，在 IPv4 中它是一个 D 类 IP 地址，范围从 224.0.0.0 到 239.255.255.255，并被划分为局部链接组播地址、预留组播地址和管理权限组播地址三类。其中，局部链接组播地址范围在 224.0.0.0—224.0.0.255，这是为路由协议和其它用途保留的地址，路由器并不转发属于此范围的 IP 包；预留组播地址为 224.0.1.0—238.255.255.255,可用于全球范围(如 Internet)或网络协议；管理权限组播地址为 239.0.0.0—239.255.255.255，可供组织内部使用，类似于私有 IP 地址不能用于 Internet，可限制组播范围。

1.2.2 IP 组播的标准模型

Stephen Deering 在文献[4]中描述了 IP 网络中标准的组播模型。

IP 组播是一种开放的服务模型，模型中具有发送者和接收者两个概念。主机通过 IGMP[5]报文与本地路由器交互，成为接收者。发送者只需将报文的目的地址指定为组播组地址就可实现发送。IP 组播使用 D 类 IP 地址作为组播组地址。

IP 组播没有提供技术用来限制用户创建一个组播组，接收组播组的数据，和向组播组发送数据。组成员身份只是实现了接收者可以收到数据，不提供任何访问控制的功能。为了收到一个组播组的数据，用户只需用 IGMP 协议与本地路由器联系。当一个主机变成组播组的接收者之后，它可以收到组播组的所有数据报文，而不管数据报文的发送者是谁，是否是恶意的发送者。发送者不需要成为组的接收者，只需将报文的目的地址指定成组播组地址，就可以实现向组播组发送报文。发送者不能对自己使用的组地址进行保留，限制别的用户使用与自己相同的组地址。

总之，IP 组播的服务模型没有提供组的管理。IP 组播数据报文与所有的 IP 数据报一样，提供尽力而为服务，没有可靠性保证。IP 组播传送功能是通过在 IP 网络设备上运行相应的组播路由协议来实现的。

1.2.3 IP 组播路由协议

组播数据沿一个连接所有组播组成员的树型结构发送，这个树型结构称为组播树。组成员可以动态的加入和退出，组播树也必须同时动态更新。

根据构造方法的不同，组播树可以分为源点树(Source-based Tree)和共享树(Shared Tree)。源点树以组播源点为根构造到所有组播组成员的生成树，通常也称为最短路径树(SPT，Shortest Path Tree)。共享树也称为 RP(汇聚点)树或基于核心的树(CBT，Core-Based Tree)，它的构造方法是以网络中的某一个指定的路由器为根节点，该路由器称为 RP，由此节点生成包含所有组成员的树。使用共享树时，组播源需要首先把组播数据发送到 RP 路由器，再由 RP 转发给其他的组成员。

组播路由协议的任务就是构造组播树，根据对网络中的组播成员的分布和

使用的不同，组播路由协议分为两类：密集模式(DM，Dense Model)路由协议和稀疏模式(SM，Sparse Model)路由协议。

●DM 路由协议通常用于组播成员较为集中，数量较多，网络中有少数发送者和大量的接收者，并且有足够带宽的链路环境，比如公司或园区的局域网。DM 路由协议有：距离向量组播路由协议(DVMRP)、组播 OSPF 协议(MOSPF)和协议无关组播协议－密集模式(PIM-DM)等。这些协议构造从发送者到接收者的最短路径。

●SM 路由协议适用于组成员稀疏分布，接收者比较少，发送者/接收者位于分散地域，大型异构的互联网络环境。SM 路由协议有：基于中心的分布树协议(CBT)和协议无关组播协议－稀疏模式(PIM-SM)等。这类协议构造从 RP 或 Core 到接收者的最短路径。

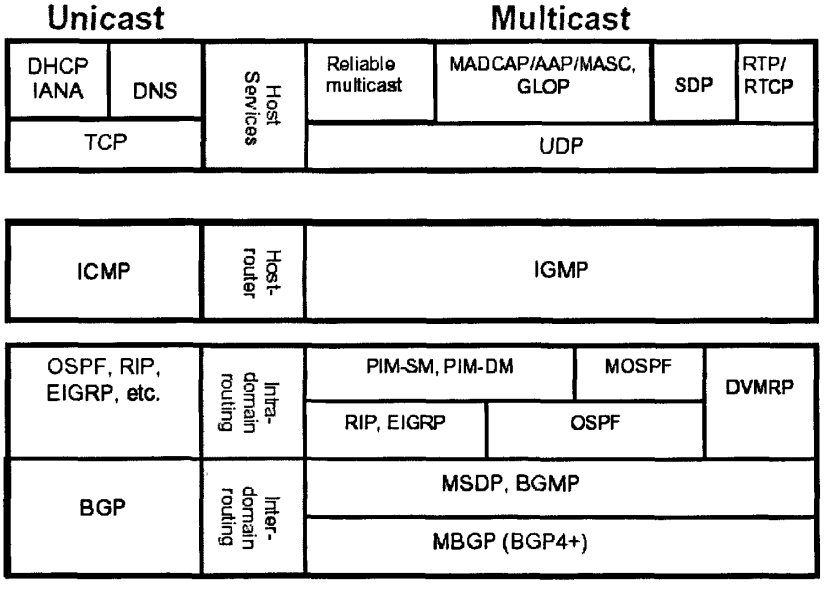

图 1.1 IP 组播的体系结构

IP 组播的体系结构如图 1.1 所示。图中左边是单播协议，右边是组播协议。组播协议从下而上，最底层是域间和域内的组播路由，它们之上是主机-路由器接口 IGMP，最上层是如 RTP/RTCP 等的一些主机服务。

1.2.4 IP 组播存在的问题

目前 IP 组播的服务模型和协议存在着一些问题，使得 IP 组播至今没有能

在 Internet 上得到广泛部署[6]。这些问题既有技术上的缺陷又有商业原因，文献[6]对其进行了详细的讨论。ISP 不愿意部署 IP 组播的主要原因是：

●路由器换代，为了实现组播部署，ISP 需要对路由器进行换代，提早结束目前尚在使用的路由器的工作周期，增加了 ISP 的营运成本。

●跨域的组播管理，每个 ISP 都对域内的组播进行独立的管理。当需要跨域的组播服务时，ISP 之间如何进行协调策略。

●域内组播的管理，IP 组播服务模型及协议实现没有提供组播管理的技术，ISP 如何对组播进行管理.

●组播与单播相比，节省带宽，降低延时，提高效率。然而部署组播的代价比单播要昂贵的多，ISP 是否有利可图。

当前 IP 组播还存在很多待解决的技术问题，这些技术问题也影响到 IP 组播被广泛的应用。IP 组播主要存在的技术问题有：

●组播组的管理问题，目前的 IP 组播服务模型及协议没有提供组播组管理的技术，对组的发送者和接收者没有进行访问控制，这将导致很多问题，如恶意发送者的泛洪攻击，组播地址冲突，未经授权的接收者和发送者。

●组播的安全问题，IP 组播使用无连接的协议 UDP 来避免响应风暴。由于 UDP 是一个无连接的协议,它不使用 ACK 或 NACK 来确保可靠传送,组播也不能被防火墙检测到,因此,最普通的防火墙类型(应用程序网关)不能对组播进行安全认证。当前的 IP 组播服务和体系结构并不执行任何认证。

●组播服务质量问题，IP 组播是一种尽力而为的服务，要想在它之上实现更高质量的服务，例如可靠性，拥塞控制，流控制等这些功能，比在 IP 单播上实现要困难很多。

●组播地址分配问题，IP 组播要求每个组能够动态的从组播地址空间中得到一个全局唯一的组播地址。但是在可扩展，分布式，相容的网络环境中，这一点很难得到保证。

IP 组播这些社会因素和技术上的问题使其没有成功在 Internet 上部署起来，有学者甚至预言如果 IP 组播继续保持复杂性和管理的困难性，它将很难实现广域范围的部署。

在应用需要组播服务支持，而 IP 组播又没有实现部署的情况下，研究者提

出了“应用层组播”的概念。

1.3 应用层组播

在最初的提议经过十多年后，研究学者们开始考虑在网络层实现组播功能是否最为合适，并提出了很多 IP 组播的替代方案。很多组织提议在应用层实现组播特征，这样一种模型称之为应用层组播(ALM)，现在被广泛认为是替代 IP 组播的合理方案。应用层组播又称为覆盖网络组播、端系统组播和基于主机的组播。

1.3.1 应用层组播基本思想和性能评价指标

应用层组播使用传统单播提供的服务，在终端主机的应用层上实现组播的特征例如组关系、寻址、组播路由和报文复制，网络只需要提供尽力传输的单播功能。应用层组播的基本思想入图 1.2 所示。图 1.2a 是网络层组播的实现示意图，分组由网络中的路由器进行复制，如果主机 A 需要向主机 B、C、D 发送分组，则分组在路由器 1 处进行复制，而在应用层组播中，分组在端系统处进行复制。端系统构成逻辑上的覆盖网络，应用层组播的目标就是便于进行数据传输，构造并维护高效的覆盖网络。评价应用层组播协议的性能通常采用下面的指标。

（1）数据路径的质量。数据路径的质量一般采用两个参数来衡量：强度(stress)、伸展度(stretch)[24]。强度定义为每条链路或者每个路由器在传输组播分组时发送相同分组的次数。对于 IP 组播来说链路上没有多余的数据报文复制，因此对于网络中的每条链路或节点来说，stress 尺度值均为 1。伸展度定义为数据分组从源沿着覆盖网络中路径到达组成员的路径长度和直接的从源到组成员的单播路径长度的比值，这是为每个组成员定义的尺度。显然，采用 1.2b 中的应用层组播方案，由于每条路径就是单播路径，每个成员的伸展度均为 1。在图 1.2c 中，AB 之间的伸展度为 1，AC 之间的伸展度为 1.5，AD 之间的伸展度为 3，平均伸展度等于(1+1.5+3)/3=1.83。

(2)控制信息负担。在覆盖网络中，每一个成员都要和它们的对等体周期性地交换刷新信息。这些信息在组中不同的路由器、链路和组成员中构成控制信

息负担。控制信息负担是衡量应用层组播可扩展性一个非常重要的度量尺度。

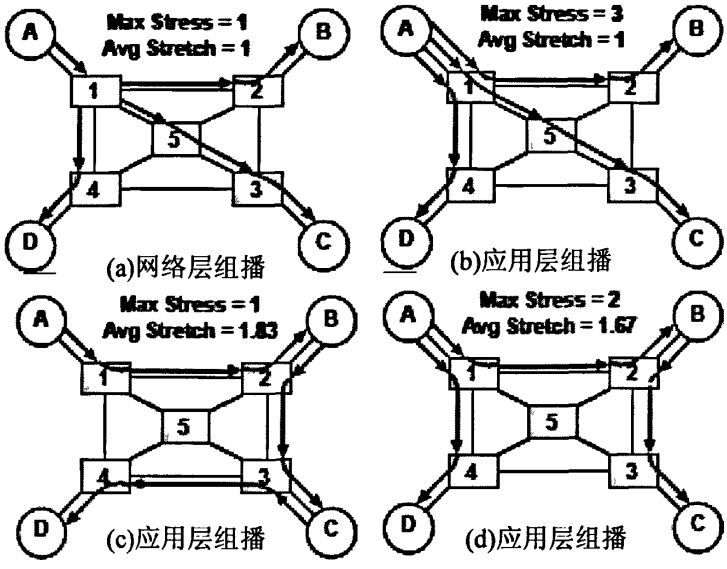

图 1.2 网络层组播和应用层组播

除了这些指标，还有其它一些衡量应用层组播的性能指标，包括数据生成树中成员的度的分布，数据传输延迟等等。

不同的应用层组播协议对这些性能指标有不同的考虑，因此不同的协议将使用不同的方法构造覆盖网络。图 1.2 给出了在相同的网络拓扑结构下，用不同的应用层组播协议构造不同覆盖网络的例子。假定图中每条链路都具有单位长度。图 1.2b 直接采用源节点 A 到其它节点的单播路径进行组播分组转发，其伸展度为 1，A 和路由器之间的链路强度为 3，其它链路强度均为 1。一般来说，在 N 个节点的组中如果全部采用单播路径进行组播转发，那么可能出现的最大强度为口因，出现在源节点的某条链路上，而组成员的伸展度为 1。另外由于数据源节点和组播组中所有其它节点交换更新消息，因此最坏情况下负载为 O(N)的。图 1.2c 采用环形组播方案，其链路最大强度为 2，平均伸展度为 O(N)。图 1.2d 是上述两种方案的折衷，最大强度和平均伸展度都位于两者之间。

1.3.2 应用层组播的优势和局限性

优势：(1)应用层组播最明显的优势是不需要底层网络结构，对路由器没有多余的要求；(2)不需要维护网络的状态信息，与当初设计无状态网络的初衷相符，可扩展性问题得到了解决 ALM 的可扩展性仅限于提供 ALM 的技术和实现

方法；(3)不再需要 D 类 IP 地址，因为 ALM 方案可以指定自己的寻址方案；(4)单播方案能够被运用到组播应用中；(5)应用层比网络层具有更大的灵活性，应用层组播可以根据应用的需求采取更多的手段来提高服务质量。

局限性：端系统的负载明显增加使得 ALM 不能像 IP 组播那样高效。这是因为在 IP 组播中分组的复制是在分支路由器处进行的，由于在 ALM 中路由器不提供任何特殊的服务，这样就不可避免分组在同一条单播链路上重复分发。ALM 的一个主要目标是组织的覆盖网络结构尽可能减少分组的重复分发，然而在应用层上要准确判断底层网络的拓扑结构是相当复杂和困难，所以这将导致覆盖网结构没有 IP 组播分发树那样高效。另外由于应用层组播通过端节点来复制和转发数据，应用层组播和网络层组播相比通常延时较大。

1.3.3 一个稍微复杂点的例子

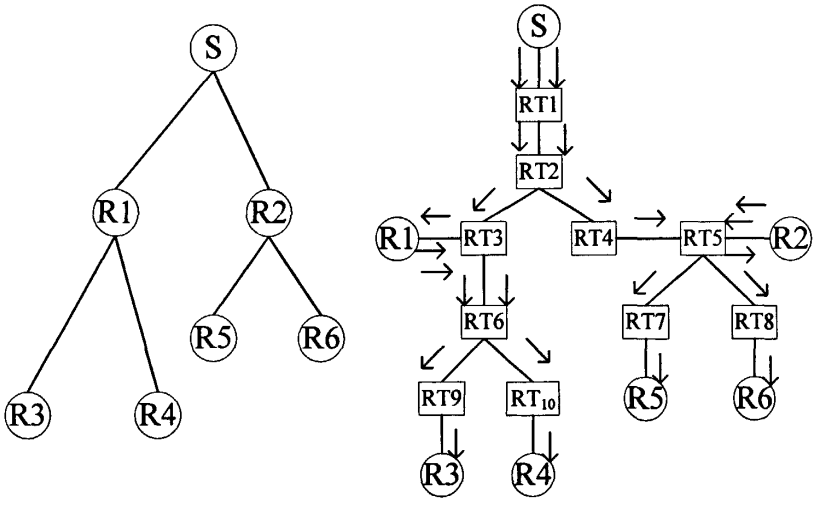

图 1.3 应用层组播

图 1.3 是应用层组播的一个稍微复杂一点的例子，通过该图进一步说明了应用层组播的特点。图 1.3a 展示了从应用层上看到的端节点的逻辑生成树的拓扑结构，图 1.3b 显示了实际的网络拓扑结构。从源 S 发送分组到所有的接收节点，如果只考虑物理链路上分组的数目，可以发现在单播组播、应用层组播和 IP 组播情况下物理链路上分组的数目分别是 33，23，16；与单播组播相比，IP 组播可以减少 52%的资源而应用层组播可以减少 33%。

1.4 研究内容和论文结构

1.4.1 研究内容

本论文在对现有应用层组播协议进行详细研究的基础上，针对现有系统中存在的缺陷提出一种具有多层次应用层结构的应用层组播协议。该协议具有较高的效率和良好的可扩展性，主要面向实时的多媒体应用，减少分组延迟，保证流媒体数据传输的实时性。论文的具体研究和实现工作主要包括以下几个方面：

●应用层组播系统结构的研究。对对等型，代理型，服务型三种应用层组播系统结构进行分析总结，提出了一种应用层组播系统的协议栈模型。

●应用层组播协议的研究。仔细分析现有的应用层组播系统的优势和不足，设计新的组播协议。协议考虑到底层网络拓扑特征，避免数据包在代价昂贵的链路上传输，从而减少延迟。同时采用基于斐波那契序列的组播算法进行群内组播。另一方面引入动态组管理策略，具有较高的扩展性。

●对所设计的协议进行仿真，通过两个不同的仿真实验来分析该协议在AED，ALS 和 ACS 性能上的优劣。第一个仿真实验考察的是单源情况，第二个仿真实验考察的是多源情况，在每种情形下变化组播组的成员数目。

