摘要

随着 Internet 的飞速发展和多媒体技术的不断成熟，流媒体应用已经成为互联网上最为重要、最具活力的应用之一。流媒体传输需要网络高带宽的支持，传统客户/服务器架构可扩展性差，无法满足大规模流媒体应用要求。近年来 P2P 技术得到迅速发展，与客户/服务器模式不同，P2P 网络中单个用户既是客户节点，也是服务器节点。P2P 技术利用了闲置在网络边缘的用户资源，提高了系统的可扩展性，使得网络技术向更大规模发展。P2P 技术的兴起给大规模的流媒体应用带来了新的解决方案——P2P 流媒体。

本文首先介绍了 P2P 流媒体出现的背景，特点和相关技术。P2P 分布式特点极大地提高了系统的可扩展性，但也带来许多挑战性问题，如节点动态性管理，数据调度机制，公平性和安全问题等。第二章介绍了 P2P 相关技术，包括数据编码机制、激励机制和网络测量技术。第三章引出我们需要解决的问题：如何设计高效的数据调度算法满足流媒体传输需求。我们介绍了已有的研究，结合实际的流媒体系统深入分析了各种数据调度算法的优缺点。

论文第四章提出了基于优先级的数据调度方案 DPC。DPC 是一种分布式的“拉”数据调度算法，由发送端和接收端调度两部分组成，在对节点和数据块划分优先级后，发送节点优先响应优先级高的邻居节点请求，接收节点优先请求优先级高的数据块。这种严格按照优先级的数据传输策略保证了数据分发的几何级数增长，从而最小化数据传输延时。但是严格的优先级传输策略需要节点之间交互大量控制消息，在深入分析 DPC 算法消息延时开销基础上，进一步提出了同构环境下基于“推”的 DPC 算法改进方案。改进方案是基于发送端“推”数据分发方式，数据发送完全由发送端控制，大大减少了消息延时开销。

与实时流媒体不同，P2P 点播系统中 VCR 操作和播放异步降低了节点之间的合作，增加了服务器负载，影响了系统的可扩展性。第五章首先讨论了 P2P 点播系统中服务器负载因素，在此基础上提出了基于预测的带宽分配方案 PBA，PBA 算法基本思想是将数据分发转化为带宽分配问题，利用额外带宽帮助落后节点预取数据块，额外带宽的分配是根据节点的稳定性和播放位置。利用额外带宽预取数据降低了节点之间拥有数据块的差异，从而减少离开丢失引起的稀有数据丢失，减少了服务器负载。

关键字：P2P，流媒体，数据调度机制

Abstract

With the widespread deployment of broadband access, multimedia services are getting more and more popular, and have became a significant application in today's Internet. Multimedia transmission needs the support of broadband access, due to poor scalability, traditional Client/Server model is not appropriate when facing growing users of multimedia services. Comparing with Client/Server model, in P2P each peer acts both as server and client. The whole system could better utilize resources from all over the network, thus greatly improve its scalability. P2P technology provides a new solution for large-scale multimedia services.

In this thesis we first introduce the background of P2P technology, features and related research issues. Although the decentralized feature of P2P greatly improve system scalability, it also brings about some challenging problems, such as peer dynamics, data scheduling mechanism, fairness and security issues. The second chapter discusses related work in P2P area, including data coding, network measurement and incentive mechanisms. The third chapter introduces one research direction: how to design an efficient data scheduling algorithm to meet the needs of multimedia transmission. Then we put data scheduling algorithms into specific P2P streaming systems to analyze their cons and pros.

In chapter four we proposed a distributed priority-based data scheduling algorithm called DPC. DPC is essentially a pull-based mechanism. It contains two parts: supplier side scheduler and receiver side scheduler. Each peer and data chunk is labeled with priority. Each supplier would choose peers to serve based on their priority, and each receiver would request data chunks in the same way. These strict rules guarantee the geometric distribution of data chunks, and thus minimize the delay for data distribution. However, the strict priority also introduces much message overhead. After analyzing the message transmission delay, we further propose a push-based DPC algorithm. The revised algorithm is pull-based scheme. The suppliers take full responsibility for data chunk distribution, and thus there is much less message exchange between peers.

Unlike live streaming, VCR operation and playback asynchrony in P2P VOD systems greatly dilute peers' ability to help each other and offload the server. In chapter five we first discuss the major causes of server load in P2P VOD system, and then propose a predicted-based bandwidth allocation algorithm called PBA. The basic

idea of PBA is that data scheduling can be equally treated as bandwidth allocation. Each peer allocates extra bandwidth to help descent peers pre-fetch data chunks. A stable peer with higher playback position is able to obtain a larger share of extra bandwidth. In this way, the differences in the amount of data chunks downloaded by different peers would decrease, and thus reduce the server load caused by departure misses.

Keywords: Peer-to-Peer, media streaming, data scheduling algorithm

中国科学技术大学学位论文原创性和授权使用声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

本人授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

保密的学位论文在解密后也遵守此规定。

作者签名：张正军

2009年5月5日

第一章 绪论

§1.1 引言

随着 Internet 网络的飞速发展和多媒体技术的不断成熟，互联网上传输的内容已逐渐由简单的文本、图像转变为包含文本、音频、视频的多媒体数据。相比传统的文本和图像数据，即使采用了编码压缩技术，音视频文件数据量仍然相当大，同时音视频数据传输对传输延迟敏感，因此流媒体服务需要消耗更多的带宽。随着因特网的快速发展，用户接入带宽的不断增大，以及音视频压缩技术的逐渐成熟，这些都为流媒体的实际应用提供了强大的动力。

在线音乐、视频直播和点播等因特网流媒体业务进入实用阶段。根据中国互联网信息中心第二十二次中国互联网发展状况统计报告，网络音乐是中国网民第一大网络应用，使用率达到 84.5%，用户量高达 2.14 亿人。网络视频也是中国网民的重要互联网娱乐方式。中国流媒体直播技术的发展非常迅速，目前国内已有超过 10 家网站采用自行研发的软件提供 P2P 流媒体业务，包括 PPLive[1]、PPstream[2]、QQLive[3]、UUsee[4]等。但是现有 Internet 上构建的大规模流媒体系统始终面临着一些挑战：网络服务质量(QoS)问题，节点异构性和系统扩展性问题。由于流媒体应用对网络带宽、丢包、延迟抖动等服务质量都有严格要求，目前的 Internet 难以提供 QoS 保证，这使得流媒体的大规模商业应用难以满足用户对服务质量的需求。另外网络和客户端通常具有较大的异构性，满足具有不同接入速度用户的不同服务质量需求，也是需要解决的问题。可扩展性问题主要表现在很多流媒体系统在用户人数不多的情况下能够提供满意的服务，但是当用户数达到一定数量，其服务质量就严重下降。如何解决大规模流媒体应用中的可扩展性问题，一直是工业界和学术界关注的重点。

为了有效地解决上述挑战，大量新技术、新架构被提出，其中网络传输技术是构建大规模流媒体系统的关键技术。目前已有的流媒体传输技术可以分为客户/服务器模式、IP 组播、CDN 内容分发技术和 P2P 技术支持的流媒体方案。

§1.2 流媒体传输技术的发展

§1.2.1 客户/服务器模式

客户/服务器(Client/Server)模式中每个客户端使用单播方式与服务器建立连

接，直接从服务器获取数据，如图 1-1 所示。这种模式优点是技术成熟，实现简单，适合于视频点播等需要“单点到单点”数据传输的流媒体应用，但是并不适合于存在大规模并发用户的网络直播应用。这是因为大量并发用户不仅可能导致服务器负荷过重，形成资源瓶颈，而且会造成大量相同音/视频数据在骨干链路重复传输的情况，从而导致网络资源的极大浪费。为此，S. Deering[5]等人提出了基于 IP 组播的传输模式。