●论文的总结和展望。

1.4.2 论文结构

本文分五章，各章节的内容安排如下：

第一章为绪论，简单介绍了 IP 组播极其存在的问题，并对此提出了应用层组播，概述了应用层组播的基本思想，性能评价，并对比 IP 组播阐述了应用层组播的优势和局限性，通过一个稍微复杂一点的例子说明了应用层组播的特点。最后给出了本论文的研究内容以及组织结构。

第二章介绍了应用层组播的相关研究工作。首先概述了 Overlay 网络的定义，抽象模型以及基于 Overlay 网络的应用层组播路由的问题描述。随后分析现有的应用层组播方案，并在此基础上总结出三中应用层组播系统结构，提出了一种应用组播系统的协议栈模型。

第三章研究了基于树型拓扑结构的应用层组播协议，给出了一种延迟较小的应用层组播设计方案—HFTM。首先分析了要获得较小组播延迟所需要考虑的问题，在此基础上，给出了一种考虑主机位置，分层分群的层次化结构，另外提供了一种动态的组管理方法，可以在组成员变化时以较低的通讯代价维持正常的组播通讯。

第四章通过模拟实验表明HFTM协议与其他经典应用层组播协议相比具有更好的组播延迟性能

第五章是结束语，总结了本文的主要工作，贡献和创新点，并展望了下一步的研究工作。

第二章 基于 Overlay 网络的应用层组播路由协议的研究

应用层组播的基本思想是在不改变网络设施，不依赖于网络层是否提供组播服务支持的情况下，利用网络边缘的端用户或是专门架设的服务器处理资源和网络资源，在应用层实现组播服务。

本章首先是 Overlay 网络的概述，包括它的定义，技术优势和抽象模型；接着介绍应用层组播路由的问题描述，包括组播服务质量抽象描述和问题定义。

2.1 OVERLAY 网络概述

目前实现应用层组播的基本技术是将组播组的所有端用户组成一个应用层的逻辑 Overlay Network（覆盖网络），利用覆盖网络实现组播组管理、组播管理、组播数据分发等一系列组播相关的功能。

2.1.1 Overlay 网络的定义

覆盖网络[7]是在真实的网络拓扑之上建立的逻辑网络，如图 2.1 给出了一个覆盖网络的模型。

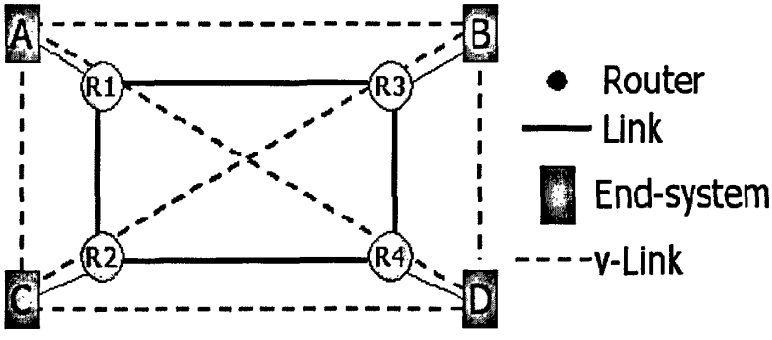

图 2.1 覆盖网络示意图

图 2.1 中圆形为路由器，方形为端用户，黑色实线表示物理链路。端用户、路由器、以及连接它们的物理链路构成了真实的物理拓扑。端用户之间的虚线表示端用户之间建立的一条网络连接，TCP 连接或是 UDP 连接。端用户和连接他们之间的虚线构成了一个覆盖网络。

覆盖网络是一个逻辑的“网络”，它具有虚拟的拓扑，拓扑中的“节点”是端用户，“边”是“节点”之间的连线，“边”也可以具有参数。覆盖网络中边的参数是对应真实物理拓扑中的路径参数的代数叠加。在覆盖网络中，将建立了边关系的两个端用户互称为覆盖网络中的“邻居”。

假定 A 与 B 之间的 IP 单播路径是 A－R1－R3－B。可为覆盖网络中的边 AB 定义代价函数 Cost。

Cost（A，B）＝Cost（A，R1）＋Cost（R1，R3）＋Cost（R3，B）

也可为覆盖网络的边 AB 定义延时函数 Delay。

Delay（A，B）＝Delay（A，R1）＋Delay（R1，R3）＋Delay（R3，B）（2－2）

同理，可为覆盖网络的每一条边定义代价和延时函数。

覆盖网络利用端用户和端用户之间建立的连接来实现组播服务，如图 2.2 所示。假定 A、B、C、D 构成了一个应用层组播的覆盖网络。A 作为发送者，B、C、D 作为接收者时的情况。

应用层组播利用覆盖网络中的边，即主机之间建立的邻居关系来实现应用层组播的转发路径。A 将数据发送到 B 和 C，B 将数据转发到 D。

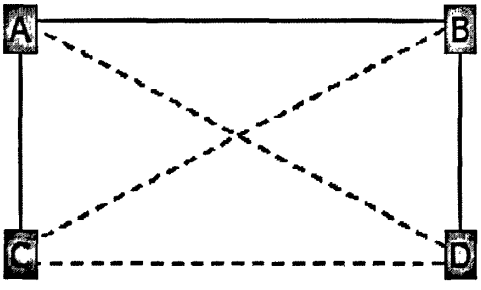

图 2.2 应用层组播示意图

应用层组播与 IP 组播的本质区别在于：

●在 IP 组播中，组播数据包的复制与转发由路由器完成；在应用层组播中，数据包的复制与转发可以由端用户，专门的服务器，或是边界路由器完成。

●在应用层组播中，端用户看见的是覆盖网络的拓扑，底层物理拓扑被隐藏了。

●在 IP 组播中，组成员关系分布在路由器上；在应用层组播中，组成员关系可以保留在 RP（汇聚点）上，源上，每个组成员上，或者是分散在成员间。

●覆盖网络的拓扑可以由应用实现完全的控制。

应用层组播的目标是建立和维护一个高效的用于组播数据分发的覆盖网络，在当前 IP 组播仍然没有在 Internet 上实现广泛部署的情况下，应用层组播服务可以代替 IP 组播，来实现群组通信以及其它具有一对多的通信模型的应用。

2.1.2 Overlay 网络的技术优势

Overlay 网络的特点决定了它的技术优势，应用层组播基于 Overlay 网络实现是因为以下这些原因：

●覆盖网的实现无需改变原有的硬、软件，具有较好的可操作性。

●在覆盖网络中部署新的功能，无需在每个节点上实施部署，只要在需要该功能的节点上部署。

●覆盖网络上的应用无需了解下层网络的拓扑结构。

●覆盖网络中的节点可以根据自己的需求选择最合适的链路，具有较好的适应性。

●可以在 Overlay 节点上部署附加的控制机制，使 Overlay 网络具有更好的健壮性。例如在两个 Overlay 节点之间完全可能存在两条互相独立的虚拟链路，如果其中之一发生错误而失效，还可以通过备用链路继续节点之间的通信。

●由于 Overlay 节点可以是完成不同任务的计算机，比较容易实现服务定制功能。

●Overlay 报文可以像 TCP/IP 报文一样处理，而无需其它特殊的处理机制。

2.1.3 Overlay 网络的抽象模型

覆盖网络的抽象模型可以表示成一个简单无向连通图 $G=\langle V,E\rangle$。其中 V 是结点的集合，表示端用户。$E=V\times V$ 是边的集合，表示任意两个端用户之间的虚拟连接。对于 $v\in V$ 和 $e\in E$ 可分别定义相应的属性函数表示网络状态。

设R^+为正实数集合,对于任意的$v \in V$，可以定义$d_{\max}(v)$属性函数。

$d_{\max}(v)=i, i \in R^+$

$d_{\max}(v)$映射到一个正整数。$d_{\max}(v)=i$表示v节点的带宽最多可以支持v节点向i个邻居转发组播数据。假定节点v的输出带宽为$B_{out}(v)$，组播流的速率为b。那么

$d_{\max}(v)=\lfloor B_{out}(v)/b \rfloor$

图 2.1 的覆盖网络可表示成：

$G=\langle V, E \rangle$

$V=\{A, B, C, D\}$

$E=\{AB, AC, AD, BC, BD, CD\}$

对于任意的$e \in E$，可定义属性函数，如 2.1.2 中定义的代价函数和延时函数。

根据 E 中的边，可以求出每个顶点的邻居。对于任意的$v \in V$

$Neighbor(v)=\{u \mid \langle v,u \rangle \in E\}$

对于图 2.2，可以求出$Neighbor(A)=\{B, C, D\}$。

2.2 基于 Overlay 网络的应用层组播路由的问题描述

因为应用层网络是建立在传统的 Internet 单播结构之上而不是点到点的链路之上的虚拟逻辑网络。这导致了在资源的配置和操作上同拥有链路的网络层组播有很大不同。主要体现在以下几个方面：

（1）网络可达性：应用层网络是全 Mesh 网，每个节点通过单播连接都能到达其它任何节点。因此，和路由器之间的路径被定义为物理连接的 IP 组播不同，在应用层网络中 n 个节点的组播会话可能有n^{n-2}个不同的生成树[8]。这就导致了更大的设计空间和更高的设计复杂度。

（2）网络开销：通常的网络开销主要被定义为图中所有链路开销的总和。

这对于网络提供者来说是合理的，但是从应用服务提供者角度看，应用层网络开销包括从服务站点获得带宽的开销和从主干网上获得访问带宽的开销。网络开销定义的不同直接影响路由的设计和路由的策略。

（3）路由限制：传统的 IP 组播通过最短路径树最小化源到全部接收者的时延，也就是减少需要承载会话流的链路数。应用层组播能够根据应用需求更灵活的配置不同的路由策略，提供更高质量的服务。

2.2.1 应用需要的组播服务质量的抽象描述

在应用层组播传输过程中,端系统需要复制、转发数据包，直到所有结点都收到数据包为止。由于这些结点通常是普通的主机，不能进行线速转发，所以复制转发工作消耗的时间比较大。消耗的时间与主机本身的处理能力和执行的任务数量有关。如果某个主机结点的下行用户数目太多(即结点度大)，那么它的工作负载会相当重，这个主机处理消耗的时间会较长，或者部分任务得不到处理。而且树的时延还受结点度的影响。结点度越小，树的深度越大，树的时延也越大。所以树的时延和结点度分配是相互矛盾的问题。鉴于此，应用层组播路由中要考虑节点度和时延的平衡问题。

对于不同类型的组播服务有不同的需求。例如对于文件传送应用，需要高吞吐率，而不太注重延时性能。对于音视频相关的应用，有实时性的要求，就要对组播树的延时进行优化。将应用的这些需求抽象成约束条件和优化目标。约束条件是指组播树必须满足的条件，优化目标是指组播树可利用各种算法，如贪婪算法，禁忌算法等尽可能找到满足约束条件的最优解，因为在约束条件下，可能找不到多项式时间的最优解。

定义 1.树的直径(Diamete)：对于树 T 中任意节点 $\forall v \in V$，$\forall r \in V$，$r \neq v$，$P(r,v)$为从 r 到 v 沿 T 上的路径的所有边的集合。$\delta(v)=\sum_{r \in P(r,s)} c(e)$为树 T 上经过 v 的最长的距离，则树的直径定义为：$Diamete:=\max \delta(v)\ \ (v \in V)$。

定义 2.剩余度(Residual degree)：对于树 T 中节点 $\forall v \in V$，$Res_T(v)=d_{\max}(v)-d_T(v)$，$d_T(v)$是树 T 中节点 v 的度。由于每个终端节点可能不只加入一个组播会话，如果某些组播会话过多地占用了那个节点的接口带宽，

那么有可能出现后续组播会话无法建立的现象。为了减少阻止将来组播会话需求的可能性，我们选择最大化树的最小剩余度

约束条件通常包括：

节点的度约束条件：在应用层组播中担任转发节点的是端用户，端用户的网络资源是有限的，因此端用户担任转发的负载应该是有限的。节点的虚拟连接的数目(节点的邻居数目)的限制，称为节点的度约束条件。节点的度约束条件用节点属性函数$d_{\max}(v)$表示。

树的约束条件是指对组播树的直径、半径进行约束。限制树上节点之间的最长路径。也可对全树的代价进行约束。

优化目标可以分为节点优化目标和组播树的优化目标，通常包括：

节点的最大带宽：在节点支持的范围内最大利用每个节点的带宽。

节点的最大平衡剩余度[14]：每个节点的剩余度尽可能达到平衡，这样可提供较高的接纳率[15]。

树的优化目标包括最短路径树，最小代价树等。

2.2.2 组播路由问题定义

假设下层 IP 网络能向端系统提供透明的端到端单播路出，则覆盖组播的网络模型可用一个完全有向图$G=(V,E)$来描述，其中，V是节点的集合，表示端系统，$E=V\times V$是边的集合，表示虚拟链路，每一虚拟链路对应于一条下层 IP 网络的物理路径。设R^+为正实数集合，R^*为非负实数集合，则$\forall v\in V$，可定义下列属性参数：输入带宽能力函数$B_{in}(v)$：$V\to R^+$，表示节点最大的接收带宽；输出带宽能力函数$B_{out}(v)$：$V\to R^+$，表示节点最大的转发带宽；代价函数$C_V(v)$：$V\to R^+$，表示节点的代价，如端系统设备的费用等。$\forall e\in E$，可定义如下属性参数：带宽容量函数$B_c(e)$：$E\to R^+$，表示边上两节点之间可获得的最大带宽，它由该边上的端系统节点的输入、输出带宽能力及其对应的下层物理路径的瓶颈带宽决定，即$\forall e\in(u,v)$，有$B_c(e)=\min(B_{out}(v),bn(u,v),B_{in}(v))$，其中

$bn(u,v)$为边(u,v)对应的物理路径的瓶颈带宽；延时函数 $D_e(e)$: $E \to R^+$，表示边上两节点之间的端到端分组传输延时；抖动函数$J_e(e)$: $E \to R^+$，表示边上两节点之间的端到端分组传输延时抖动；分组丢失率函数$P_e(e)$: $E \to R^+$，$P_e(e) \in [0,1)$，表示边上两节点之间的端到端分组传输丢失率；代价函数 $C_e(e)$: $E \to R^+$，表示边的代价，如网络资源占用量等。

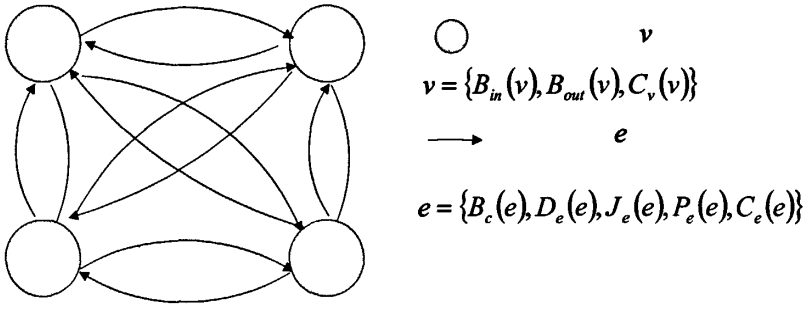

图 2.3 覆盖组播网络模型示意图

图 2.3 给出了覆盖组播网络模型的示意图，该网络模型有 4 个端系统节点组成，它们之间通过 12 条有向边（虚拟链路）构成一完全有向图，对于每个端系统节点和每条虚拟链路分别定义相应的属性参数。与 IP 组播网络模型不同[9]，覆盖组播网络模型中的边是虚拟链路，它可能与其他边共享端系统节点和下层物理链路的带宽，这将增加覆盖组播网络中带宽资源管理的复杂性。但是，考虑到端系统的接入带宽和 CPU 处理能力的限制，当端系统成为虚拟链路的带宽瓶颈时，$B_e(e)$将由端系统的带宽能力属性决定，这将简化上述模型。进一步，对于媒体流的平均带宽为b的组播组，若$\forall v \in V, b \le B_{in}(v)$，则节点$v$的输出能力带宽可用“度”来表示，定义度约束属性$d_{\max}(v)=\lfloor B_{out}(v)/b \rfloor$，即节点$v$最多只能同时向$d_{\max}(v)$个下游节点转发分组。此时，覆盖组播网络就可用度约束模型来描述。但是，在上述简化条件不成立时，组播应用通常采用端到端的测量来估计虚拟链路的可用带宽。因此，本章下面将从端系统的角度来讨论和描述

带宽参数。

覆盖组播路由问题的一般形式描述如下：给定一个完全有向图$G=\langle V,E\rangle$，一个源（或根）节点$s\in V$，以及目的节点的集合$V-\{s\}$。对于一组约束条件C和一个可能优化目标O，构造一棵G的最优生成树，使其满足C。

根据上述模型，我们发现覆盖组播路由问题的优化目标和约束条件均可分为二类：节点优化和树优化；节点约束和树约束。其中，节点优化（约束）针对节点相关的属性参数，如节点的带宽或度、节点的代价等；树优化（约束）针对与组播树相关的性能参数，如树的延时直径（或半径）、树的平均延时、树的代价等。将这些约束条件和优化目标进行组合，可形成下面各种覆盖组播路由问题：