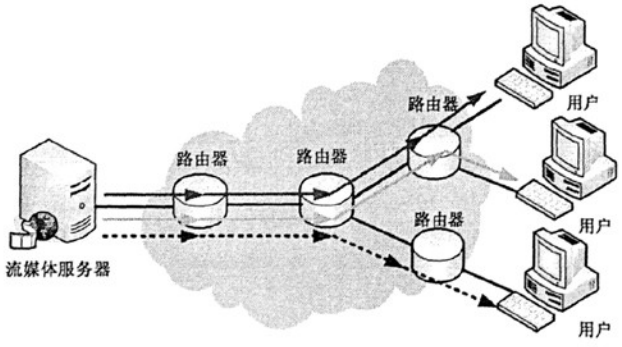


图 1-1 客户/服务器模式示意图

§1.2.2 IP 组播

在 IP 组播中，接收相同内容的用户构成一个组播组，由路由器负责维护组播组的状态和组成员的变化，并在组成员之间构造一棵数据分发树。流媒体服务器只需要向组内发送一份组播数据，由路由器在恰当的分支点复制、转发数据，就可让所有组成员收到数据，而任何一份数据包的拷贝只会在组播树的每条链路上出现一次。由于 IP 组播能够有效利用网络资源并降低服务器的负荷，因此它被认为是解决“单点到多点（One-to-Many）”数据传输最理想的方式。然而 IP 组播并没有得到广泛应用，这是因为 IP 组播需要路由器的支持，组播路由器需要维护每个组播组状态，增加了组播路由器的负载。

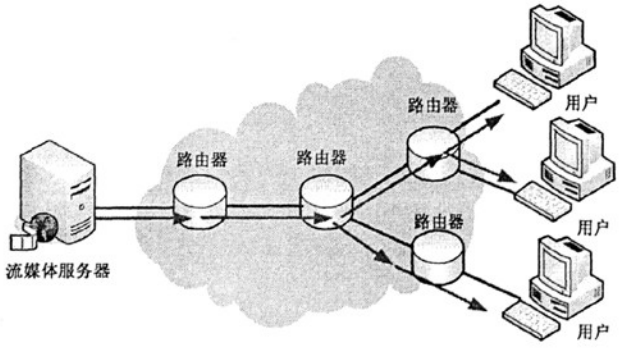


图 1-2 IP 组播模式示意图

§1.2.3 CDN 方式

内容分发网络(Content Delivery Networks)[6]是一种基于分布式 C/S 结构的流媒体服务平台，CDN 主要机制是通过在多个 ISP 接入处布置多个层次的缓存服务器节点，通过智能化策略，一方面通过全局内容复制策略把内容推送到各个边缘服务器，将中心服务器流媒体内容分发到这些距离最近，服务质量最好的节点，同时通过全局负载均衡技术把用户引导到最近的边缘服务器，用户可以就近取得所需流媒体内容，从而实现负载分散，减少了主干网络流量，提高用户访问流媒体服务的响应速度。但是 CDN 网络构造成本高，可扩展性差，不能适应快速增长的用户需求。

§1.2.4 P2P 流媒体技术

不同于传统的 C/S 网络架构，P2P(peer-to-peer)架构中网络中每个节点都是逻辑对等的，网络中节点既是服务器又是客户端。由于充分利用了参与节点的闲散资源，解决了单一服务器资源瓶颈问题。P2P 流媒体系统就是通过 P2P 架构在节点间传输流媒体服务的系统。在 P2P 流媒体系统中，对相同视频流感兴趣的节点协同合作传输数据，数据转发功能从 IP 层转移到应用层。与 IP 组播方案相比，不需要中间路由器的支持，易于部署；与 CDN 方案相比，由于用户参与数据分发，P2P 架构极大地提供了系统可扩展性。图 1-3 是 P2P 流媒体数据分发示意图。

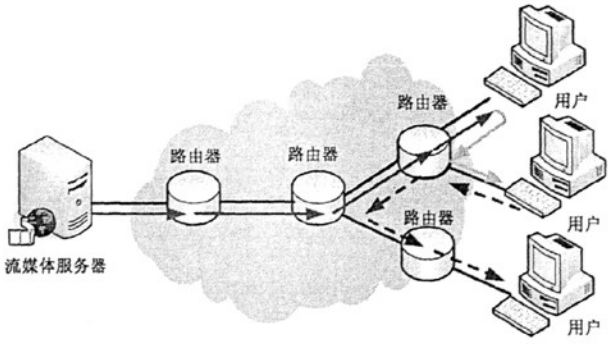


图 1-3 P2P 流媒体技术示意图

P2P 具有的良好特性使基于 P2P 技术的流媒体成为近年来研究热点，国际上众多知名大学和研究所纷纷投入了该领域的研究，并取得了一系列的研究进展，如文件共享系统 Bittorrent[7]，广泛应用的实时流媒体系统 CoolStreaming[8]和 PPLive，新兴的视频点播系统 GridCast[9]等。

§1.3 P2P 流媒体技术研究的关键问题

P2P 流媒体系统中节点具有相同的地位，这种非中心化特点带来了其在可扩展性、健壮性方面的优势，同时也引入了许多新的挑战。经过近几年的研究和发展，P2P 流媒体技术得到了长足的发展，但是仍然有许多待解决的挑战性难题。我们归纳研究的关键问题主要有以下几点：

(1)节点动态性管理

P2P 系统中节点的行为是不可预测的，每个节点可以随时加入系统请求服务，也会随时终止服务，甚至在没有预先通知下退出系统。极端情况下，大量节点在短时间内加入或者退出网络。节点的高度动态性对整个系统的服务质量带来了挑战。

(2)节点异构性

网络中节点的能力是异构的，它们的上下行带宽，可提供的存储空间和计算能力是不同的。高质量视频流媒体需要足够的带宽支持，由于节点带宽的异构性，一些节点可用带宽低于视频速率的要求。进一步提高流媒体视频速率，满足高带宽节点对视频质量需求，帮助低带宽节点获取满意的服务质量是流媒体系统需要研究的一个关键问题。

(3)数据调度算法

P2P 流媒体中每个节点既是服务提供者又是接收者，节点之间相互合作获取共同感兴趣的视频数据。满足流媒体视频数据播放的连续性是数据调度算法的基本要求。除了连续性要求外，流媒体传输还需要考虑视频播放的启动延时和播放延时。启动延时是指节点从选定视频到实际播放需要等待的时间，播放延时是指收看相同视频节点之间的播放位置差。如何高效地分发数据，充分利用系统节点带宽是流媒体数据调度算法设计的关键。

(4)节点组成员管理

节点组成员管理是将参与系统中的节点有效组织起来形成逻辑上的覆盖网络，除了需要考虑节点的高度动态性，将覆盖网络时与实际的底层拓扑相结合是 P2P 流媒体系统的一个关键，考虑了底层拓扑的流媒体系统可以对节点进行聚类，这样做的好处是缩短数据传输路径，降低了数据传输延时，减少跨 ISP 流量。

(5)公平性和安全问题

由于 P2P 网络的去中心化特征，单个节点未必会积极的参与数据传输服务，即所谓的“搭便车”现象，这使得分散在客户端的资源无法得到充分利用。激励机制目的是鼓励节点贡献带宽参与数据传输。设计有效的激励机制是 P2P 流媒

体系统需要解决的关键问题。除了“搭便车”节点以外，P2P 系统中还可能存在恶意节点。去中心化特点使得恶意节点发布的虚假信息或者恶意代码可以迅速充斥网络，在 P2P 系统中加入安全机制是一个关键研究问题。

§1.4 本章小结

本章概括性介绍了 P2P 流媒体系统出现的背景、特点和需要解决的关键问题。流媒体传输需要网络高带宽的支持，传统客户/服务器架构可扩展性差，存在单一服务器资源瓶颈问题。P2P 网络架构充分利用了网络边缘分布的大量计算和存储资源，节点在网络中既是服务提供者又是接收者。P2P 架构去中心化特征极大地提高了系统的可扩展性，同时也带来了许多挑战性问题，如节点的有效管理，数据的高效分发，公平性和安全问题等。

§1.5 论文结构

第一章介绍了 P2P 流媒体系统出现的背景、特点和需要解决的关键问题。

第二章详细阐述了 P2P 流媒体的相关技术，包括视频编码技术、激励机制、网络测量和 P2P 分布式协同攻击。

第三章分析 P2P 流媒体系统节点组成员管理和数据调度机制。在此基础之上详细分析了典型的 P2P 实时流媒体和点播系统。

第四章提出了基于优先级的数据调度算法 DPC，DPC 算法由发送端和接收端调度两部分组成，在对节点和数据块划分优先级后，发送节点优先响应优先级高的邻居节点请求，接收节点优先请求优先级高的数据块。这种严格按照优先级的数据传输策略保证了数据分发的几何级数增长，从而最小化数据传输延时。但是严格的优先级传输策略需要节点之间交互大量控制消息，在深入分析了 DPC 算法消息延时开销，进一步提出了同构环境下基于“推”的 DPC 算法改进方案。改进方案中数据发送完全由发送端控制，大大减少了消息延时开销。

第五章深入分析了 P2P 点播系统中服务器负载因素，在此基础上提出了基于预测的带宽分配方案 PBA，PBA 算法将数据分发等价为带宽分配问题，利用额外带宽帮助落后节点预取数据块。利用额外带宽预取数据降低了节点之间的数据差异，降低了离开丢失引起的服务器负载。

第六章总结全文，并指出下一步的研究方向。

第二章 P2P 流媒体相关技术

§2.1 数据编码技术

传统的视频编码方案是面向存储的，其目标是将视频压缩成一个或几个固定速率的码流，然而网络的异构性和缺乏 QoS 保证，因此传统的固定速率的编码方案并不适合网络传输。在采用 IP 组播时，解决这个问题最简单的方法是最小速率传输，即组播系统的数据码率是由接入带宽最小的用户决定，这类似于“木桶效应”，具有高带宽接入的用户无法享受到与其带宽相匹配的视频质量。

§2.1.1 精细粒度可扩展编码

理想的视频编码方案具有可以在任何位置截断的特性，解码质量随之接收码率的增加而提高。为此研究人员提出了精细粒度可扩展编码方案(Fine Granular Scalable, FGS)[10][11]。FGS 编码的基本思想是将原始视频编码为一个基础层（Base Layer）码流和若干增强层（Enhancement Layers）码流，其中基础层包含视频流中最重要的信息，而增强层则“逐层递近”地包含更细粒度的冗余信息，用户只需接收基础层的码流就可获得基本的播放质量。在接收基础层码流的基础上，接收的增强层码流越多，解码器重构的图像质量也就越好。如图 2-1 所示，精细粒度可扩展编码的视频质量随着接收带宽的增大而逐渐增大，因此它能够更好地适应带宽的异构性和网络带宽的波动。

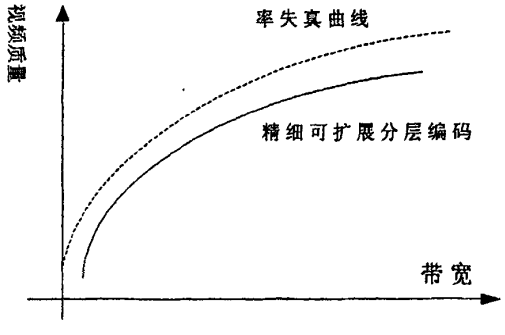


图 2-1 视频质量随着接收带宽增加而增加

将精细粒度可扩展编码与组播相结合是解决带宽异构性的办法，称为分层组播(layer multicast)。在分层组播中，节点从多个邻居节点处获取数据，优先请求基础层数据，保证基本的视频播放，然后根据邻居节点的缓存状况和带宽，请求

增强层数据。由于分层编码中的各个层的地位不是平等的，下层的优先级高于上层的优先级，只有下层的数据接收到了，上层的数据才可以被解码，如何有效地分配带宽获取多层数据是一个值得研究的问题[12]。图 2-2 是节点请求多层视频数据示意图。(a)是最保守的请求方式，节点从基础层开始逐层请求，数据层次优先，这种方案的是牺牲了视频质量来保证基本视频播放连续性；(b)是一种激进的请求方式，节点请求相应数据所有层数据，相比较(a)，(b)方案能够提高短期的视频接收质量，但牺牲了播放的连续性；(c)是一种折中方案，节点先请求一部分基础层数据，然后获取一部分增强层数据，这样就可以保证播放的连续性，又能够在一定程度上提高视频接收质量。

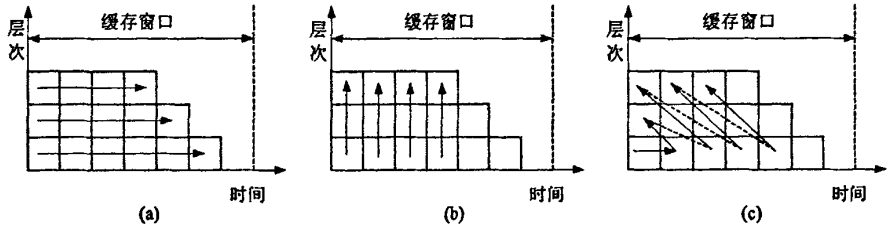


图 2-2 节点请求多层视频数据示意图

§2.1.2 多描述编码

多描述编码 MDC (Multiple Description Coding)[13][14]把视频数据编码成 M 个独立的流，称为描述。其中的任何一个描述都可以在一定的失真率下单独解码，不同描述流的叠加可以使得率失真降低。所以用户接收到的描述流的个数越多，重构出来的视频质量就越高。MDC 与分层编码的不同之处在于 MDC 中的 M 个描述流的地位是平等的，它们中的任意子集都可以解码。而分层编码中的各个层的地位不是平等的，下层的优先级高于上层的优先级，只有下层的数据接收到了，上层的数据才可以被解码。

没有分层编码层之间优先级关系的限制，MDC 编码中各个独立的描述流相比分层编码更适合组播应用。Splitstream[15]系统采用了多棵树的覆盖网络结构，每棵数据转发树上分发一个 MDC 描述流，用户可以加入到多棵组播树获取数据，每个节点允许加入的组播树数量是由其贡献带宽决定的，节点在系统中贡献越大，就可以加入到越多的组播树中，因而获取更高的视频接收质量。如果多棵树结构采用分层编码，每棵组播树对应着一层视频流。由于 P2P 网络中节点的动态性，当组播树中的内部节点退出网络时，其下游节点将会有数据丢失，如果退出的节点正在分发基础层数据，那么下游节点将不能解码获取的高层数据，从而造成了视频播放的不连续性和带宽的浪费。相比之下，MDC 编码视频流之间是

平等的，尽管内部节点的退出造成了数据的丢失，其下游节点从其他树中获取的数据依然可以解码，内部节点的退出对其下游节点的影响只是降低了视频质量。图 2-3 是 Splitstream 系统中多棵树覆盖网络示意图。节点 1，2 和 3 承担 MDC 1 流的数据转发任务，在其余两棵树中是叶子节点，获取 MDC 2 和 MDC 3 视频流。