（1）节点约束：具有节点属性约束的组播路由问题，如：度（带宽）约束的组播树等。

（2）树约束：具有树性能参数约束的组播路由问题，如：延时约束的组播树等。

（3）多树约束：具有多个树约束的组播路由问题，如延时和代价约束的组播树等。

（4）节点和树约束：具有节点和树约束的组播路由问题，如度（带宽）和延时约束组播树等。

（5）节点优化：节点属性优化的组播路由问题，如：最小（或平均）带宽最大组播树等。

（6）树约束节点优化：具有树约束且节点属性优化的组播路由问题，如：延时约束的剩余度（带宽）平衡组播树等。

（7）树优化：树性能参数优化的组播路由问题，如：延时（或代价）最小组播树等。

（8）节点约束树优化：具有节点约束且树优化的组播路由问题，如：度（带宽）约束的延时（或代价）最小组播树等。

（9）树约束树优化：具有树约束且树优化的组播路由问题，如：延时约束的代价最小组播树等。

（10）节点和树约束树优化：具有节点和树约束且树优化的组播路由问题，

如度（带宽）和延时约束的代价最小组播树等。

表 2.1 覆盖组播路由问题分类表

	无优化	节点优化	树优化
无约束	——	(5)节点优化 “NP 完全”复杂度	(7)树优化 多项式复杂度
节点约束	(1)节点约束 多项式复杂度	——	(8)节点约束树优化 “NP 完全”复杂度
树约束	(2)树约束 多项式复杂度 (3)多树约束 “NP 完全”复杂度	(6)树约束节点优化 “NP 完全”复杂度	(9)树约束树优化 “NP 完全”复杂度
节点和树约束	(4)节点和树约束 “NP 完全”复杂度	——	(10)节点和树约束树优化 “NP 完全”复杂度

问题（1）容易在多项式时间内解决，如对于度约束组播树的构造，可从源节点开始，每次选择任一未成树的节点加入到树上任意有剩余度的节点，直到所有节点都加入树为止。问题（7）可分为延时半径（源到最远节点的延时）最小树、延时直径（最远两节点之间延时）最小树、代价最小树等问题。不难发现，从源节点到所有目的节点构成的星型（Star）树，其延时半径最小；Hassin 等[10]则提出了在多项式时间内寻找延时直径最小树的算法；而代价最小生成树可以采用 Prim 算法解决。另外，问题（7）的算法可用于解决问题（2）。所以，问题（2）和（7）都是多项式问题。根据 Wang 等的结论[11]，问题（3）属于多个“相加型”约束组合问题，是“NP 完全”的[12]。问题（9）中的延时约束的代价最小生成树问题也是“NP 难”（NP-Hard）问题[13]。Garg 等[14]证明了问题（5）中的最小带宽最大树的构造是“NP 难”的，则平均带宽最大树也是“NP 难”的。现已证明，度约束的延时（平均延时、半径、直径）最小以及代价最小生成树问题都是“NP 难”[15]或“NP 完全”的[16][17][18]。Shi 等[16]还证明了延时约束度平衡生成树问题是“NP 完全”问题。由此易知，问题（4）、（6）、（8）、

（10）都具有“NP 完全”复杂度。覆盖组播路由问题的分类汇总，如表 2.1 所示。

在上述问题中，问题（1）过于简单，没有单独研究的价值。问题（2）、（3）、（7）、（9）与节点的属性无关，可以归结为 IP 组播路由问题的特例，不是覆盖组播路由研究的重点。因此，当前覆盖组播路由协议和算法重点解决的问题在于（4）、（5）、（6）、（8）、（10）。

2.3 现有的应用层组播方案

图 2.4 给出了目前主要的应用层组播方案分类及每种分类的代表[19]。方案主要分为集中式算法和分布式算法两大类，其中每类又包含几种类型。

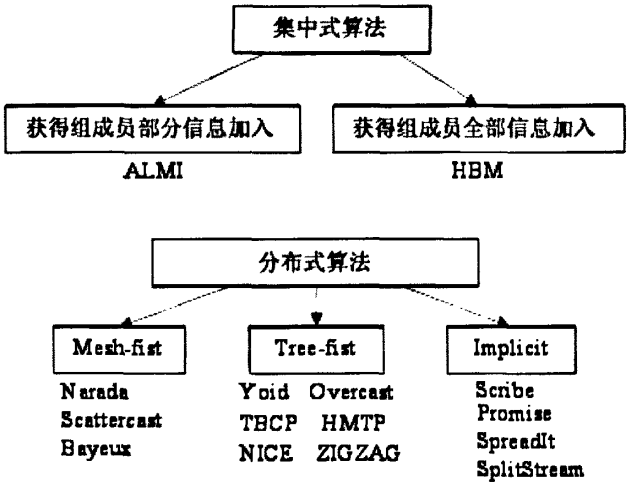

图 2.4 当前的应用层组播方案

集中式系统中通常由一个节点来集中控制和处理节点的加入、离开和失效，因此效率较高，但同时也限制了系统的可扩展性，现有的系统有：ALMI 和 HBM；分布式系统可以接纳较多用户，具有较好的可扩展性，但是对节点的加入、离开和失效的处理效率较低。

分布式系统按组播树构建的顺序，又可以分为树优先(Tree-first)，转发网优先(Mesh-first)和隐式构建组播树(Implicit)三种。应用层组播协议中一般定义两种拓扑结构，控制拓扑和数据转发拓扑。组成员通过控制拓扑传递和更新信息，互相之间辨别是否仍然“活跃”，还是已经失效或离开，控制拓扑可能存在回路，又称为转发网(Mesh)；而数据转发拓扑通常是控制拓扑的一个子集，组播数据

沿转发拓扑传递，因此它不能存在回路，也称为组播树(Tree)。树优先方式先生成组播树，再生成转发网，适合对延时敏感的应用，如实时应用，现有的系统如 Yoid、Overcast、TBCP、HMTP、NICE、ZIGZAG 等；转发网优先方式先构造一个有冗余链路的转发网，再使用如 DVMRP 这样的路由协议在转发网的基础上构建组播树。这种方式可以检测到组播树节点的失效和组播树的断裂，并进行高效的恢复和处理，适用于底层网络性能不好或可靠性较差的情况。由于它的可扩展性较差，适用于规模较小的组，比较著名的有 Narada、Scattercast、Bayeux 等；隐式方式是基于某种特性隐式地构建控制拓扑，同时构建转发网和树，两种拓扑的转化不需要额外的成员之间的交互，适合规模较大的组播组，现有的系统如 Scribe、Promise、SpreadIt、SplitStream 等。

2.3.1 集中式算法

集中式算法如 ALMI[20]、HBM[21]。集中式算法中存在一个总控模块，总控模块是一个独立的运行实体，总控模块需要被所有的成员访问。总控模块可以放在一个组成员（通常是组播组的发起者）所在的主机，也可以放在一个专门的服务器上，或是向 ISP 租用的组播代理服务器上。如 ALMI 中的会话控制器 session controller[20]，HBM 中的 RP[21]模块。

集中式算法内又细分为两种，差别是成员节点对应用层组播的组播组成员关系了解的程度，分为部分了解和完全了解两种。

1.集中式算法的思想是：

1)总控模块与组成员之间、组成员之间通过单点投递路径交换控制信息。

2)总控模块负责处理成员加入、根据搜集的控制信息计算组播树和维护组播树。

3)总控模块向每个组成员发出指示，需要监控与哪些其它组成员之间的“距离”参数，“距离”是组成员之间的单点投递路径的度量参数，在 ALMI 和 HBM 中“距离”是组成员之间的延时。

4)组成员根据总控模块的指示，实行监测，周期性的向总控模块报告。

5)总控模块根据所有成员的监测信息，根据组播树算法计算组播树。在 ALMI 中，采用了度约束的最小代价树；在 HBM 中，采用将低带宽、不稳定的

节点逐渐推向组播树的叶的算法。

6)总控模块将计算的结果中与每个成员相关的部分告诉每个成员，计算结果通常是$\langle parent,child\rangle$对，这样每个成员可得知其在组播树上的父节点和子节点。

7)成员通过组播树负责对组播数据进行转发，不需要总控模块干预。

8)总控模块负责在新成员加入和成员离开、网络失效、成员失效时，维护组播树的连通。

2.集中式算法的优势

可以提供较好的可靠性，减少组成员处理组成员管理和维护组播树的开销。总控模块与组成员之间只使用单点投递路径交换控制信息，不会造成较大的数据流量。

3.集中式算法的缺陷

在于扩展性较差，适用于组播组的组成员数目较小的情况。并且因为依赖于总控模块计算组播树、维护、处理新成员加入，因此总控模块容易成为系统的瓶颈，也会造成单点失效的情况。

ALMI 属于节点约束下的树优化算法，集中式算法还有 CT[22]算法－度约束的最小直径生成树（MDDL[22]），CT 同属于节点约束下的树优化。BCT[23]算法－直径约束的剩余度平衡生成树（LDRB[22]），BCT 属于树约束下的节点优化。

2.3.2 分布式算法

分布式算法大致可以分为三类，分别是 Mesh[24]优先，Tree[24]优先和隐含构造[24]。

2.3.2.1 Mesh 优先算法

1.Mesh 优先算法思想是：

1)新成员加入组播组时，将其加入应用层组播的覆盖网络，并没有加入到组播树。

2)组成员共同协作，在新成员加入和成员离开、网络失效、成员失效时，维护应用层组播的覆盖网络的连通性。

3)每个成员本地只保留有关覆盖网络的状态信息，通常是与邻居节点之间的路径状态参数，如延时参数。没有类似于$\langle parent, child \rangle$这样的关于组播树的信息。

4)当一个成员收到组播数据包时，它根据本地的邻居状态信息、组播数据的源、按照某种转发策略来确定向哪些邻居转发组播数据包。

Mesh 优先算法没有专门构造组播树的过程，每个成员本地也不维护组播树状态。只是在组播数据到来时，按照约定的策略确定向哪些邻居转发。假定在 Mesh 优先中，A 向 B 转发，A 就是 B 的父亲节点。那么从数据源开始，也可得到一棵组播树。这是“数据驱动”的树，当数据到来时，就可建立树，当没有数据时，树也就消释了。

Narada[25]和 Scattercast[26]按照“逆向路径转发”来确定数据包的转发路线的。“逆向路径转发”思想：当一个成员 A 收到邻居 B 发来的数据源是 S 的组播数据包时，如果 B 是 A 到 S 的最短路径上的下一跳，那么 A 就向它的邻居中以 A 作为到 S 的下一跳的转发。Delaunay triangulation[27]按照“三角路由”来确定数据包的转发路线。

2.Mesh 优先的主要不足

由于通过在 Mesh 中加入转发策略的方式来建立组播树，因此对组播树的结构很多方面都不能直接控制，例如无法控制树上节点的度，无法控制节点在树上的关系等。因此，组播树的性能好坏只能依赖于建立的 Mesh 的性能。

2.3.2.2 Tree 优先算法

Tree 优先算法与 Mesh 优先不同，当成员加入时，主要目标是为成员找到组播树上一个合适的节点作为它在树上的父节点，将成员加入到组播树中去，让它可以实现组播数据的发送与接收。之后再选择树上除父亲节点以外的其它一些节点，建立覆盖网络上的连接关系，用于当成员离开、网络或节点失效，或是树性能优化时维护组播树。

Tree 优先的算法通常为组播组建立一棵共享树，树上的每个节点都可以进行度约束，限制在树上孩子的数目。

Banerjee 算法[28]、HMTP[29]算法、Switch-tree[30]算法、Yoid[31]都属于 Tree

优先算法，Tree 优先算法中都包含一种递归思想。算法中包含新成员加入算法、树的维护及性能优化算法等部分。

1.新成员加入算法

Banerjee 算法中新成员加入时：

1)新成员 n 向树的根节点发送加入请求。

2)节点 p 收到新成员 n 的加入请求后，可以有三种可选的操作：

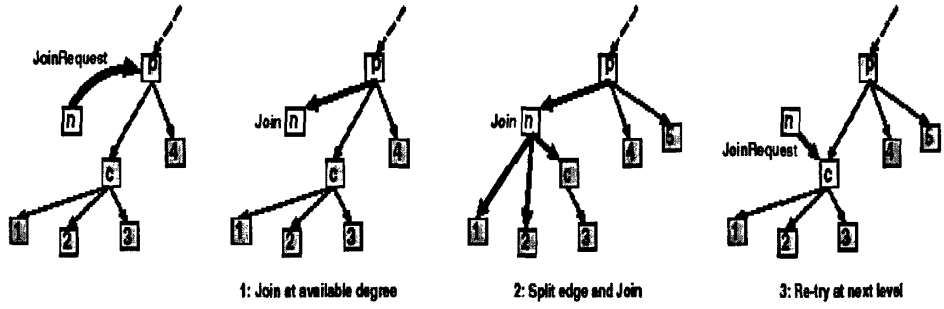

图 2.5 BANERIJEE 加入算法

●如果本身“度”不满，就接纳新成员作为子节点。

●n 选择 p 的一个孩子 c，分裂 p 与 c 之间的父子关系。将 n 变变为 n 的子节点。

●n 选择 p 的一个孩子 c，向 c 重新发出加入请求。(如图 2.5)

HMTP 算法中新成员加入时，首先向根请求，在根的孩子中选择一个最“合适”的节点，再向下重复这个请求过程，直到找到一个合适的父节点。

Switch-tree 算法新成员加入时，全部连接到根节点。

2.树的性能优化算法

性能优化算法简而言之，就是将树上的节点位置进行调整，实现树的优化目标。在 Banerjee 算法中，树的优化目标是最小化树上节点的最大延迟。在 Switch-tree 算法中，树的优化目标是最小化树的代价或是最小化延迟。

Banerjee 算法和 Switch-tree 算法中分别定义了几种节点位置调整策略。

3.树的维护算法

树的维护算法用于树上节点之间状态维护，以及处理节点离开和失效。当节点离开或是失效时，该节点的子树将与组播树断开，需要将该节点的子树重新连接到组播树上。离开与失效的区别在于，节点离开时，可通知其它节点它

的离开。而失效时，需要由其它节点检测到。

Tree 优先的算法主要优势在于可对树的结构实现直接控制，可以根据不同的应用需求确立组播树的约束条件与优化目标，建立满足应用需求的组播树。

在 Tree 优先算法树算法中，Switch-tree、HMTP、Banerjee、Hostcast[32]都是节点约束下的树优化算法。Malouch[33]算法是节点约束树约束下的树优化算法。

2.3.2.3 不显式构造算法

不显式构造算法是指没有显式的 Mesh 优先或是 Tree 优先过程的算法。这一类算法可支持大规模的组播组。这一类算法大致可以分为两类：一类是集群算法，另一类是基于 Peer-to-Peer 覆盖网络的应用层组播算法。集群算法的思想是将组播组的成员构成一个层次式的结构。集群算法包括 NICE[34]、Zigzag[35]等。

1.NICE 算法思想：

(1).所有的成员都属于最低层 L0，

(2).在每一层将成员按照集群策略组成若干个族，每个成员只能属于一个族，再为每个族确定一个族的头领

(3).每个族的头领组成了较高一层。然后再重复步骤 2，直到最高层，最高层中只有一个成员

(4).对于任意一个成员 m，它属于 L0，L1，...Li，Li 是 m 属于的层中最高的层数。那么 m 一定是它所属各层的族的头领。

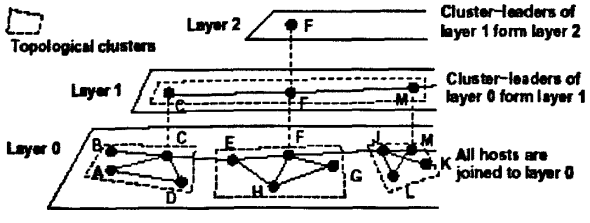

图 2.6 NICE 的层次式结构

如图 2.6 所示，A、B、C、D、E、F、G、J、K、L、M 是覆盖网络的成员。在 L0 层，A、B、C、D 组成一个族，E、F、G、H 组成一个族，J、K、L、M