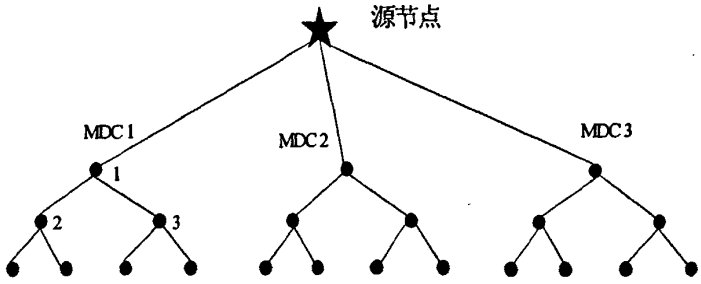


图 2-3 多棵树覆盖网络分发 MDC 描述流

相比传统的单一视频流，分层和多描述编码允许任意的切割视频流，能够很好地支持网络的异构性。但是为了协调不同视频子流，这些高级的编码方式在原始视频数据中增加了足够的冗余信息，相比单一视频流，一定程度上带来了带宽的浪费，另外它们的解码计算开销大，这些缺点都限制了它们的实际应用。

§2.1.3 网络编码

网络编码(network coding)[16]是一种融合编码和路由的信息交换技术，在传统存储转发的路由方法基础上，通过允许中间节点对接收到的多个数据包进行编码信息融合，增加单次传输的信息量，从而提高整个网络的性能。以著名的“蝴蝶网络”模型为例，阐述了网络编码的基本原理。图 2-4 所示的“单信源二信宿”蝴蝶网络，设各链路容量为 1，S 是信源节点，Y 和 Z 是信宿节点，其余为中间节点。根据“最大流最小割”定理，该多播的理论传输容量为 2，即理论上信宿 Y 和 Z 能够同时收到信源 S 发出的 2 个单位信息，也就是说能够同时收到 a 和 b。图(a)表示的是传统的路由方式，节点 W 单纯地执行存储和转发操作。假定 W 转发信息 b，则链路 WX、XY 和 XZ 上传输的信息均为 b，信宿 Y 能够同时收到 a 和 b，但信宿 Z 只能收到 b，同时收到一个多余的 b，因此，信宿 Y 和 Z 无法同时收到 a 和 b，该多播不能实现最大传输容量。

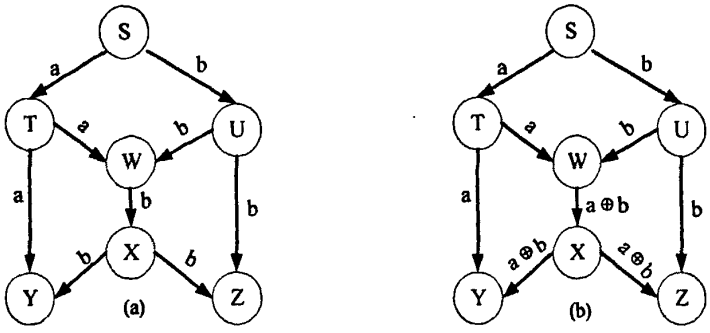


图 2-4 路由方法与网络编码比较：(a) 传统路由方法，(b) 网络编码

图(b)表示的是网络编码方案，节点 W 对输入的信息进行编码操作，如异或操作，然后将编码结果 a⊕b 发送到输出链路 WX，然后又通过链路 XY 和 XZ 最终到达信宿 Y 和 Z。Y 将收到的 a 和 a⊕b 进行译码操作，就能解出 b，同理，信宿 Z 也能解出 a，因此 Y 和 Z 能够同时收到 2 个信息，从而实现了多播理论上的最大传输容量。由此可见，网络编码的核心思想是：具备编码条件的网络节点对接收到的信息进行一定方式的处理，然后传输给下一级的节点，收到消息的下一级节点如果具备编码条件，又可以对接收到的信息进行再编码，如此反复，直到所有经过处理后的信息都汇聚到信宿节点为止。最后，在信宿节点，通过译码就可以译出信源发送的原始信息。从图论的角度，入度至少为 2 的节点具备编码条件。

网络编码虽然起源于多播传输，主要是为解决多播传输中的最大流问题，但是随之研究的深入，网络编码与其他技术的结合越来越受到人们的关注，与 P2P 技术结合是当前的一个研究热点。Microsoft 开发的 Avalanche[17]第一次将网络编码应用到 P2P 文件共享系统中，实验表明如果所有的节点都运用网络编码，则 P2P 系统的平均下载速率比仅仅服务器端进行编码提高了 20%-30%，比不用网络编码提高了 2-3 倍。另外，网络编码实际上提高了系统中数据冗余，因此它能够很好的解决 P2P 文件共享中由于节点离开导致的“稀有块”问题，更能适应 P2P 网络节点高度动态性。图 2-5 是 Avalanche 的工作原理，假设服务器需要传输信息给节点 A，则首先将服务器上的信息分解成 n 个数据块，$B_1, B_2, B_3, \ldots B_n$，然后随机选择编码系数 $c_1, c_2, \ldots, c_n$，将线性网络编码后的信息 $E1= c_1B_1 + c_2B_2+\ldots+c_nB_n$ 传输给节点 A。同时节点 A 从其他处获取另外的信息 $E2= c'_1B_1 + c'_2B_2+\ldots+c'_nB_n$，节点 A 随机选择系数对获取的信息进行再编码操作，将编码后结果 $E_3=d_1E_1+d_2E_2$ 发送给节点 B，依次类推，只要每个节点收集到足够的信息，就可以通过解线性方程译出原始信息。

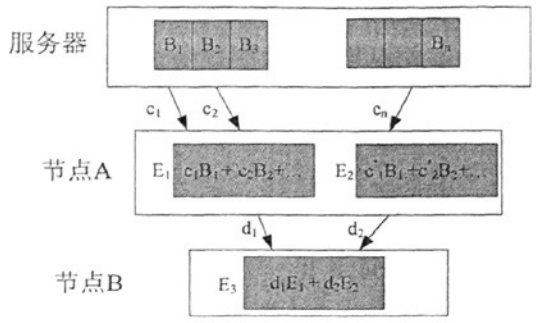


图 2-5 Avalanche 工作原理

网络编码可以提高系统中数据冗余，能够很好地解决 P2P 文件共享系统中的“稀有块”问题。对流媒体系统来说，引入网络编码提高数据冗余可以有效地解决“内容瓶颈”现象，提高节点的带宽利用率。但流媒体应用中数据实时性要求高，传统的视频编码，节点收到单个数据块即可解码，实时性高，网络编码必须收集到足够的信息才能够解码，降低了解码实时性。由于实际网络中不同路径上传输的分组通常会经历各异的传播和排队延迟，分组的异步传输将极大地增大节点的存储开销，并影响解码速度。因此，网络编码并不能简单的扩展应用到流媒体系统。已有的将网络编码应用到实时流媒体方案中，一般都采用了一种基于“代”(generation)的缓存模型[18]，源节点将同一组原始数据块归为一“代”，中继节点以“代”为索引存储接收编码分组并对属于同一代的分组进行编码操作，该方案可以有效地减低网络异步传输影响，提高了数据实时性。最后，采用了网络编码的系统中，源节点和中继节点需要进行编码操作，随着数据在网络中的传输，编码开销逐渐增加，同时解码开销也逐渐增加，因此优化编解码速率也是网络编码能够应用到流媒体系统的急需解决的问题。

§2.2 P2P 网络激励机制

与传统的集中式的客户/服务器模式不同，基于 P2P 的系统是去中心化的，服务器的分发功能被转移到普通参与节点，节点不仅是数据的接收者，又是数据提供者，向系统中邻居节点提供服务。因此，节点是否积极参与数据分发直接影响了 P2P 流媒体系统性能。