组成一个族。三个族的头领，C、F、M 组成了 L1 层的一个族，族的头领 F 成了 L2 层的唯一一个族的成员。

(5).当一个成员 m 收到一个组播数据包 p 时，按照组播数据转发算法进行转发。

MulticastDataForward(m，p){//m 是组成员，p 是数据包

假定 m 属于层 L0，L1，。。。Li，在每层中分别属于族 CL0，CL1，。。。CLi

for j in[L0..Li]

if p 不是来自于族 CLj

那么向族内所有成员转发

}

如图 2.7，给出了一个 NICE 的拓扑和数据转发路径。左上为 NICE 的拓扑图，之后的三幅分别给出了当以 A0，A7，C0 为源时的数据转发路径。

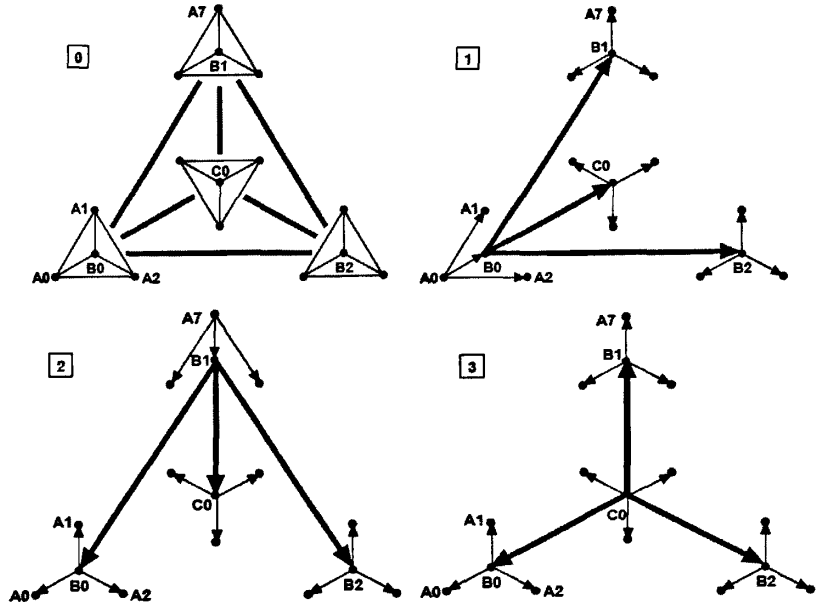

图 2.7 NICE 的组播数据转发

集群算法的优点在于支持大规模，并且树的高度可控制在 $\log_k^n$。其不足在于对树的结构缺乏控制，不易实现树的性能优化。

2.基于 Peer-to-Peer 覆盖网络的应用层组播方案

目前在 Peer-to-Peer 网络上实现的应用层组播方案主要有三种：CAN

Multicast[36]、Scribe[37]、Bayeux[38]。它们都是在基于动态哈希路由的 Peer-to-Peer 网络上实现的，其中 CAN Multicast 是在 CAN 之上实现的，Scribe 是在 Pastry 上实现的，Bayeux 是在 Tapestry 上实现的。

这几种方案都充分利用了 Peer-to-Peer 网络的路由机制,因此只需增加少量的模块就可以实现组播功能。与原先的 Peer-to-Peer 网络相比，只增加少量的开销就实现了组播功能，同时继承了 Peer-to-Peer 网络的支持大规模、支持成员动态变化的特性。可用于分布式仿真、多方实时游戏、大规模协作应用等。Peer-to-Peer 网络可支持两种模式的应用层组播，服务接入点模式和端用户模式。Peer-to-Peer 网络屏蔽了物理网络的特性，可支持应用层组播与 IP 组播的兼容、混合；IPV4 与 IPV6 的兼容、混合。

Peer-to-Peer 覆盖网络上的应用层组播研究还处于探索阶段，目前的三种方案只是利用了 Peer-to-Peer 路由实现了组播路径。对于应用层组播的模型、性能分析、性能优化都没有进行研究。

2.4 应用层组播体系结构（ALMSA）

2.4.1 应用层组播的系统结构

目前主要有 3 种系统结构来实现应用层组播：

(1)对等型

每个组播组成员节点都是平等的，动态变化且完全分布。节点之间通过一定的算法、协议自组织成控制网络和数据转发树。每个节点仅维护自身参与组的状态信息，所有组播相关功能以软件实体形态集成于参与组播会话的节点中，每个节点完成相同的工作。如 ESM(End System Multicast)[25]。

(2) 代理型

这是一种基于固定节点配置的覆盖式组播技术，一般由增值服务提供商根据一定策略在 Internet 的某些位置部署应用层代理节点。代理节点之间的数据传输路径和传输方式也由增值服务商预先确定。终端主机通过接入距自身最近的代理节点获取数据。从某种程度上来看，代理节点类似于组播路由器。

(3)服务器型

介于对等型和代理型之间。转发树的主干由一些负载较大的服务器构成，不同于代理类型，这些服务器不一定来源于 ISP，可能是作为普通用户加入的、性能较高的网络终端主机。在服务器型和代理型应用层组播中存在一些节点，其性能相对较高，所以基于这些节点构建的转发树也比较稳定，可支持规模相对较大的应用层组播服务。

2.4.2 应用层组播系统的协议栈模型

对于应用层网络而言，其是一个定义主机之间通信的寻址方式、路由方式和服务模型，位于现有的 Internet 传输网络之上的一个完全位于应用层的网络系统。在其中，拓扑发现，路由转发等功能完全由应用层自己完成，不依赖网络层。我们可以把它看作是基于 Internet 网络的大规模的分布式应用。

应用层组播网的节点是组播成员主机，数据路由、复制、转发功能都由成员主机完成。成员主机之间建立一个叠加在 IP 网络之上的，实现组播业务逻辑的功能性网络，称为覆盖网（overlay network），主机给予自组织算法建立和维护叠加网。为了实现组播服务，需要在主机上部署组管理，组成员管理，应用层组播路由协议等功能模块。图 2.8 给出了应用层组播的示意图。

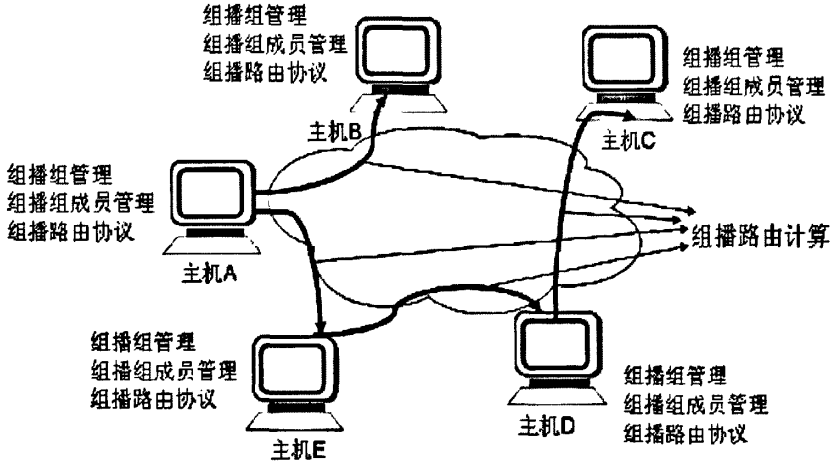

图 2.8 应用层组播

图 2.8 中，每个主机都部署了组播组管理、组播组成员管理、组播路由协议等功能模块，实现主机的组播路由和组播管理功能。图中的主机构成一个组播组，主机 A 通过组播路由计算，将自己的数据报文单播到主机 B 和主机 C。主机 C 再通过组播路由计算将自己收到的源为 A 的数据报文单播到主机 D。主

机D通过组播路由计算将自己收到的C发来的源为A的数据报文单播到主机E。通过这一过程，每个成员都收到了源为 A 的组播数据报文。

自组织算法[46]是端系统组网的核心功能和机制，自组织算法的主要功能包括：周期性地交换节点状态信息，通报组成员状态；周期性地收集网络逻辑连接的带宽、时延等动态参数；动态地调整叠加网拓扑。

直观上，端系统实现组播功能可以避开网络层实现组播功能的许多难题：一是应用层组播的状态在主机系统中维护，不需要路由器保持组的状态，解决了业务的扩展性问题，网络可以支持大量的组播组。二是组播应用可以随时部署，不需要网络设备的升级和功能扩展。三是可以简化组播的控制、可靠等功能的实现，建立在网络连接之上的应用层组播可以使用 TCP、UDP 服务，如可以利用 TCP 的可靠和拥塞控制简化组播的可靠和拥塞控制。

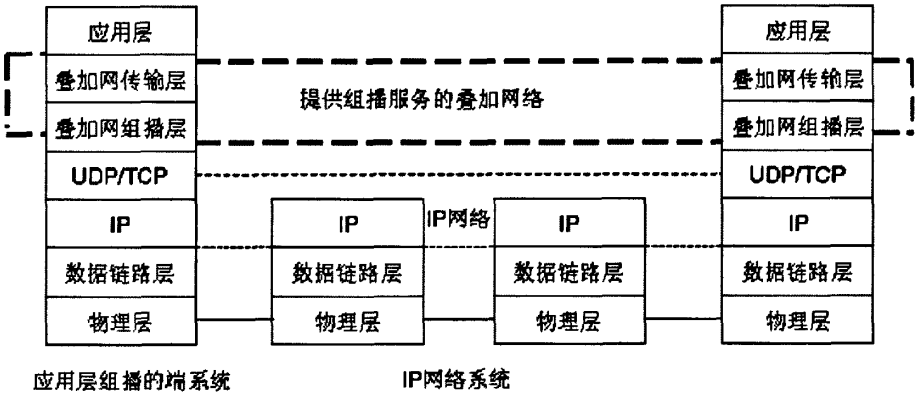

图 2.9 应用层组播体系结构

图 2.9[39]显示了应用层组播在 IP 体系结构中所处的位置。在应用层组播可以根据需要提供的服务不同可以选择是在边界路由器平台上直接实现附加功能还是通过转发数据流到服务器上来实现。例如，差异服务（DiffServ）和网络安全可以在接入路由器上实现,因为这样的功能适合所有的数据流并且也只需要少量的附加状态；但是对于内容分发（content distribution）和网络存储这样的服务则更可能用到的是服务器，因为这些服务器是针对一部分数据流而且需要更多的管理资源。

2.5 本章小结

本章我们首先介绍了目前实现应用层组播的关键技术-覆盖网络的定义，技术优势和抽象模型，以及基于覆盖网络的应用组播服务质量的描述和路由问题定义。接着在对现有的应用层组播方案进行分析，一般来说有集中式算法和分布式算法两大类，其中分布式算法按照组播树的构建方案又分为树优先(Tree-first)，转发网优先(Mesh-first)和隐式构建组播树(Implicit)三种。最后在分析目前常用的三种应用层组播系统结构的基础上提出较为普遍的应用层组播系统协议栈模型。

第三章 一种新的应用层组播协议 HFTM

当前基于 tree 型覆盖网络拓扑的应用层组播协议在组播性能方面并不尽人意，本章提出一种基于 tree 覆盖网络拓扑的可扩展高效应用层组播协议—HFTM。HFTM 通过分层和分群的思想将所有组播组成员构造成一个特殊的层次化结构，在进行群划分时，充分考虑了底层网络拓扑特征，尽量避免数据包在代价昂贵的链路上进行传输，从而减少组播延迟。另外，采用一种新颖的基于斐波那契序列的组播算法将群内成员构造成一棵高效的斐波那契组播树，利用此树进行群内组播。HFTM 所设计的动态组管理策略可以在组成员发生变化的时候，快速恢复正常的组播通讯。实验结果表明底层网络拓扑特点的考虑以及斐波那契组播树的构造使 HFTM 协议获得更好的组播延迟性能。

3.1 协议整体描述

在应用层组播中，常用的两种覆盖网络拓扑结构是 tree 和 m-D mesh，本章提出的新颖协议是基于 tree 的覆盖网络拓扑结构。应用层组播的提出是为了在域间网络实现组播功能，但由于应用层组播自身的固有特点，存在一些性能上的损失：延迟上的损失和可扩展性上的损失[39]，应用层组播要想大规模投入使用，必须把这些性能损失问题解决好。因为树型的拓扑结构扩展性比较好[2][34]，所以我们主要考虑如何改善延迟性能上的损失，然而，所提的新协议也兼顾了对可扩展性的改善，具有较好的可扩展性。

一般来说，延迟可以分为平均延迟和最大延迟，因为每个组成员都希望以较短的延迟收到数据包，所以我们在协议设计的时候主要侧重于减小平均组播延迟。为了获得较小的平均延迟，我们充分利用了之前很多协议都忽略的底层网络特征，虽然应用层组播的执行是独立于底层的，但可以通过利用底层网络特征来获得更好的性能，因为底层网络是覆盖网络的基础，而且数据传输最终还是要通过底层网络进行。在我们的设计中，引入了新的概念：local area 和 backbone area(具体定义会在后面章节介绍)，将组播组所在的网络划分成不同的 local area 和 backbone area，为每个 local area 选择一个 local core。这种机制由于考虑了主机的实际位置，从而减少了数据在 backbone area 的链路上的传输机会，

又因为 backbone area 的链路都是比较耗时的，所以此举减少了主机之间数据传输的延迟。

本章提出了一种高效可扩展的应用层组播协议—HFTM(Hierarchical Fibonacci Tree Multicast)，在 HFTM 中，充分利用底层网络特征进行群划分，尽量减少在代价昂贵的链路上进行数据传输的机会，从而达到缩减延迟、减少维护代价的目的。其次，HFTM 给出了一种动态的群领导(cluster leader)选择机制，通过发送探测 probe 消息来为每个群选择一个领导者，这样做的好处是可以反映实时的网络形势，从而使选出的领导更合适、更准确，继而使群内组播路由效率更高。另外，协议通过 local area 概念的引入，在每个 local area 内进行群划分，尽量减少将不同 local area 的成员划分到同一个 cluster 内，避免了在不同 local area 之间进行频繁的数据传输。

在路由方面，HFTM 给出了一个新颖有效的混和路由算法，在群间采用层次化的路由机制，而在群内，则采用基于斐波那契序列的组播算法，构造了一棵斐波那契组播树进行群内组播。另外 HFTM 采用一种动态的组维护机制，能以较低的代价来处理成员加入或离开时组播通讯的维护和恢复。

本章是如下组织的：首先给出关于 HFTM 的简单介绍，然后给出 HFTM 的具体设计细节，包括组播结构的构造、组播路由机制的设计和动态组管理机制。最后是本章小结。

3.2 构建应用层覆盖网络

3.2.1 基本结构

新协议 HFTM 采用了分层和分群的层次化结构，在 NICE[34]中已经证明此结构有助于提高应用层组播的可扩展性和效率。但 NICE 协议在划分群的过程中没有考虑主机在底层网络的拓扑结构，因此造成的一些不合适的划分导致了资源的浪费。和 NICE 协议不同，我们的协议在采用分层分群思想的基础上，考虑了主机的实际位置，充分利用了底层网络特征来进行组成员的群划分，避免了 NICE 存在的上述问题。为了利用底层网络特征，本协议给出如下两个新的概念：

(l)本地区域(local area)：由直接连到同一个路由器的主机或通过几个本地网络部件(比如 hub)连接起来的主机以及连接它们的本地网络资源(如物理链路)所组成的区域。

(2)骨干区域(backbone area)：由连接组成员的路由器以及路由器之间的其他网络资源(比如物理链路)所组成的区域。

根据上述定义，组播成员所在的网络可以被划分为不同的 local area 和 backbone area，这样做的好处将在后面章节进行具体描述。

在我们的设计中，假定存在一套聚集点集合 $RPS=\{RP_0,RP_1,\cdots,RP_{R-1}\}$，其中 R 是此集合中的聚集点数目，当新的成员要加入某个组播组时，它寻找一个“最近”的聚集点 RPr，然后向 RPr 进行注册，此处的“最近”是指聚集点到要加入主机的单播距离而言。注册完毕之后，RPr 要为这个新的主机保存一条记录，通过这样的注册过程，每个聚集点都保存一个向它注册过的成员列表，而这个成员列表在构建和维护 HFTM 结构方面能起到很重要的作用，这一点将在后面章节详细介绍。

3.2.2 Cluster 的构造

对整个组播组的划分首先是以 local area 为单位进行的，首先根据 local area 的定义将整个组成员划分为不同的 local area，然后对每个 local area 内的组播组成员进行层(layer)和群(cluster)的划分。

为了构造一个 cluster，首先需要在 RPS 中选择一个合适 RP_r，RP_r 被选出之后，就要从组中随机选择一个主机，这个主机被称作 FM(first member)节点，然后 RP_r 将本地成员列表发送给 FM 节点，FM 从成员列表里选择与自己距离最近的主机划分到自己的 cluster 里，此处的“距离最近”是根据主机的 IP 地址进行衡量的，一般认为，两台主机的 IP 地址的区别越小，它们的距离就越近。这样的选择过程持续进行，直到 cluster 的大小达到上限为止。其中，cluster 的大小 s_c。满足 $s_c=(k, 3k-1)$的表达式，其中$(k, 3k-1)$表示大小介于 k 和 3k-1 之间的一个随机数。和 NICE 一样的是，在实验中，k 的取值是一个随机的常数，但不同的一点是，我们的取值比 NICE 中大，取值为 k=6，主要目的是为了构造斐波那契组播树的方便，具体原因会在后面章节详细分析。按照上述的 cluster 尺寸

将组播成员划分到不同的 cluster 中，当 local area 中还未被划分的主机数目少于 k，则将它们划分到一个 cluster 内，而不与其他 local area 中的成员混到一起。这样保证了初步划分的时候，只有同一个 local area 的成员能划到一个 cluster 内，从而尽量减少了不同 local area 中成员之间数据交流机会，此处只包含同一个 local area 内成员的 cluster 被称为 intra-cluster。

采用(k，3k-1)的大小进行 cluster 划分是比较合理的，可以避免 cluster 的频繁分裂和合并。如果上限选择 2k-1 的话，当超过上限即达到 2k 之后，就分裂为 2 个大小为 k 的 cluster，这样，一旦有一个成员离开，就要引起和其他 cluster 的合并。而如果上限选作大于 3k-1 的常数，cluster leader 要负责管理较多的组成员，因此容易成为瓶颈。

所有的组播成员首先都位于最低层 L_1 上，按照上述群划分方法，首先在最低层 L_1 进行 cluster 的划分，每个 cluster 拥有一个 cluster leader，然后 L_1 层上所有的 cluster leader 构成了 L_2 层，继续采用上述相同的群划分方法来对 L_2 层上的成员进行群划分，L_2 层上所有的 cluster leader 构成了更高一层 L_3，依此类推，直到最后一层上只剩一个节点。

对于每个 local area，最后都要选择一个 local core 来作为一个 local area 的代表，也是每个 local area 中最高层上唯一出现的成员，然后，所有的 local core 继续构成后面的 layer 和 cluster。因为这样的 cluster 会包含位于不同 local area 中的主机，因为我们称它们为 inter-cluster。

我们的 HFTM 协议采用一种动态的选择算法来为每个 cluster 选择 leader。Cluster leader 负责将内部主机要发送的数据包发送给本 cluster 之外的组成员，并且将外部组成员发来的数据包在本 cluster 内部进行转发。由于 cluster leader 的地位很重要，因此 leader 的选择也至关重要，它会影响到包组播的延迟。因此提出一种基于当前网络状态的动态选择算法，因为我们的目的是获得较短的组播延迟，所以所选的 cluster leader 应该满足到其他 cluster 成员的单播延迟最小。

在前人的工作中，leader 选择一般采用静态的网络状态做参数(如链路跳数)，因为不能反映出当前的网络形势，所以会造成一定的误差。在我们采用的基于当前网络形势的动态算法中，采用探测机制[40][41][42]，通过发送 probe 消息

来选择一个到其他成员单播延迟最小的节点。在 HFTM 中，发送的 probe 消息通过携带发送和接收时间戳来收集 overlay 网络上逻辑链路的延迟特征—RTT(round trip time)。

一般来说，发送 probe 消息的方式有两种：按需发送和阶段性地发送。按需发送方式是当有请求发来的时候再发送 probe 消息，这样做的好处是由发送消息而引入网络的开支比较小，但是 leader 选择的延迟可能比较大。阶段性地发送方式则相反，它不管外界如何，总是阶段性地发送消息来探测网络形势，因此它得到的信息是最新的，一旦有要求选择一个新 leader，它可以迅速选出一个合适的主机，因此，和按需发送的方式相比，它的 leader 选择的延迟减少了，但它也存在缺点，即会引入太多的开销。在我们的协议中，考虑到希望探测代价较小的要求，采用了按需发送 probe 的方式。

在我们的按需发送机制中，probe 消息的发送是由 cluster 成员的变化所触发。在 cluster 构造好之后，FM 节点就获得了 cluster 成员列表，然后它将这个列表发送给每个 cluster 成员以用来执行探测。一旦某个 cluster 成员发生了变化，其他的 cluster 成员就向网络发送带时间戳的 probe 消息，并等待其他 cluster 成员的回应。另一方面，当 cluster 成员收到了其他成员发来的 probe 消息之后，它就立即发送带时间戳的回应消息，当这个回应消息到达 probe 消息的发起者之后，发起者就通过两个时间戳来计算流逝的时间，这个时间就是 RTT。

在选择 cluster leader 的时候，我们需要用到下面的参数：

(1) RTT_{ij}：这是当前从成员 m_i 到成员 m_j 的 probe 的 RTT 值，其中 $i,j\in[0,n-1],i\neq j$，n 是 cluster 内的成员个数；

(2) $EARTT_{ij}$：这是 RTT 的指数均值。考虑到数据通信的爆发性和波动性，用 RTT_{ij} 的值是评价从成员 m_i 到成员 m_j 之间的网络性能不是太合适，因此用根据 RTT_{ij} 得到的参数 $EARTT_{ij}$ 来评价网络性能比较好，它的值是通过下面的公式来得到的：

$$EARTT_{ij}(m)=\alpha\left(EARTT_{ij}(m-1)\right)+(1-\alpha)RTT_{ij}(m) \qquad (3.1)$$

其中 $RTT_{ij}(m)$ 是从成员 m_i 到成员 m_j 的当前 RTT 值，$EARTT_{ij}(m-1)$ 是 RTT_{ij}

的前一个指数均值，$\alpha(0<\alpha<1)$是一个平滑因子，因为倾向给最近的 RTT 值较大的权值，故实验中将α的值设为 0.15；

(3) $SRTT_i$：这是成员m_i所获得的$EARTT_{ij}$值总和，它的值是通过下面公式计算

$$SRTT_i=\sum_{j=0,j\neq i}^{n-1}EARTT_{ij},i\in[0,n-1] \qquad (3.2)$$

$SRTT$是用来选择 cluster leader 的参数，选出的 leader 必须满足下面条件

$$SRTT_{core}=\min\{SRTT_i,i\in[0,n-1]\} \qquad (3.3)$$

为了防止 leader 选择时间的过长，我们的协议中定义了一个时间间隔I_P，一旦探测过程被激活，cluster 成员就发送 probe 请求给其他所有的成员，并等待回应消息。经过I_P长的时间之后，$n^1(n^1\leq n)$个已经获得$SRTT$值的 cluster 成员将它们的$SRTT$值发送给 FM 节点，FM 选出具有最小$SRTT$值的那个节点作为 cluster leader。其中，I_P是通过当今大部分 TCP 执行所采用的 Jacobson 算法来计算的：

$$I_P=\max\{EARTT_{ij}(m)+4*D_{ij}(m)\} \qquad (3.4)$$

其中$i\in[0,n^1-1],j\in[0,n-1],j\neq i,D_{ij}(m)$是一个起平滑作用的函数。$D_{ij}(m)$的计算是通过下式：

$$D_{ij}=\beta D_{ij}(m-1)+(1-\beta)|EARTT_{ij}(m)-RTT_{ij}(m)| \qquad (3.5)$$

其中β是一个平滑因子，取值和公式 3.1 中的α取值一样。

按需探测机制只在成员发生变化的时候才会产生开销，而且，由于协议所采用的分层和分群的层次化结构，一个 cluster 内的成员变化只会触发一小部分组成员的探测活动，而 cluster 之外的网络链接则不会被探测开支所影响，另外，关于时间间隔I_P的定义也避免了过长时间的探测，因此，我们采用的这种动态 cluster leader 选择机制只产生很小的延迟。

3.2.3 Layer 的构造

首先在每个 local area 范围内进行 layer 的划分，所有 local area 内的组成员首先都属于最低层 L_1，对这一层的成员进行 cluster 划分，并为每个 cluster 选择一个 cluster leader，然后本层上的 cluster leader 继续充当上一层 L_2 的普通成员，即第 L_i 层上的所有 cluster leader 构成了第 L_{i+1} 层，这个过程重复进行，直到当前层上的成员数目小于等于 3k-1 才停止，剩下的这些成员都划分到同一个 cluster 里，所选出的 cluster leader 充当本 local area 的 local core，至此，所有 local area 内的 layer 划分结束，每个 local area 都选出一个 local core，然后对所有的 local core 采用上述同样划分方式进行 layer 划分，直到最后选出一个节点作为全局的 core。到此为止，整个组播组的 layer 划分过程完成。

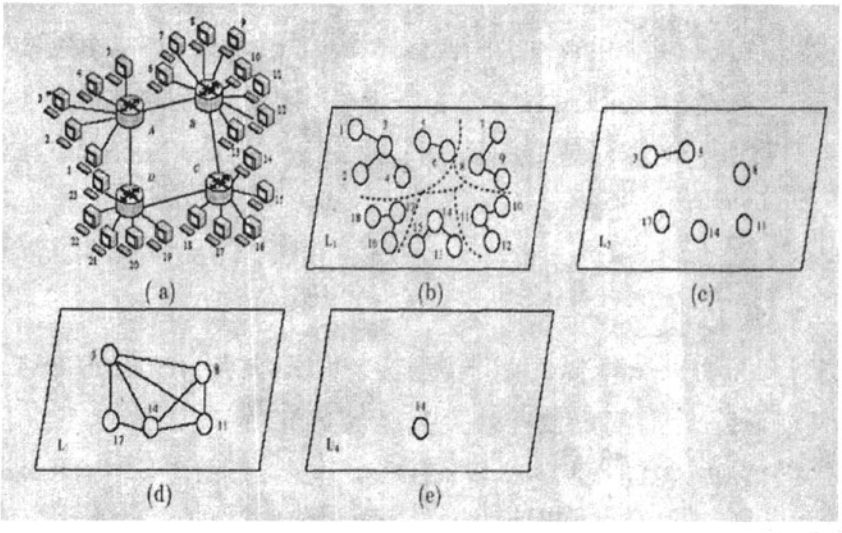

图 3.1 (a)初始网络拓扑（b）最低层 L1 （c）第 L2 层（d）第 L3 层（e）第 L4 层

下面通过图 3.1 的例子来说明 HFTM 的结构构造，图 3.1(a)给出的是初始网络，图 3.1(b)给出的是最低层 L_1 的情况，按照初始的网络拓扑特征，主机们被划分到不同的 local area 中。图 3.1(c)则给出了第 L_2 层上的成员构成，在本例中，主机 3、主机 5、主机 8、主机 11、主机 14 和主机 17 分别是 L_1 层上相应 cluster 的 cluster leader，在这一层上，主机 3 和主机 5 都是属于同一个 local area，所以构成了一个 intra-duster。图 3，1(d)则描绘了第 L_3 层上的情况，可以看出，主机 3 被选作对应 local area 中的 local core，其他主机也都是相应 local area 中

的 local core，所有这些 local core 组成了一个 cluster，主机 14 被选作本 cluster 地 Cluster leader，同时也是全局的 core，是位于第 L_4 层上的唯一节点，这一点从图 3.1(e)可以很清楚地看到。

3.3 生成路由信息

在 NICE 中，cluster 内的数据传输是由 leader 来负责，它将数据轮流发送给其他的 cluster 成员，但因为这种传送是串行进行的，所以花费的延迟比较大，效率不高，针对 NICE 的这个缺陷，HFTM 提出了改进，采用不同的组播路由方式。

HFTM 的组播路由分为 cluster 内部和 cluster 外部两种不同的情况分别讨论在 cluster 外部，HFTM 按照上节提到的含 layer 和 cluster 的层次化结构进行路由主要是通过 cluster leader 进行的组播路由，当有主机要发送数据给整个组播组时它首先将数据发送给所在 cluster 的 cluster leader，然后 cluster leader 除了负责将数据发送给本 cluster 内的成员之外，还负责将数据发送给它在上层所在的 cluster 的 cluster leader，然后 leader 继续负责向上层发送，直到数据传送到所有的组播成员为止。

在 cluster 内部，则采用不同的组播路由方式，这也是 HFTM 协议的重要创新之一，引入了一个新颖的基于斐波那契序列的组播算法[43]，它通过此算法将每个 cluster 的成员构造成一棵组播树，用此树对 cluster 内部成员进行组播路由。下面首先具体介绍基于斐波那契序列的组播算法。

基于斐波那契序列的组播算法是针对每个 cluster 的所有成员进行的，首先通过一种特定的方法将 cluster 成员构造成一个有序的成员序列，作为算法的输入，在介绍具体组播算法之前，先来介绍构造 cluster 成员序列的方法。

构造 cluster 成员序列的方法分两种情况讨论：

(1) 如果 cluster 内部的成员都属于同一个 local area，则将每个成员的权值定义为它到 cluster leader 的逻辑链路上的延迟距离。然后根据权值由低到高的顺序来排列这些成员。这样做的目的是保证延迟较小的成员可以携带更多的孩子节点，从而减少所花费的全部延迟。

(2)如果 cluster 内部的成员属于不同的 local area，即这是一类 inter-cluster，

则要采用不同的方法来确定成员的权值。每个成员 m_i，要维护两个参数，一个是 n_i 即成员 m_i 所属的 local area 内的成员数目，另一个是 d_i，即成员 m_i 到 cluster leader 的逻辑链路上的延迟距离。其中 n_i 的值在选择 leader 的时候已经得到，d_i 的值也已得到，所以都无需额外计算量。然后构造一个新参数 $w_i = \beta\frac{n_i}{n_{\max}} + (1-\beta)\frac{d}{d_{\max}}$，其中参数 $n_{\max}$ 表示 cluster 中拥有的 n_i 的最大值，$d_{\max}$ 表示 d_i 的最大值，参数 β 是一个平衡因子，用来决定给延迟距离多大的比重，在实验中，为了给延迟较大比重，我们采用 β=0.4。在构造好新的参数 w_i 之后，将此参数的值作为成员 m_i 的权值，然后按照 w_i 值的大小以递增顺序将成员构造成一个成员序列。在给成员指定权值的过程中，不仅考虑了延迟，而且还考虑了 local area 所包含的成员数目，这样保证了后面获得的组播树是一棵比较平衡的树，这也是 HFTM 比 NICE 考虑周到的一点。

将 cluster 成员成功构造成一个序列之后，就开始用基于斐波那契序列的组播算法来构造组播树。本算法的基本思想是基于斐波那契序列，将 cluster 成员序列进行不等长划分，使划分后一个子序列长度满足斐波那契序列某项的大小，最后得到一棵特殊高效的组播树。

该算法利用下面定义的斐波那契序列 $\{f_i\}$ 对目的节点进行不等长划分，使源负责更多目的节点的数据发送，$\{f_i\}$ 满足如下条件

$f_0 = 0,\ f_1 = 1,\ f_n = f_{n-1} + f_{n-2},\ if\ n > 1$

下面给出基于斐波那契组播算法的具体描述：

算法 1(基于斐波那契序列的组播算法)：

输入：成员序列 $\Phi = \langle d_1, d_2, \ldots d_k \rangle$，$d_s$ 是 cluster leader，充当源节点，$d_s \in \Phi$ 序列 Φ 中的成员数目是 K，而且 K 满足 $f_n \le K < f_{n+1}$，

输出：一棵由 Φ 中成员所构造的组播树

1 如果 K=2，d_s 将数据包传送给目的节点；

2 如果 K>2，将序列Φ划分为两个子序列Φ_1 和Φ_2，其中源要属于较大的一个子序列，使较小的子序列包含f_{n-2}个成员；

2.1 如果$(s > f_{n-2})$ 则$\{\Phi_1 = \langle d_1, d_2, d_3 \cdots, d_{f_{n-2}} \rangle; \Phi_2 = \langle d_{f_{n-2+1}}, d_{f_{n-2+2}}, \cdots, d_K \rangle;\}$

否则，则有$\{\Phi_1 = \langle d_1, d_2, \cdots, d_{K-f_{n-2}} \rangle; \Phi_2 = \langle d_{K-f_{n-2}+1}, d_{K-f_{n-2}+2}, \cdots, d_K \rangle;\}$

2.2 如果$(s > f_{n-2})$ {源d_s先将数据包传给d_1,然后d_1负责子序列Φ_1内的组播，d_s负责子序列Φ_2内的组播；}

否则{源d_s首先将数据包传给$d_{K-f_{n-2}+1}$，然后d_s负责子序列Φ_1内的组播，$d_{K-f_{n-2}+1}$负责Φ_2内的组播；}

3 通过递归调用算法 1 进行源d_1到子序列Φ_1的组播,以及从源d_s到子序列Φ_2的组播(或者从源d_s到子序列Φ_1的组播和从$d_{K-f_{n-2}+1}$到子序列Φ_2的组播)。

基于斐波那契序列的组播算法到此描述结束，图 3.2 给出了一个根据此算法获得的一棵斐波那契组播树的实例，其中成员序列是Φ=(4，6，5，3，7，2，1，0)。源是节点 0。

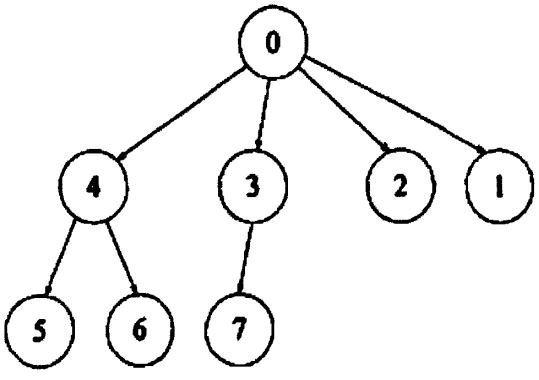

图 3.2 斐波那契组播树实例，成员序列Φ= (4，6，5，3，7，2，1，0)

3.4 成员维护

和之前的协议一样，HFTM 也采用阶段性的更新消息来跟踪组成员的变化，

将携带更新消息的 overlay 路径定义为维护路径，在任两个意图交换更新消息的主机之间会建立起一条维护路径。在 HFTM 中，每台主机只和它的直接上游成员和他的直接孩子成员之间建立维护路径。