在 P2P 网络发展早期，激励机制并未引起足够的重视，绝大多数系统并没有将其包含在系统设计中，随着 P2P 网络系统的应用越来越广泛，系统规模越来越大，缺乏激励机制所导致的问题也越来越严重。E.Adar[19]等人最早在研究中指出 P2P 网络中存在大量的“搭便车”(free-rider)节点， Gnutella 系统中 70%的用户都只是从系统获取自己需要的文件而不向系统提供共享文件，而系统中

50%的共享文件来自 1%的节点。在实时性和带宽需求更高的流媒体应用中，节点的不合作将严重影响系统的性能。因此，为 P2P 流媒体应用设计激励机制成为当前的研究热点，目前已有的激励方式主要有两类：虚拟支付和自然激励。

§2.2.1 虚拟支付

为了量化节点的贡献带宽，直观的做法是引入虚拟计量单位，如虚拟货币，积分或者信誉。节点贡献一定量的带宽换取相应的货币或者积分，节点利用获得的货币或者积分“购买”需要的数据或者服务质量。Habib[20]等设计了一个基于积分的鼓励用户贡献转发资源的激励机制。其主要思想是：记录每个节点给其他节点转发数据的贡献并将其对应于一定的积分值，又把积分值换算成优先级，优先级被应用在每个节点选择邻居节点的过程中。规定每个节点都可以选择优先级与自己相当或比自己低的节点为自己提供数据。因此，贡献分值越高的节点，可选择的邻居节点范围越广，能够选择到更好邻居节点的机会也就越大。这样，做转发贡献较多的节点就可以获得较好的服务质量。图 2-6 表示了带宽换积分的示意图。

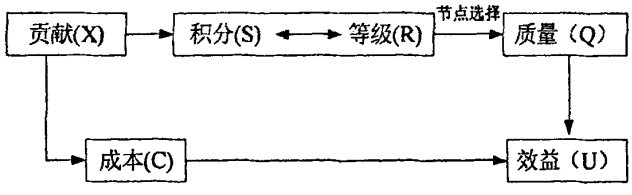


图 2-6 带宽转换积分的示意图

Tan[21]等人提出了一种基于拍卖的鼓励用户贡献带宽的激励方式，其主要思想是：每个节点通过向其他节点转发数据赚取积分，贡献的越多，获得积分越多。流媒体数据发送分为若干个周期，每个周期内节点请求一个数据段并且接收其他节点数据段请求。数据段的请求和发送采用“市场”机制，即在每个数据调度周期初始，每个请求节点分别向数据提供节点出价，而在数据提供节点的带宽允许范围内，那些出价较高的请求节点将赢得此次数据段的发送，即“最高竞价”策略。因此，贡献分值越高的节点，越有可能在竞争中获胜，选择延时小数据提供节点，从而获得较好的服务质量。可以说每次数据调度重新构造了覆盖网络结构，图 2-7 表示了基于拍卖的覆盖网络的构建过程。图(a)表示 1-6 共六个节点向节点 0 竞价，节点 0 的外出带宽为 2，只能接收 2 个出价最高的节点；图(b)表示节点 5 和 3 成为节点 0 的直接子节点，同时节点 0 把竞价结果通知其他落选节点；图(c)表示新一轮的竞价，落选节点向节点 0 的直接子节点 5 和 3 出价。

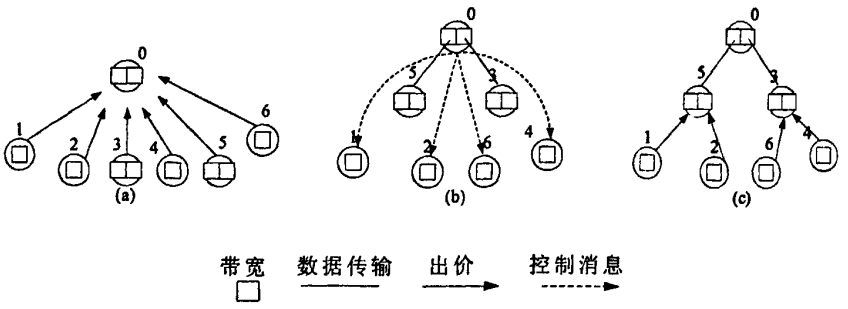


图 2-7 基于拍卖的覆盖网络构建

基于计费机制的虚拟支付是一种有效的激励方案，但其缺点也很明显。为了有效地量化节点的贡献，计算节点的虚拟货币和积分，需要集中式的中心服务器。因此，这种激励方式普遍存在可扩展性问题。另外除了“搭便车”节点，P2P 网络中还存在恶意节点，恶意节点相互勾结向服务器谎报自己的贡献赚取虚拟货币或者积分。如何有效及时地识别恶意节点并将其隔离出 P2P 网络，也是一个重要的研究问题。

§2.2.2 自然激励

自然激励方式是指节点的贡献与自己实际的接收状况相匹配，由自己的接收状态决定贡献。最成功的自然激励是文件共享 Bittorrent 系统采用的“tit-for-tat”策略，其主要思想是：节点将自己拥有的数据传输给固定数量的请求下载节点，这些被允许下载的节点是过去一段时间内提供给它最大下载速率的节点。从博弈论的角度，“tit-for-tat”是囚徒困境博弈中的一个重要策略，即依据对手的策略选择自己的优势策略，因此，“tit-for-tat”可以有效地解决恶意节点勾结欺骗问题。“tit-for-tat”要求节点之间存在双向数据流，也就是节点之间各自拥有对方感兴趣的数据，但是在基于“推”数据分发流媒体系统中，数据流从源节点自顶向下分发，节点之间没有双向数据流，在基于“拉”数据分发流媒体系统，由于实时流媒体系统中节点播放是半同步的，P2P 流媒体系统普遍存在“内容瓶颈”现象，即尽管邻居节点存在剩余带宽，邻居节点没有节点需要的数据。因而流媒体系统中节点之间的双向数据流并不明显。“tit-for-tat”策略并不能简单地扩展到流媒体系统。

为了提高节点之间双向数据流量，Liu[22]等人提出了基于分层编码的实时流媒体激励方案，其主要思想是：由于分层编码的引入，节点之间分布不同层的数据，极大地降低了“内容瓶颈”的可能性，提高了节点之间双向数据流量，在此基础上，“tit-for-tat”用来对节点进行区分服务，上传带宽大并且愿意承担转发任

务的节点，可以从邻居节点处获取更多层的视频数据，改善自己的视频质量。Mol[23]等人为 P2P 视频点播系统设计了激励方案——"Give-to-Get"，与"tit-for-tat"不同，"Give-to-Get"不依赖节点之间的双向数据流，每个调度周期节点根据孙子节点的反馈评估直接子节点的转发性能，允许转发性能最好的固定数量的直接子节点在下一个调度周期继续向本节点请求数据。图 2-8 说明了"Give-to-Get"算法中的反馈和评估过程。C 是节点 P 的直接子节点，节点 G 既是 C 的直接子节点，又是节点 P 的孙子节点。节点 P 根据孙子节点 G 的反馈信息依据以下标准对直接子节点 C 进行评估：1）节点 C 向其子节点 G 传输的数据总量；2）节点 C 向子节点 G 传输来自节点 P 数据的数据量。