下面对 HFTM 和 NICE 协议中 cluster 的最大维护代价进行比较。在 HFTM 和 NICE 中，最大的 cluster 大小分别是 $3k_1$-1 和 $3k_2$-1 其中 $k_1=2k_2$。而对 HFTM 和 NICE 这两种协议来说，cluster 的最大维护代价都发生在最大 cluster 的情况下。假定每条更新消息有 r 比特，在 NICE 中，一台主机的变化应该被通知给本 cluster 内的所有其他成员，因此 NICE 中的最大维护代价是 $r(3k_2-1)(3k_2-2)$。然而在 HFTM 中，每台主机只和它直接上游主机和他的直接孩子主机建造维护路径，所以它的最大维护代价是 $2r(3k_1-1)$。显而易见，由 HFTM 所创造的维护代价比 NICE 创造的要少的多，从而导致了 HFTM 具有较少的控制通讯量，并进一步导致了 HFTM 拥有较小组播延迟。接下来分别介绍 HFTM 如何处理成员加入和成员离开的情况：

3.4.1 组成员加入

在 HFTM 协议中存在一系列聚集点 *RPS*，当有新主机 v 想加入组播组时，首先查找离 v 最近的聚集点 RP_r，并向它发送 JOIN REQ 消息，然后聚集点 RP_r 在自己的 local core 列表里找到离 v 最近的 local core(用 l_a 表示)，并将 v 的加入通知 l_a，接着 l_a 检查自己在所有 layer 上的 cluster 成员列表，选出离自己最近的 cluster 成员，以决定将主机 v 加入到哪一个 cluster 中。HFTM 中 local core 的存在使得这个过程中的维护代价大大减少，因为它不像在 NICE 中那样，新加入的主机需要在每一层上分别查询离自己最近的成员，这是 HFTM 比 NICE 有优势的一点。新主机加入正确的 cluster 之后，cluster 的成员数目就增加了，新的尺寸记为 k_{new}，此时 cluster leader 需要根据 k_{new} 的大小做出相应的调整。我们知道，之前的成员个数表示为 K，而且满足 $f_n \le K < f_{n+1}$，此限制条件起着一个分界线的作用，如果 $k_{new} < f_{n+1}$，则只需将新加入的主机简单连到 leader 之上即可，而一旦满足了 $k_{new} > f_{n+1}$，就需要在包含新加入成员在内的所有 cluster 成员范围

内选择新的 leader，并构造一棵新的斐波那契组播树。

3.4.2 组成员退出

当某台主机 H 要离开现在所在的组播组，它需要发送一条 Remove 消息给 leader，然后 leader 计算当前 cluster 拥有成员的数目，标记为 K_{new}，当满足 $K_{new} < f_n$ 或者 H 本身就是 leader 的时候，需要在当前剩下的成员里选出新的 leader，并重新构造一棵斐波那契组播树。否则，如果上两个条件都不满足，则只需将成员 H 从组播树上删除，如果它有孩子节点，则将孩子节点简单连接到它的祖父节点上。

3.5 本章小结

当前的主流应用层组播协议在组播效率上仍存在不足，本章研究了如何改进应用层组播协议的组播效率，即如何减小平均组播延迟。提出了一种基于斐波那契序列的高效，可扩展的应用层组播协议—HFTM，它构造了一个特殊的层次化结构，采用划分 layer 和 cluster 的思想将整个组播组划分成了多个小型组。与之前协议不同的是，HFTM 充分考虑了底层网络拓扑结构，在划分 cluster 的时候考虑主机的实际物理位置，充分利用底层网络特征来大大减小了数据在昂贵链路上的传输概率，从而降低了组播延迟。另外，HFTM 采用了一种新颖的基于斐波那契序列的组播算法来将 cluster 内部成员构造成一棵有效的组播树以进行组播路由，从而减小了 cluster 内的组播延迟。除此之外，HFTM 的整体树高度也由于 cluster 大小的增加而进一步降低，从而使组播路径变短。以上的种种因素都使得 HFTM 的组播延迟性能得到大幅的改善。

第四章 性能分析与模拟实验

4.1 定性分析

下面对基于斐波那契序列的应用层组播协议—HFTM 进行性能分析，通过分析表明它是一种高效的应用层组播协议。

在斐波那契组播树的构造过程中，每个成员都被处理了一次，很显然，构造树的复杂度为$O(K)$，其中K是 cluster 的成员数目。

接下来是分析通过此算法将数据组播到K个成员所花的时间复杂度，首先定义几个符号：L表示一条 overlay 链路上的传输时间，t表示每台主机上的包处理时间。为了分析的简单起见，假定每条链路上的L值是一样的。$T(K)$表示将一份数据包从一个源节点组播到K个目的节点所花的时间，很显然，$T(K)$是一个关于K的单调递增函数。当数据包开始组播传输之后，源首先要将数据传送给分割节点，这个动作要花费$L+t$的时间，然后最初的成员序列就被划分为两个子序列。因为此算法具有递归特性，$T(K)$可以被表示成：

$$T(K)=\max\{T(f_{n-2})+L+t,T(K-f_{n-2})\}$$

其中$T(1)=0,T(2)=L+t;$

因为$f_n \le K < f_{n+1}$，所以$T(K)=\max\{T(f_{n-2})+L+t,T(K-f_{n-2})\}$

$$\le \max\{T(f_{n-2})+L+t,T(f_{n+1}-f_{n-2})\}$$

$$\le \max\{T(f_{n-2})+L+2t,T(f_{n+1}-2f_{n-2})\}$$

另外，因为$f_{n+1}-2f_{n-2}=f_{n-1}+f_{n-3}$，所以推出$f_{n-1} \le f_{n+1}-2f_{n-2} < f_n$

因此 $T(K) \le \max\{T(f_{n-2})+L+2t,\max\{T(f_{n-3})+L+2t,T(f_{n-1})\}\}$

$$=\max\{T(f_{n-2})+L+2t,T(f_{n-3})+L+2t,T(f_{n-1})\}$$

$$=\max\{T(f_{n-2})+L+2t,T(f_{n-1})\}$$

另外，我们知道$T(f_{n+1}) \le \max\{T(f_{n-2})+L+2t, T(f_{n-1})\}$

因此推出以下两个等式：

$$T(f_{n-1}) \le \max\{T(f_{n-4})+L+2t, T(f_{n-3})\}$$

$$T(f_{n-2}) \le \max\{T(f_{n-5})+L+2t, T(f_{n-4})\}$$

因此，当$f_n \le K < f_{n+1}$,有$T(K) \le \max\{T(f_{n-2})+L+2t, T(f_{n-1})\}$

$$\le \max\{T(f_{n-5})+2(L+2t), T(f_{n-3})\}$$

$$\le \cdots\cdots$$

$$\le \left\lfloor \frac{n}{2} \right\rfloor \cdot (L+2t)$$

此外，已知$f_n = \left\lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \right\rfloor$，其中$\varphi = \frac{1+\sqrt{5}}{2} \approx 1.62$（在 44 中证明过）

$$\Rightarrow \quad \log_2 f_n \approx n \cdot \log_2 \varphi + \frac{1}{2}\log_2 5$$

$$n \approx \frac{\log_2 f_n - \frac{1}{2}\log_2 5}{\log_2 \varphi} \approx 1.44(\log_2 f_n - 1.16)$$

另外，由于$f_n \le K < f_{n+1}$，因此可以获得$n < 1.44(\log_2 K - 1.16)$

$$\Rightarrow T(K) \le \left\lfloor \frac{n}{2} \right\rfloor \cdot (L+2t) \le \lfloor 0.72 \log_2 K \rfloor \cdot (L+2t)$$

从上面的分析可以看出，采用基于斐波那契序列的组播算法，要将一份数据包从一个源节点组播到 K 个成员所需要的时间为$O(\log_2 K)$。

下面通过形式化的方法对 HFTM 的层次化结构进行分析。将包含n个成员的组播组表示为$G = \{g_0, g_1, \ldots, g_i, \ldots g_{n-1}\}(i \in [0, n-1])$，包含$|S|$个源的源集合表示为$S = \{s_0, s_1, \ldots s_{|s|-1}\}(j \in [0, |S|-1])$。将源$s_j$和组成员$g_i$之间的 overlay 路径表示为$\langle s_j - g_i \rangle$。在路径$\langle s_j - g_i \rangle$上包含$|F|$个转发者的转发者集合表示为$F = \{f_0, f_1, \ldots, f_{|F|-1}\}(m \in [0, |F|-1])$。假定路径$\langle s_j - g_i \rangle$覆盖了$L$条底层物理链路，则穿越路径$\langle s_j - g_i \rangle$的一份数据包$p$在时刻$t$所经历的延迟是

$d_{\langle s_j-g_i\rangle}(p,t)=\sum_{l=0}^{L-1}d_l^{pr}+\sum_{l=0}^{L-1}d_l^{tr}(p,t)+\sum_{m=0}^{|F|-1}d_{f_m}^{e}(p,t)$，其中$d_l^{pr}$表示第$l$条物理链路上的传输延迟；$d_l^{tr}(p,t)$表示在第$l$条物理路径上的包延迟，其中包含链路传输延迟和包的排队延迟；$d_{f_m}^{e}(p,t)$则表示在主机f_m上的包处理延迟，即包复制转发的时间。由此得到从源s_j发出的数据包p在时刻t的平均组播延迟是$\overline{d_{s_j}(p,t)}=\frac{\sum_{i=0,g_i\neq s_j}^{n-1}d_{\langle s_j-g_i\rangle}(p,t)}{n-1}$。为了获得较小的平均组播延迟，可以通过减小$\sum_{l=0}^{L-1}d_l^{pr}$,$\sum_{l=0}^{L-1}d_l^{tr}(p,t)$和$\sum_{m=0}^{|F|-1}d_{f_m}^{e}(p,t)$这三部分来获得。

在HFTM中，分层和分群的层次化结构将整个组播组划分成了多个小的“迷你组”，因此大大的减少了组成员之间的阶段性组维护代价，从而减小了$\sum_{l=0}^{L-1}d_l^{tr}(p,t)$的值。另外，HFTM 在划分 cluster 的时候考虑了底层的网络拓扑特征，从而减少了数据包在 backbone area 内代价昂贵链路上的传输机会，这导致了$\sum_{l=0}^{L-1}d_l^{pr}$的减小。除此之外，HFTM 协议将 cluster 成员构造成一棵高效的斐波那契组播树，因为 cluster 的尺寸增加了，所以整体结构的高度减，导致路径$\langle s_j-g_i\rangle$上的转发者数目减小，从而使得$\sum_{m=0}^{|F|-1}d_{f_m}^{e}(p,t)$也减小了。因此，HFTM 协议的平均端到端延迟比较小。图 4.1 给出了一个实例，用来阐述 HFTM 和 NICE 协议在划分 cluster 方法上的不同，也可以由此例看出 HFTM 相对于 NICE 的优势所在。在图 4.1(a)中的 NICE 协议，在 cluster1 和 cluster3 内的通讯需要分别用到 R_1 和 R_2 之间的链路、R_2 和 R_3 之间的链路、以及 R_3 和 R_4 之间的链路。然而在图 4.1(b)中的 HFTM 协议，cluster 内部的数据传输只需要使用本地局部链路即可，由此大大减少了数据传输延迟。

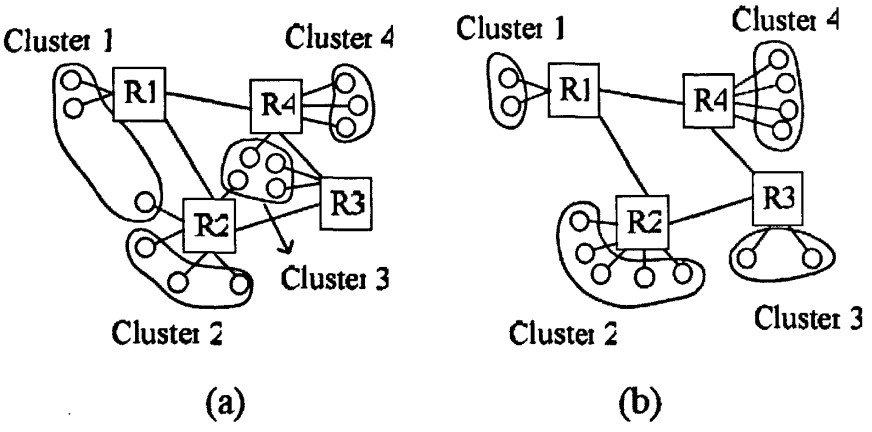

图 4.1 在最低层划分 cluster 时 NICE 与 HFTM 的区别：(a)NICE;(b)HFTM

另外，HFTM 在构造组播树的时候考虑了不同 local area 所包含的组播组成员数，这样做的目的是为了避免由于组播组成员在 local area 内分布极度不均匀而获得一棵极度不均衡的组播树。HFTM 协议通过对 local area 尺寸的考虑，可以获得一棵均衡的组播树。

4.2 仿真模拟

通常来说，针对网络情况的测试可以在实际环境中进行，也可以在仿真环境中进行。因为在实际环境中的测试具有难以再现等劣势，所以选用了仿真环境。学术界比较偏爱的网络仿真工具是 NS2[45]。但是 NS2 本身并不能自动生成节点，并指定节点间的物理连接(将网络中的节点和节点间的连接称为物理拓扑)物理拓扑，即必须通过手工输入节点的分布和连接，所以需要我们自己来生成物理拓扑。

4.2.1 NS2 简介

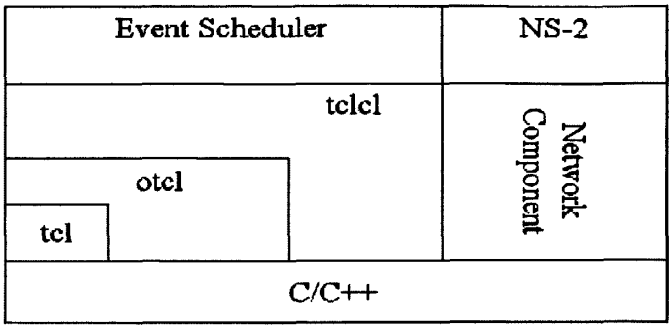

图 4.1 NS2 体系结构

网络仿真器(Network Simulator，简写 NS)是劳伦斯伯克利国家实验室网络研究组开发的仿真器，它的组成如图 4.1 所示。NS 可扩展性好，易配置，是由事件驱动的仿真引擎。

NS 设计的出发点是网络仿真，集成了多种网络协议（如 TCP、UDP），业务类型（如 FTP、Telnet，Web，CBR&VBR 等），路由排队管理机制（如 Drop Tail，RED&CBO 等），路由算法（如 Dijkstra 算法等）。此外，NS 还集成了组播业务和应用于局域网仿真的部分 MAC 层协议。使用 ns 命令可以定义网络拓扑，配置传输的源和接收器，收集统计信息，然后启动仿真

NS2 采用 C++和 OTcl 两种语言是从需要出发的，协议的细节实现需要运行速度快，而网络研究需要不断调整配置参数，但是每种参数只需要运行一次，因此不要求很快的运行速度。这两个要求分别通过 C++和 OTcl 可以实现。从这个应用要求出发，我们能够很明白的知道什么功能应该通过 C++来实现，什么功能应该通过 OTcl 来实现，如表 4.1 所示

表 4.1 OTcl 和 C++功能简介

语言	实现功能	举例
OTcl	配置，结构和“一次性”事务 使用已存在 C++对象	物理拓扑和逻辑拓扑的连接 Agent 的连接
C++	处理每个包 修改已存在 C++类的行为	定义新的包 对接受的包进行处理

NS2 还提供了使网络拓扑可视化的工具，使得网络拓扑简明易。此外，NS2 还提供的 trace 文件，记录下每个节点的发送包、接收包以及丢失包的情况。

在 NS2 中，可以创建终端、路由器等网络节点。每个节点使用的网络协议可以自己定义。节点间的互发数据通过 agent 进行。agent 完全由使用者定义，可以在 agent 中使用用户需要的包结构，通过在包中加入一定的结构，在此结构中填入相应的数据，从而通过数据分析得到相关信息。使用者也可以在 agent 中定义发送包前和接收包后的动作，例如发送前的数据准备，收包后的数据分析等。这样就提供了极大的自由度。

4.2.2 NS2 上加入应用层协议

NS2 允许用户采用自己的协议，可以在 MAC 层至应用层中的任何一层加入协议。

加入应用层协议可能涉及的修改有：

一、对包结构的修改，定义自己的包头。通常，包头的定义在 C++代码中实现，然后还需要在 packet.h 文件中加入新包的类型，在 tcl/lib/ns-default.tcl 中加入包大小的定义，否则为默认的包大小。但是这一步并不一定是必须的，例如在加入 HFTM 协议中，我们采用的即是 NS2 中已经集成的 udp 包。

二、定义协议相关的 agent，即定义 Agent 类的子类，重点是处理 command() 函数。command 函数形式为：int command(int argc, const char*const* argv)，它负责处理自定义 Agent 子类的对象的 Tcl 命令。例如子类对象为 pa，则$pa send 则交由 command 函数处理。通常由 command 函数来调用其它成员函数，例如 send、recv 等。

我们还需要修改 Makefile 文件里的依赖关系，并重新编译 ns2。

4.2.3 NS2 与 GT-ITM 的联合应用

本文采用 Georgia 理工学院的网络拓扑生成器 GT-ITM [47] 来生成物理拓扑。GT-ITM 运行在 Linux 环境下，可以根据用户设定的参数生成具有一定特性的物理拓扑。GT-ITM 生成的物理拓扑以.gb 文件的形式保存，而 NS2 能处理的语言是 C/C++或 tcl 语言，而拓扑构建部分只能用 tcl 语言撰写，因此涉及到

两个工具之间的联合应用。

幸运的是，目前有很多工具来完成这一工作，通常是 sgb2xxx，例如 sgb2ns，sgb2comns，sgb2hierns。它们都用于 gb 文件到 tcl 文件的转变，只是生成的 tcl 文件在形式上略有不同。这些工具都必须在 Linux 环境下使用，使用方式很简单，例如 sgb2comns，在命令行中输入

sgb2ns <sgb-graph> <outfile.tcl>

例如我们有 ts100.gb 文件，那么输入

sgb2ns ts100.gb ts100.tcl r

就可以得到对应的 tcl 文件。这个 tcl 文件只有一个函数 create-topology，这个函数负责生成节点，并根据 gb 文件的描述生成节点间的双向链接，链接的带宽作为函数参数，由用户指定，所有的链接带宽是一定的

实验所用的骨干网如图 4.2 所示，在这个骨干网中，包含 19 个路由器，组播组中的主机直接或间接连接到这些路由器上。

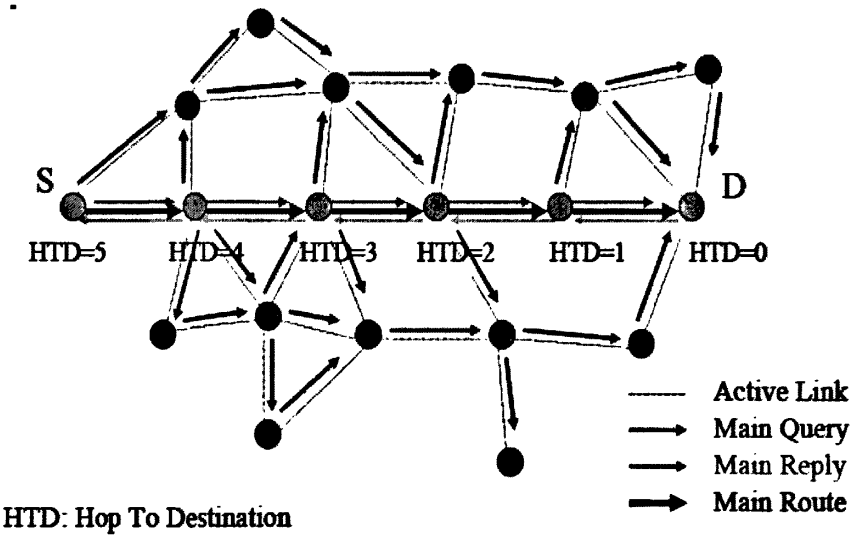

图 4.2 模拟实验中所用的骨干网结构

4.2.4 仿真参数

在模拟实验中，将骨干网内的链路带宽设为 1000Mbps，局部区域内链路带宽设为 100Mbps。另外，将骨干网内每条链路的链路代价设为介于 400 和 600 之间的一个随机整数，局部区域内每条链路的代价设为介于 40 和 60 之间的随

机整数，模拟所用的通信量是一个流速为 1.5Mbps 的 ftp 背景流。

在模拟实验中，我们对 NICE、CAN-based multicast 和 HFTM 这三种应用层组播协议进行了性能比较。为了比较过程的方便，定义了以下几个参数：1)平均端到端延迟 AED(Average End-to-end Delay)：为从一个源到整个组播组成员的端到端延迟总和与组播组成员数目的比值；2)平均链路压力 ALS(Average Link stress)：它的定义是同一份数据包拷贝在底层链路上传输的总次数与链路数目的比值，代表的是同一份数据包在一条底层链路上流过的平均次数，在应用层组播中，链路压力是不可避免的，它会导致消耗额外的网络资源，从而增加组播延迟，并且影响协议的可扩展性；3)平均代价压力 ACS(Average Cost stretch)：ACS 的定义是数据包在组播到所有组播组成员的过程中在所有链路上所花费的代价之和与数据包传输所经过的所有链路数目的比值。一般来说，拥有较小的代价压力表示数据传输时经过了较少的骨干区域内的链路。

我们通过两个不同的仿真实验来比较三种协议在 AED、ALS 和 ACS 性能上的不同。第一个仿真实验考察的是单源情况下三种协议的性能比较，随着组播组的成员数目从 50 变化到 600，比较三种协议 NICE、CAN-based multicast 和 HFTM 在性能 AED、ALS 和 ACS 方面的表现。在第二个仿真实验中，考察的是多源情况下的三种协议性能比较，其中，组播组成员的数目是固定的，随着发送源数目的变化，对 NICE、CAN-based multicast 和 HFTM 三种协议在 AED、ALS 和 ACS 性能上进行比较。通过这两个仿真实验表明 HFTM 协议无论在单源还是多源情况下都具有更好的组播延迟性能。

4.3 结果比较与分析

我们通过两个不同的仿真实验来比较了三种协议在 AED、ALS 和 ACS 性能上的不同。第一个仿真实验考察的是在只有一个发送源的时候，当组播组的成员数目从 50 变化到 600 时，三种协议 NICE、CAN-based multicast 和 HFTM 在性能 AED、ALS 和 ACS 方面的表现。

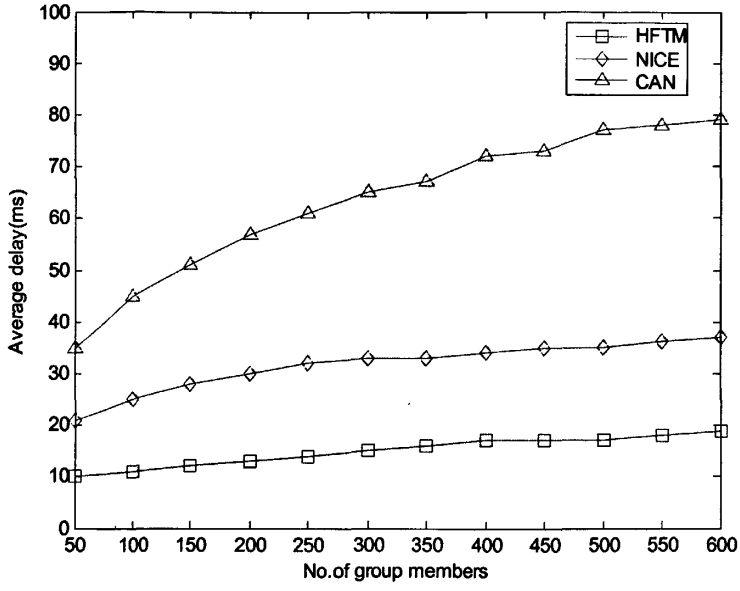

图 4.3 单源情况下三种协议的 AED 性能比较

图 4.3 给出的是三种应用层组播协议的 AED 性能比较。从图上曲线可以看出，由于 CAN-based multicast 采用洪泛的方法进行组播路由，导致了较大的延迟，因此它的 AED 性能比另两个协议都差。而 HFTM 因为在划分 cluster 的时候考虑了底层网络拓扑特点，引入了一个 local area 的概念，从而大大减少了在代价昂贵的链路上数据传输的机会，而 NICE 则忽视了底层拓扑特点这个因素，因此 NICE 的 AED 性能比 HFTM 差；另外，HFTM 中的 cluster 尺寸比 NICE 中的大，所以，在组播组成员数目相同的情况下，HFTM 中的 cluster 个数比 NICE 的少，于是整个结构的高度也比 NICE 中的低，从而使组播转发路径长度减小，这也使 HFTM 的组播延迟明显减小，以上因素都导致了在三种协议中，HFTM 有最好的 AED 性能。

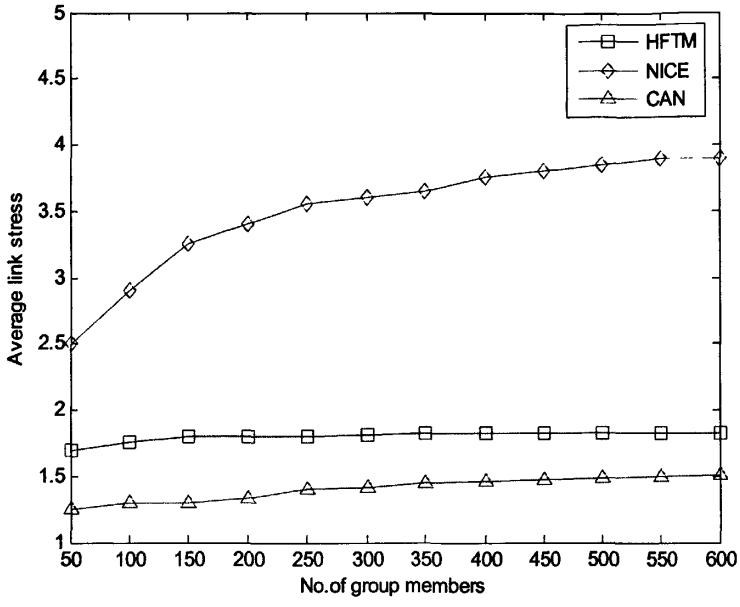

图 4.4 单源情况下三种协议的 ALS 性能比较

三种协议的 ALS 性能比较如图 4.4 所示，从图中可以看出，HFTM 的 ALS 性能比 NICE 优越，这是因为在 NICE 中，数据都是由 cluster leader 负责发送给 cluster 内部成员，因此在较高层上的 cluster leader 的流出链路通常会受到较大的链路压力的困扰。然而，在 HFTM 协议中，通过使用斐波那契组播树，cluster leader 的部分压力被转移到普通成员身上，因此，HFTM 的 ALS 性能比 NICE 的好。而在 CAN-based multicast 协议中，采用洪泛路由方法来进行数据组播，使得数据通信量和链路压力可以在所有的链路上均匀分配，因此在这三种协议中，CAN-based multicast 的 ALS 是最小的。

图 4.5 给出了三种协议的 ACS 性能比较，NICE 和 CAN-based multicast 这两个协议都忽略了底层网络特点，因此，数据就会在一些代价昂贵的链路上频繁的经过，导致一些多余的代价产生，而 HFTM 考虑了这个因素，因此它的 ACS 性能是三个协议中最好的。

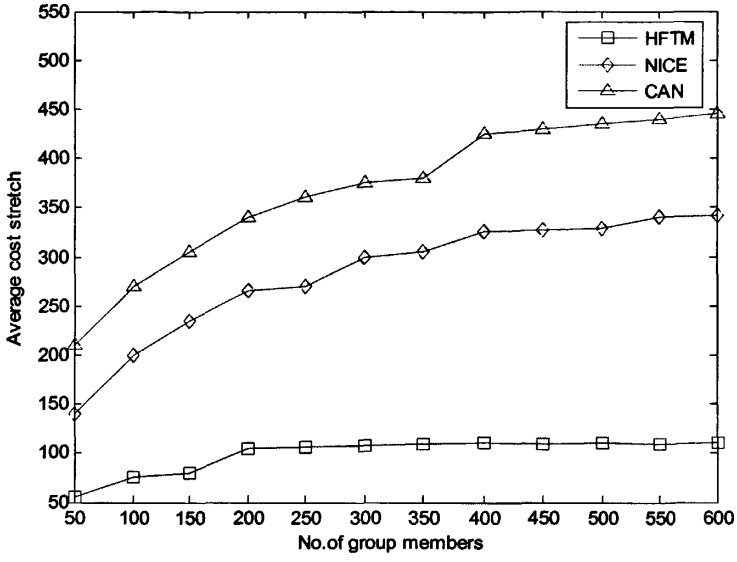

图 4.5 单源情况下三中协议的 ACS 性能比较

在第二个仿真实验中，考察的环境有所改变，发送源的数目不是局限为一个，而是在一定的范围内变化，其中，这些多余的发送源是随机产生的。组播组成员的数目一直维持在 100，在这种情况下，对 NICE、CAN-based multicast 和 HFTM 三种协议在 AED、ALS 和 ACS 性能上进行比较。另外，我们分别对发送源数目在 1 到 10 范围内变化的情况下给出协议的性能比较。

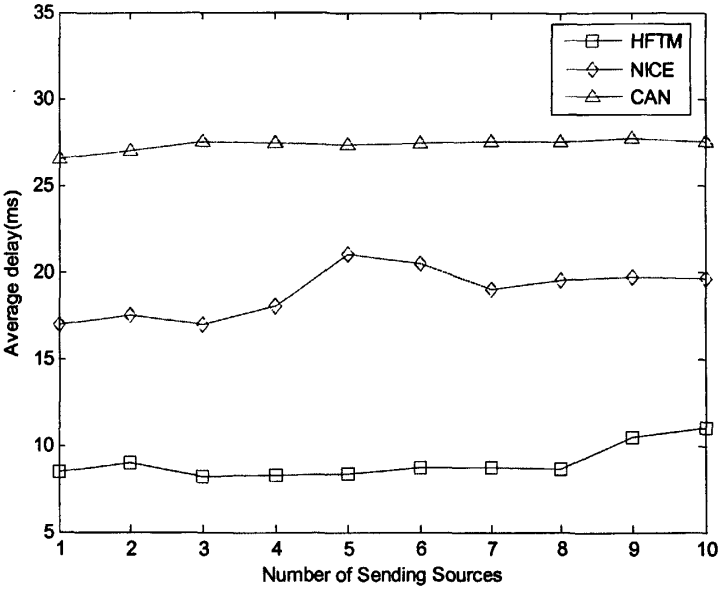

图 4.6 源数目在 1 到 10 范围内三种协议的 AED 性能比较

图 4.6 描绘了当发送源数目从 1 到 10 之间变化时，上述三种协议的 AED 性能比较。此图中的曲线关系与图 4.3 有点类似，因为 CAN-based multicast 采用的洪泛算法导致了它具有最差的 AED 性能，而 NICE 中划分层次和 cluster 时底层网络特点的忽视也导致数据包花费较长的延迟到达目的地，而 HFTM 中 local area 的引入和斐波那契组播树的构造使得数据包的传送所花费的延迟比 NICE 中的小，因此这种情况下，HFTM 具有比 NICE 更好的 AED 性能。

图 4.7 表现的是当发送源数目在 1 到 10 之间变化时，上述三种协议的 ALS 性能比较。从图中可以看出，多源情况下，HFTM 的 ALS 性能比 NICE 优越，在多源情况下，NICE 中在较高层上的 cluster leader 的流出链路更容易受到较大的链路压力的困扰。然而，在 HFTM 协议中，通过斐波那契组播树的使用将 cluster leader 的部分压力转移到普通成员身上，因此，HFTM 的 ALS 性能比 NICE 的好。由于 CAN-based multicast 协议采用的洪泛路由方法使得数据通信量和链路压力可以在所有的链路上均匀分配，因此 CAN-based multicast 是这三种协议中 ALS 性能最好的。

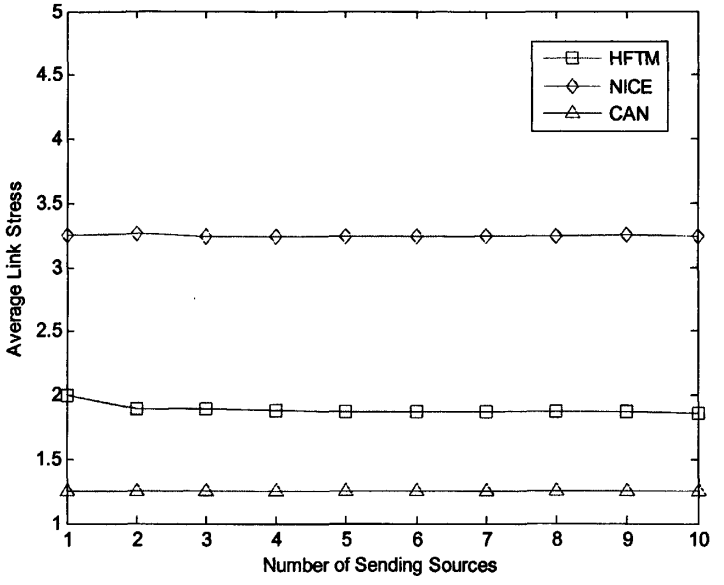

图 4.7 源数目在 1 到 10 范围内三种协议的 ALS 性能比较

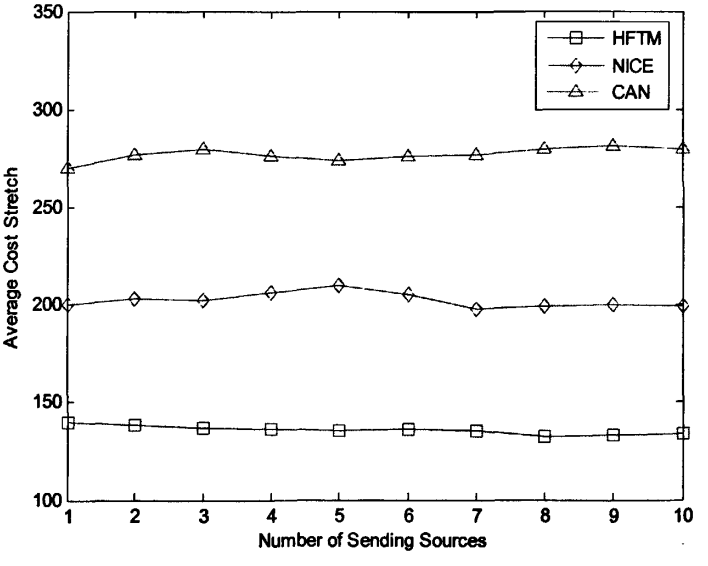

图 4.8 源数目在 1 到 10 范围内三种协议的 ACS 性能比较

图 4.8 表现的是当发送源数目在 1 到 10 之间变化时，上述三种协议的 ACS 性能比较。由于 NICE 和 CAN-based multicast 这两个协议都忽略了底层网络特征，因此，组播数据在一些代价昂贵的链路上(比如不同 local area 之间的链路)频繁地经过，导致产生的多余代价较多，而 HFTM 协议在设计时考虑了底层网