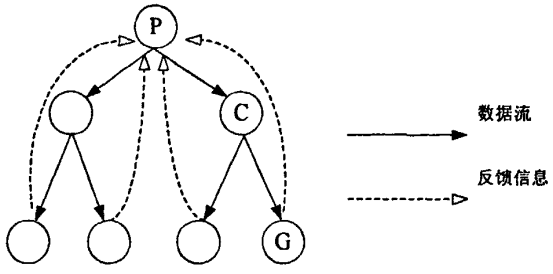


图 2-8 "Give-to-Get"算法反馈过程

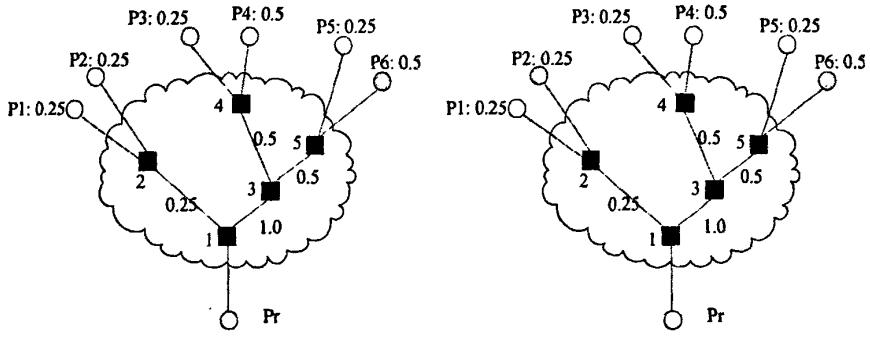
由于流媒体中节点播放的半同步性，节点之间双向数据流量小，因此"tit-for-tat"策略并不能直接应用到流媒体系统。研究人员引入了分层编码等高级视频编码格式增加节点的数据分布多样性，从而提高双向数据流量；但是高级视频编码技术目前还不够成熟，还不能应用到 P2P 流媒体系统中。"Give-to-Get"算法虽然绕开了双向数据流量的问题，但是依赖于节点反馈信息，增加了控制消息开销，同时它也无法解决可能存在的恶意节点相互勾结。

§2.3 网络测量技术

测量技术在 P2P 流媒体应用中占有重要的一席之地，主要原因有以下两个方面：第一，P2P 流媒体构建覆盖网络，选择邻居节点需要测量节点的性能，每个节点都希望与延时小、带宽大的邻居节点交换数据，从而减小数据传输延时，提高视频接收质量，这些参数需要通过网络测量得到。这个因素可以看成是对 P2P 覆盖网络的优化。第二，P2P 应用产生的流量在 Internet 上所占的比例已经超过了传统的 Web 应用，占用了大量的网络带宽，影响着其他业务的服务质量。对 P2P 流量进行测量控制和优化是 P2P 应用能够继续发展的前提。可以说这两个因素是相辅相成的，对覆盖网络的优化不仅仅提高了系统的性能，也可以实现对 P2P 流量控制和优化；对 P2P 流量的测量可以更好地指导我们对 P2P 拓扑结构的优化。

§2.3.1 覆盖网络优化

为了有效地实现数据分发，P2P 网络首先需要将参与会话的节点组织起来。由于节点的异构性和网络分布特点，不同的节点组织方法对流媒体性能有着重要的影响。图 2-9 说明了基于网络测量的节点选择对流媒体传输的意义。云形图中蓝色方块表示路由器，路由器之间存在着共享链路，链路是由容量限制的，节点旁边标注的数字表示可以提供的速率。接收节点 Pr 随机选择 P3，P5 和 P6 作为邻居节点请求数据，如图 2-9(a)所示。尽管这三个节点能够共同提供 1 的视频速率，但是由于路由器 3 和 5 之间的链路只能够传输 0.5 倍的视频速率，节点 P5 和 P6 共享这一段链路，这将会导致链路拥塞，接收节点 Pr 无法获得满意的视频质量。在基于网络测量的节点选择中，接收节点根据底层测量获取的信息重新选择节点 P2，P3 和 P6。这样的选择避免了底层的拥塞链路，从而获得 1 的视频速率。



(a) 随机选择 (b) 基于网络测量

图 2-9 基于网络测量的节点选择

因此，考虑节点邻近性和异构性，优化覆盖网络时使其与实际的底层拓扑相结合是覆盖网络构建的关键问题，我们将在第三章进一步讨论这个问题。考虑了节点邻近性的覆盖网络，一方面可以降低数据的传播延时，降低了拥塞链路存在的可能性；另一方面可以减少跨 ISP 流量，使 P2P 流媒体能够更好的得到网络运营商的支持。支持节点异构性感知的覆盖网络可以对节点进行聚类，对不同接入带宽的节点实现区分服务，满足不同用户对服务质量的需求。

§2.3.2 P2P 流量测量

P2P 流量测量按照测量方法的不同可以分为主动测量和被动测量两种。主动测量方法主要通过设计网络爬虫主动加入网络，通过监听 P2P 应用的控制消息，

获取相关的拓扑属性和用户属性。被动测量方法是通过在网络上部署监测点，监测相关的 P2P 流量信息。在 P2P 流媒体方面，文献[37]对实际部署的流媒体系统 PPLive 进行了被动测量，全面揭示了 P2P 实时流媒体的用户行为特征（动态性和地理位置分布）、用户观看体验（包括视频播放的连续性，用户的启动延时和播放延时）、覆盖网络拓扑特点以及 P2P 域间流量特点。文献[41]率先根据系统运行数据对实际部署的 P2P 点播系统进行测量分析，说明了点播系统中用户的行为特征（节点的动态性和 VCR 操作）、用户的满意度（VCR 操作延时、播放质量和连续性）以及各种系统参数对性能的影响。

实际系统的测量表明，P2P 自组织特性导致了大量的跨 ISP 域间流量，这也是 P2P 应用受到 ISP 提供商抵制的重要因素。优化 P2P 节点组织方式，基于底层物理网络信息构建覆盖网络，对 P2P 流量进行控制和优化是一个重要的研究课题。

§2.4 P2P 分布式协同攻击

P2P 网络的分布式特征为蠕虫、僵尸等恶意代码传播提供了一个理想场所，在 P2P 环境下它们传播速度更快，覆盖面更广，防范和检测的难度也更大。基于 P2P 网络的分布式协同攻击主要分为两大类，一类是发掘现有 P2P 网络的协议缺陷进行攻击[24]，另一类则是自主构建 P2P 僵尸网络进行攻击。其中第一类攻击主要是通过虚假的索引和路由信息发动对某个特定目标的 DDoS 攻击。第二类攻击是攻击者通过控制傀儡机组成 P2P 僵尸网络[25][26]，然后发动 DDoS 攻击等。由于成型的 P2P 网络用户数量巨大，而且互联网上存在大量有漏洞的主机。一旦攻击者利用 P2P 网络或者构建 P2P 僵尸网络发动分布式协同攻击，都将会在短时间内汇聚大量攻击流，严重威胁了网络的正常运行，对受害者和社会造成很大的危害。

§2.4.1 索引和路由污染

这一类攻击主要是利用 P2P 组成员管理算法的安全缺陷。P2P 组成员管理算法主要有三种：1) 基于 DHT 的分布式管理方案；2) 基于集中式目录服务器的管理方案；3) 基于流言的管理方案。

在基于 DHT 的管理方案中，内容是通过内容索引(key, value)来标识的，key 是内容关键字，value 是内容的实际存放地址。索引污染是指恶意节点发布虚假的内容索引，将 value 修改为目标主机的 IP 地址，从而实现对目标主机的 DDoS 攻击。路由表污染是指恶意节点将目标主机加入到许多正常节点的邻居节点集

中。正常节点误以为目标主机是自己的邻居节点，因而与目标主机进行通信。当通信量足够大时，就实现了对目标主机的 DDoS 攻击。

在基于流言的管理方案中，DDoS 攻击的方式类似于 DHT 中的路由表攻击。恶意节点通知正常节点目标主机参与了 P2P 会话。正常节点将目标主机加入到邻居节点集中，并与之通信，当通信量足够大，就可以实现对目标主机的 DDoS 攻击。