络特征，因此它的 ACS 性能是三个协议中最好的。

另外在第二个仿真实验中，还对上述三种协议的链路特征进行了比较，具体见表 4.2，表格中的数据表明了 HFTM 的 ALS 性能比 NICE 好，因此导致它的效率和可扩展性也比 NICE 好。另外，可以看到，在 HFTM 中，所用的链路数目比 CAN-based multicast 协议中所用的少，可以使得更多的资源节省下来进行数据传输。

表 4.2 CAN-based multicast, NICE 和 HFTM 三种协议在链路特征上的比较

协议名称	ALS	使用链路数目
CAN-based multicast	1.464	197
NICE	3.122	130
HFTM	1.854	123

4.4 本章小结

为了对 HFTM 进行性能评估，本章采用 NS-2 结合 GT-ITM 的模拟实验将它与两种主流应用层组播算法 NICE 和 CAN-based multicast 进行了性能比较，主要针对单源情况和多源情况在平均组播延迟、平均链路压力和平均代价压力三方面进行了比较。分析了 NS2 的仿真数据，结果表明无论是单源情况还是多源情况，HFTM 协议在组播延迟方面都具有明显优势，ALS 性能比 NICE 好，所用的链路数目比 CAN-based multicast 协议中所用的少。因此，HFTM 是一种可扩展，分布式，高效的应用层组播协议。

第五章 总结

通过上面的分析比较，可以得出，在三种协议中，无论是单源还是多源情况，HFTM 在平均端到端延迟 AED 上的性能都是最优的，它的延迟性能可以满足实时流应用的需要。另外，单源和多源情况下 HFTM 的 ALS 性能也都比 NICE 要好。因此综合来讲，HFTM 是一种较好的应用层组播协议，在组播效率方面和可扩展性方面都具有优势。

作为一种有效的组播通讯方式，IP 组播可以将数据同时分发给多个接受者。但由于 IP 组播的固有限制，以及当前网络体系结构和操作模型的限制和制约，使它在全球范围内的推广步速大大减缓。随着网络中涌现出的多样化应用越来越多，对组播通讯模式的需求也越来越大，因此急需一种有效的组播替代方案来解决 IP 组播在推广中遇到的问题。在这种情况下，研究者提出了应用层组播，希望可以在现行网络基础设施上大规模推广组播通讯。

应用层组播的基本思想是屏蔽底层物理网络的拓扑细节，将组成员节点自组织成一个逻辑覆盖网络，由组成员节点来实现组播的复制，转发功能。应用层组播的提出有利于高级网络服务的推广和全球推广的加速。然而，由于应用层组播的组播功能由主机实现，相对于 IP 组播而言，应用层组播在节点的稳定性，组播效率等方面都面临严峻的挑战。因此，对应用层组播算法进行研究并致力于提高应用层组播的相关性能对于互联网的发展有很重要的意义。

作为全文的总结，本章首先对本文的工作进行了概括，然后给出了本文的贡献和创新之处，最后指出了今后进一步的研究方向。

5.1 本文工作总结

本文针对如何在覆盖网络上提供高效，可扩展，分布式的组播服务，开展了以下研究工作：

深入分析了当前的应用层组播方案，对应用层组播问题进行了抽象描述，提取其中的关键元素，在抽象层次上对应用层组播路由问题进行了分类，在此基础上对应用层组播的组播路由问题进行了分析讨论。

其次把现有的应用层组播方案划分为两大类集中式算法和分布式算法，并

按照数据拓扑和控制拓扑的构造顺序把分布式算法划分为 Mesh 优先，Tree 优先和隐式构造三大类。在总结现有应用层组播应用的基础上，提出了一种适用的应用层组播系统的协议栈模型。

最后关于如何减少在实时的多媒体应用中的组播延迟，虽然目前在树型拓扑上提出的应用层组播不少，但在延迟这个衡量标准上一直不尽人意。因此本文研究如何提高应用层组播协议的组播延迟性能，提出了一种基于树型拓扑结构的可扩展，高效应用层组播协议的组播协议。该协议在设计时充分利用了底层的网络特征，并且构造了高效的斐波那契组播树以进行组播路由。与当前主流的应用层组播协议相比，该协议的组播延迟性能得到了很大提高。

5.2 本文的主要贡献和创新

基于树型拓扑结构，提出一种可扩展的高效应用层组播协议—HFTM。本文采用考虑主机实际拓扑位置的方法构建了一个分层(layer)和分群(cluster)的层次化结构。另外，通过基于斐波那契序列的组播算法的使用，构造了一棵斐波那契树来进行 cluster 内部的组播数据路由。HFTM 协议的创新之一就是充分利用底层的网络特征，即主机的实际拓扑位置，将整个组播组基于不同的 local area 分割成多个小型组，从而减少了数据包在代价昂贵的骨干网链路上传输的频率。另外，为每个 cluster 选择 cluster leader 的时候，不同于以往的静态方法，本文采用的是基于瞬间网络形势的动态方案，通过发送探测消息的动态机制来动态地选择 leader，从而使所选出的 leader 可以反映网络的动态变化，能更好地负责 cluster 内的组播，继而使 HFTM 的组播延迟性能更好。实验表明本文提出的 HFTM 协议同传统的协议相比具有更好的组播延迟性能。

5.3 下一步的研究工作

本论文着重于覆盖组播路由协议和算法的研究。针对实时多媒体应用的应用层组播服务的开展，以下几个方面可以作进一步的研究工作：

●HFTM 的仿真测试还只是局限在模拟环境中，可以在该协议的基础上进一步实现应用层组播原型系统，实现服务定制和实时多媒体流的传输。有条件的可以在实际的局域网中完成安装和测试。

●Peer-to-Peer 技术是当前的热点，P2P 网络是一种特殊的 Overlay 网络，如何将 P2P 技术自身的一些特性与组播结合起来，带来更好的性能改进，也是值得进一步研究的课题。

参考文献

[1]. 余义建，史能，IP 组播分析及模式改进。计算机工程，2001 年 11 月。第 27 卷，第 11 期.

[2]. Francis P.Yoid: Extending the Multicast Internet Architecture[R].Technical Report, Berkeley: AT&T Center for Internet Research at ICSI(ACIRI), 2000. http://www.aciri.org/yoid/.

[3]. Abad C, Yurcik W, Campbell R.A survey and comparison of end system overlay multicast solutions suitable for network centric warfare[C].In Proceedings of SPIE(The International Society of Optical Enginerring),2004.

[4]. Stephen Deenrig. Multicasting routing in a datagram internetwork. In Ph.D. dissertation,1991.

[5]. Internet Group Management Protocol, Version 3,ftp://ftp.rfc-editor.org/in-notes/rfc3376.txt

[6]. Diot C, Levine BN, Lyles B, et al. Deployment Issues for the IP Multicast Service and Architecture[J].IEEE Network,Jan.2000:78–88.

[7]. 蒋东星,郑少仁. IP 网络组播技术的新发展. 电信科学. 2003

[8]. R.Braynard, D.Kostic, A.Rodriguez, J.Chase and A.Vahdat. Opus:An overlay utilityservice.In OPENARCH,2002.

[9]. Wang B, Hou J. Multicast Routing and Its QoS Extension: Problems,Algorithms,and Protocols[J]. IEEE Network, 2000, 14(1):22-36.

[10]. Hassin R, Tamir A. On the minimum diameter spanning tree problem[J]. Inform. Processing Lett, 1995, 53:109-111.

[11]. Wang Z, Crowcroft J. QoS Routing for Supporting Resource Reservation[J]. IEEE Journal on Selected Areas in Communications, 1996, 14(7):1228-1234.

[12]. Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness[M]. San Francisco, CA: W.H.Freeman, 1979.

[13]. Salama HF, RSalama HF, Reeves DS, Viniotis Y. The Delay-Constrained Minimum Spanning Tree Problem[C]. In Proceedings of the International Symposium on Computers and Communications (ISCC'97), June 1997.

[14]. Garg N, Khandekar R, Kunal K, et al. Bandwidth Maximization in Multicasting[C]. In

Proceedings of the 11th Annual European Symposium on Algorithms, September 2003. 242-253.

[15].Banerjee S, Kommareddy C, Kar K, et al. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications[C]. In Proceedings of the IEEE INFOCOM. 2002. San Francisco. 1521-1531.

[16].Shi S, Turner J. Multicast Routing and Bandwidth Dimensioning in overlay networks[J]. IEEE Journal on Selected Areas in Communications, 2002,20(8):1444-1455.

[17].Bauer F, Verma A. Degree-constrained multicasting in point-to-point networks[C]. In Proceedings of IEEE Infocom'95, 1995.369-376.

[18].Malouch NM, Liu Z, Rubenstein D, et al. A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted, End-System Multicast[A]. In Proceedings of ACM NOSSDAV'02[C], 2002.

[19].李伟,沈长宁. 应用层组播协议的研究[J]. 计算机工程与应用,2004 (24) : 157 - 158.

[20].Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. Almi:An application level multicast infrastructure. In Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems(USITS), pages 49--60, 2001.

[21].V.Roca and A.El-sayed, "A host-based Multicast(hbm) Solution for Group Communications," 1st IEEE Int'l. Conf. Networking, Colmar, France, July 2001.

[22].S.Shi and J.Turner. Routing in Overlay Multicast Networks. In Proc.of IEEE INFOCOM, June 2002.

[23].S.Shi and J.Turner. Multicast Routing and Bandwidth Dimensioning in Overlay Networks. Journal on Selected Areas in Communications, 2002.

[24].S.Banerjee and B.Bhattacharjee. A comparative study of application layer multicast protocols. In Submitted for review, 2002.

[25].Y.Chu, S.G.Rao, and H.Zhang."A Case for End System Multicast".In ACM Sigmetrics 2000, Santa Clara, California, USA,2000.

[26].Y.Chawathe.Scattercast:An Architecture for Internet Broadcast Distribution as an Infrastructure Service.PhD thesis,University of California,Berkeley,USA,Dec. 2000

[27].J.Liebeherr, M.Nahas, and W.Si. "Application-layer multicasting with delaunay triangulation overlays". IEEE JSAC,20(8):1472--1488, October 2002.

[28].S.Banerjee, C.Kommareddy, K.Kar, B.Bhattacharjee, and S.Khuller.Construction of an efficient overlay multicast infrastructure for real-time applications.In Proceedings of INFOCOM, San Francisco, CA, April 2003.

[29].P.Francis. "Yoid: extending the multicast internet architecture", 1999.

[30].B.Zhang, S.Jamin, and L.Zhang. Host multicast:A framework for delivering multicast to end users. In Proceedings of IEEE Infocom, June 2002.

[31].D.Helder and S.Jamin. "End-host multicast communication using switchtrees protocols". Global and Peer-to-Peer Computing on Large Scale Distributed Systems, 2002.

[32].Zhi Li, Prasant Mohapatra. "HostCast:A New Overlay Multicasting Protocol".

[33].N.M.Malouch, Z.Liu, D.Rubenstein, and S.Sahu. A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted,End-System Multicast. In 12th International Workshop on Network and Operating System Support for Digital Audio and Video(NOSSDAV'02), May 2002

[34].S.Banerjee, B.Bhattacharjee and C.Kommareddy.Scalable Application Layer Multicast.In Proceedings of ACM SIGCOMM, Pittsburgh, PA, August 2002.

[35].D.Tran, K.Hua, and T.Do. "ZIGZAG:An Efficient Peer-to-Peer Scheme for Media Streaming," Univ.Central FL,Orlondo,tech.rep.CS-UCF 2002,2002.

[36].S.Ratnasamy, M.Handley, R.Karp, and S.Shenker,"ApplicationLevel Multicast Using Content-Addressable Networks,"Proc.Third Int'l Workshop Networked Group Comm.,Nov. 2001.

[37].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(JSAC),2002.

[38].S.Q.Zhuang, B.Y.Zhao, A.D.Joseph, R.H.Katz, andJ.Kubiatowicz, "Bayeux:An Architecture for Scalable and Fault tolerant Wide-Area Data Dissemination,"Proc.11th Int'l Workshop Network and Operating System Support for Digital Audio and Video(NOSSDAV 2001.

[39].刘业,刘林峰,庄艳艳. 面向服务的P2P网络体系结构层次参考模型研究[J]. 中国工程科学, 2007,(09) .

[40].E.Zegura, M.Ammar, 2.Fei, and S.Bhattaeharjee. Application-Layer Anycasting: A Server Selection Architecture And Use in A Replicated Web Service. IEEE/ACM Transactions on Networking, vol.8, no.4, Page455 — 466, August2000.

[41].L.Breslau, E.Knightly, S.Schenker, I.Stoica, and H.Zhang. Endpoint Admission Control: Architecture Issues And Performance". In Proc.of SIGCOM 2000, page 57-69, August 28-Septermber 1, 2000, Stockholm, Sweden.

[42].W.Jia, D.Xuan, W.Tu, L.Lin, and W.Zhao. Distributed admission control for anycast flows. IEEE Transaction on Parallel and Distributed Systems, vol.15, no.6, page 673-686, June 2004.

[43].顾乃杰, 李伟, 刘婧. 基于斐波那契序列的多播算法[J]. 计算机学报, 2002,25(4) : 365 - 372.

[44].R.M.Karp, A.Sahay et al.. Optimal broadcast and summation in the LogP model. Proc the 5th Annual ACM Symposium on Parallel Algorithms and Architectures, Velen, Germany, 1993.

[45].http://nsnam.isi.edu/nsnam/ DB

[46].刘业. 适应自组织管理模式的 P2P 网络技术的研究[D]. 东南大学, 2006 .

[47].E.W. Zequra, K.L.Calvert, S.Bhattacharjee. How to model an internetwork. INFOCOM 1996, Vol. 2: 594~602.

致 谢

值此论文完成之际，谨向多年来所有关心和帮助过我的人们表达最诚挚的谢意！感谢我的导师达力教授，没有他的悉心指导和帮助，本论文的研究工作不可能顺利完成。期间达老师渊博的学识，严谨求实的治学态度，创新的精神，敏锐的学术洞察力给我留下非常深刻的印象，必将使我一生受益。

感谢计算机学院的其他老师，王博亮，李翠华，赵致琢，陆达，杨晨辉，鞠颖等，正是他们在学术上孜孜不倦的科研精神和平日里对学生的严格要求激励着我敢于克服苦难，迎接挑战。他们积极进取的科研精神，循循善诱的治学态度，光明磊落的人格特质，细微体贴的关怀帮助，这些都使我倍感欣慰。

在厦大这三年，我又结识了来自五湖四海的很多同学，和他们在一起学习和生活，使我获益良多。在这里，我要感谢计算机系06级研的全体同学。感谢实验室的冯星泉，胡斌，吕武玲，陈伟，许晓晨，刘建，施亮等同学，正是由于他们的相伴，才使得原本枯燥的实验室生活变得其乐融融，在科研之余多了些欢声笑语。还要感谢已经毕业的实验室师兄师姐们，像蒋业逢，刘峰，林炼，余宾，杨剑萍等，正是由于他们的指引才使我渐渐融入到实验室这个集体中，为我们这些师弟师妹们树立了榜样。还要感谢一些在生活中志同道合的朋友，正是从他们身上我看到了自己的许多不足，从他们身上我得到了朋友般的关怀，使我敢于面对脆弱，乐观处世，以平和的心态看待得失。

感谢所有本文引用的参考文献的作者，他们的大量前期探索为本文研究内容奠定了扎实的基础，使得本文能够在更高的层次上开展工作。

一如过去的二十多年，我的父母无私地给予我关心，照顾和支持，与我共同分享这三年的喜怒哀乐。家是休憩的港湾，每一次遭受挫折或是获得成就之后，我都能从家里汲取力量再次启航。感谢他们对我永无止境的关爱！