基于集中式目录服务器的 P2P 系统，目录服务器负责所有节点信息维护，帮助节点选择邻居节点。恶意节点谎报目标主机是目录服务器，当大量正常节点与目标主机通信，就可以实现对目标主机的 DDoS 攻击。

§2.4.2 P2P 蠕虫和僵尸网络

网络蠕虫是一种智能化、自动化，综合网络攻击、密码学和计算机病毒技术，不需要计算机使用者干预就可运行的攻击程序或代码。它会扫描和攻击网络上存在系统漏洞的节点主机，通过局域网或者互联网从一个节点传播到另外一个节点。P2P 网络非常适合蠕虫传播的特征。传统的蠕虫的主要传播方式是随机扫描，而 P2P 网络中的蠕虫传播策略可分为两种：第一种是被动传播型，worm 附着文件实现传播或者利用正常的网络活动进行传播。第二种主动传播型，在 P2P 网络中，蠕虫只要感染其中一个节点，就可以通过节点的邻居节点列表探测到其他易攻击节点。P2P 蠕虫相比传统蠕虫具有以下特点：1) 由于不需要进行扫描探测，传播更迅速；2) 传播成功率很高；3) 隐藏在 P2P 流量中，具有更好的隐蔽性。

传统的僵尸网络采用了基于 IRC 协议的命令与控制机制，僵尸程序直接与 C&C 服务器进行通信，C&C 服务器充当控制主机与僵尸程序之间通信的桥梁。因此，C&C 服务器很容易成为传统僵尸网络的瓶颈。这使得传统僵尸网络容易被跟踪、检测和反制。为了使僵尸网络更具隐蔽性和韧性，僵尸网络中引入了基于 P2P 协议的命令与控制机制。相比传统的僵尸网络，P2P 僵尸网络中不存在只充当服务器角色的僵尸网络控制器，而是由 P2P 僵尸程序同时承担客户端和服务器的双重角色。图 2-10 和图 2-11 分别表示了传统僵尸网络模型和基于 P2P 方式构建的僵尸网络。

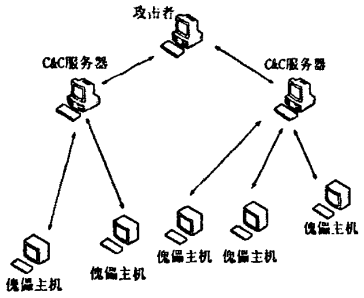


图 2-10 传统僵尸网络模型

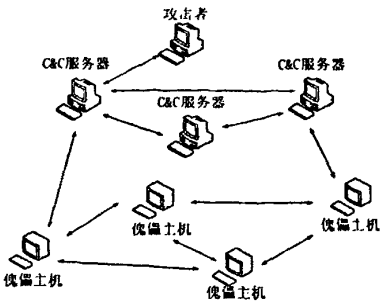


图 2-11 基于 P2P 方式构建的僵尸网络

§2.5 本章小结

本章详细介绍了 P2P 流媒体相关技术，包括数据编码技术、激励机制、网络测量和 P2P 分布式协同攻击。数据编码技术包括精细粒度可扩展编码、多描述编码和网络编码。精细粒度可扩展编码和多描述编码是改善视频数据编码格式以适应网络流媒体传输特点，视频数据被分解成多个码流，节点接收到的码流越多，解码视频质量越高，与传统单一码流的数据格式相比，很好地适应了用户接入带宽的异构性。源节点服务器对视频数据进行处理，中间转发节点只是负责存储转发功能。网络编码与这两个高级数据编码不同在于打破了传统通信观念，允许中间节点对数据进行再编码，利用节点的计算能力提高链路带宽的利用率，提高了组播系统的容量。

激励机制是为了解决 P2P 网络中的“搭便车”现象。已有的激励机制分为虚拟支付和自然激励，虚拟支付是指节点通过贡献带宽赚取积分或者虚拟货币，然后利用获得的货币或者积分“购买”需要的数据或者服务质量。自然激励是指节点根据自己接收状况决定贡献量，节点将自己拥有的数据传输给固定数量的请求下载节点，这些被允许下载的节点是过去一段时间内提供给它最大下载速率的节点。

网络测量在 P2P 流媒体中具有重要作用。P2P 流媒体构建覆盖网络，每个节点都希望与延时小、带宽大的邻居节点交换数据，减小数据传输延时，提高视频接收质量，为了选择性能优良的邻居节点需要测量技术支持。同时 P2P 应用产生的流量在 Internet 上所占的比例已经超过了传统的 Web 应用，占用了大量的网络带宽，影响着其他业务的服务质量。对 P2P 流量进行控制和优化是 P2P 应用能够继续发展的前提。

最后我们介绍了 P2P 分布式协同攻击。基于 P2P 网络的分布式协同攻击分为两大类，一类是发掘现有 P2P 网络组成员协议缺陷，通过发布虚假的索引和路由信息对某特定目标发起 DDoS 攻击。另一类则是自主构建 P2P 僵尸网络进

行攻击，攻击者控制大量的傀儡机，组成 P2P 僵尸网络进行 DDoS 攻击。针对虚假索引和路由信息，解决方案主要是完善 P2P 组成员管理方案，如引入验证机制判断路由信息的可靠性[27]，虚拟信誉机制等。针对 P2P 蠕虫和僵尸网络，已有的研究工作如蜜罐技术属于被动防御，主动探测提前发现潜在攻击者是当前的一个研究热点。

第三章 P2P 流媒体系统组成

近年来 P2P 流媒体技术得到了长足的发展，从最初的实时流媒体到新兴的点播系统，P2P 已经成为 Internet 上主要的流媒体分发技术。节点组成员管理和数据调度机制是 P2P 流媒体两个关键组成。节点组成员管理是将参与会话的节点有效的组织起来；数据调度机制解决了节点之间高效转发数据的问题。本章首先阐述 P2P 流媒体系统这两个关键组成，然后从这两个方面对现有实际部署的系统进行比较分析。

§3.1 节点组成员管理

P2P 网络中每个节点既是服务器，又是客户端，节点之间相互合作传输数据。将这些节点有效的组织起来是一个关键，节点组成员管理具体任务有：帮助新节点加入 P2P 网络，查找性能优良的节点替换失效的、性能较差的邻居节点等。目前常用的节点组成员管理机制包括：集中式、分布式和混合模式。

§3.1.1 集中式

集中式的节点组成员管理方案通常具有一个中心服务器，如 Bittorrent 中的 tracker 服务器，InfoTV 系统 YP 黄页服务器。Bittorrent 中的 tracker 服务器维护了拥有文件的有效节点的信息，InfoTV 中的 YP 服务器维护了正在收看同一频道的节点信息。因此，中心服务器的功能是维护参与会话的所有节点信息，包括节点的 IP 地址，端口号等。

当新节点加入到 P2P 网络时，首先与中心服务器进行联系，中心服务器的 IP 地址是众所周知的。中心服务器随机选择一部分节点返回给新节点，新节点与返回列表中节点建立连接，从而加入到 P2P 网络。图 3-1 是集中式组成员管理示意图。为了保证维护节点的有效性，节点周期性的向中心服务器发送存活消息。节点正常退出系统时，会向服务器发送离开消息，服务器收到消息后将删除相应的记录。在非正常退出的情况下，服务器在一段时间内没有收到节点的存活消息，相应节点信息也会被删除。

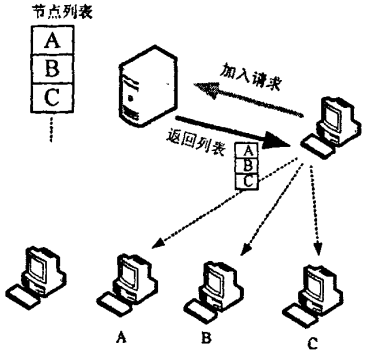


图 3-1 集中式组成员管理示意图

集中式组管理方式优点是实现部署简单。由于服务器维护了所有节点信息，易于对覆盖网络优化。但其缺点也很明显，中心服务器需要维护所有节点信息负载大，存在着“单点失效”的问题。

§3.1.2 分布式

分布式的组成员管理方式主要有基于流言(gossip)和 DHT(Distributed Hash Table, 分布式哈希表)方式两种。

基于流言组成员管理存在一个启动节点。与集中式的中心服务器不同，启动节点只维护了部分存活节点信息。在实际系统中，启动节点往往就是视频源节点。当新节点加入系统时，首先会与启动节点进行联系，启动节点将新节点的加入请求重定向到随机选择的代理节点，代理节点将自己维护的邻居节点列表返回给新加入节点，新节点向返回列表中的节点发起加入请求。图 3-2 表示新节点加入过程。为了保证邻居节点的有效性，节点之间采用流言协议[28]进行更新和维护状态。

基于 DHT 的组成员管理机制广泛应用在 P2P 文件共享系统中。在 DHT 中每个节点都有一个节点标识符，通常是节点 IP 地址的哈希值。内容索引被抽象为<Key, Value>对，其中 Key 是内容查询关键字，Value 是文件的实际存储位置。所有的<Key, Value >对组成一张大的哈希表。根据特定的规则，即内容关键字 Key 和节点 ID 之间的映射关系，哈希表被分割为许多不同的小块由不同的节点负责存储。不同的 DHT 协议如 Chord[29]，CAN[30]，Pastry[31]和 Tapstry[32]区别主要在以下方面：节点 ID 与内容关键字 Key 的映射关系；查询消息的路由机制和节点动态性的处理机制。图 3-3 是 m=6 Chord 环的一个例子。从图中可以看到，关键字标识符 K10 的后继节点是 N14，因此 K10 的（K, V）对就存储在 N14

上。关键字标识符 K24 的后继节点是 N32，因此 K24 的（K, V）对就存储在 N32 上。只要找到关键字的后继节点，就可以找到实际存储文件内容的节点。

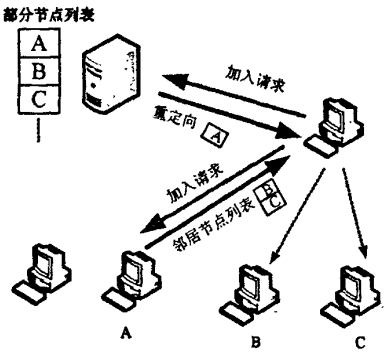


图 3-2 基于流言方案新节点加入过程

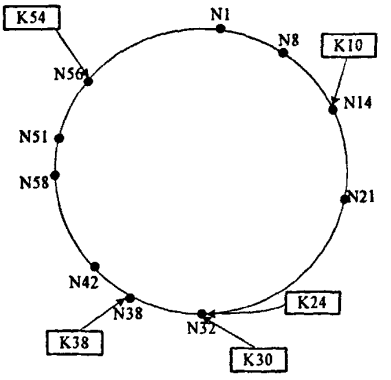


图 3-3 Chord 环

§3.1.3 节点组成员管理优化

综上所述，P2P 节点组成员管理首要目标是帮助新节点加入系统，集中式方案部署简单但是集中式方案的存在着“单点失效”和扩展性问题，并不适合大规模的 P2P 流媒体应用。基于流言的分布式方案具有好的扩展性，能够很好适应大规模系统要求，其关键是流言协议的设计。典型的流言协议中，节点随机的给自己的邻居节点发送状态消息，每个收到消息的节点继续向自己的邻居节点转发该消息，直到这个消息到达网络中所有节点。因此，衡量流言协议的一个重要标准就是能否以尽量小的消息开销及时有效的将节点的动态信息散发到整个网络。

为了更好地支持数据分发，满足流媒体传输速率、丢包、延时和抖动等要求，在帮助节点选择邻居节点，对覆盖网络进行优化也是节点组成员管理重要目标。分布式的节点组管理方案中流言算法随机选择邻居节点具有很大的盲目性，集中式方式由于维护了所有节点信息，易于对覆盖网络进行优化。考虑节点邻近性和异构性，优化覆盖网络时使其与实际的底层拓扑相结合是覆盖网络构建的关键问题。考虑了节点邻近性的覆盖网络，一方面可以降低数据的传播延时，降低了拥塞链路存在的可能性；另一方面可以减少跨 ISP 流量，使 P2P 流媒体能够更好的得到网络运营商的支持。支持节点异构性感知的覆盖网络可以对节点进行聚类，对不同接入带宽的节点实现区分服务，满足不同用户对服务质量的需求。

§3.2 数据调度机制

数据调度机制解决的问题是将数据高效的分发到整个 P2P 网络。好的数据

调度机制应具备以下特点：1) 能够充分利用参与会话节点的带宽，这也是 P2P 相比传统客户/服务器模式的最大特色之处；2) 能够满足应用需求，对流媒体来说最重要的就是播放连续性。3) 能够适应节点动态性带来的网络结构变化，这需要与节点组成员管理方式相结合。已有的数据调度机制分为三种："推"、"拉"和"推拉"结合方式。从某种程度上说，覆盖网络的结构决定了数据分发方式，因此我们将结合覆盖网络结构分别阐述这三种数据调度方式。

§3.2.1 "推"数据分发机制

"推"数据分发思想最初来源与 IP 组播，参与会话的节点组织成一棵树或者多颗树的逻辑结构。如图 3-4 所示，源节点是树根，数据从源节点自顶向下分发，树中的节点层次性决定了数据转发关系，上层父节点将自己接收到的数据"推"给底层子节点，因此，整个数据分发过程是由发送端控制的，接收端被动的从发送端接收数据。

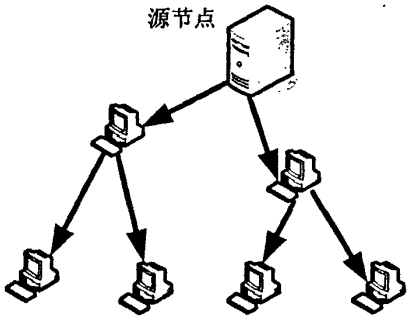


图 3-4 单棵树数据分发

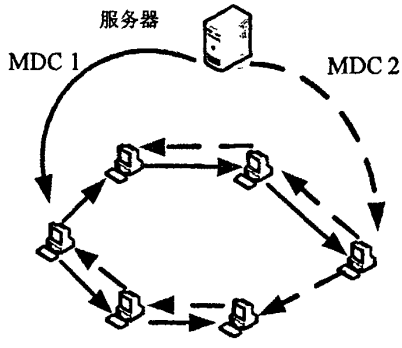


图 3-5 多棵树数据分发

与树结构相结合的"推"方案数据转发路径是确定的，数据根据已形成的覆盖网络自顶向下地分发到每个节点。这种传输机制是非常高效的。由于节点之间不需要传输控制消息，上层父节点只是单纯地将数据转发给底层子节点，数据传输延时小，控制消息开销小。但是这种方案的缺点也是很明显的。第一，树中最底层叶子节点带宽没有得到充分利用。相比较而言，少数内部节点承担了所有的数据转发任务，从公平性角度来说，系统的公平性不好。第二，父节点的性能直接影响到了子节点的视频接收质量，当父节点没有充足带宽满足流媒体传输需求时，其直接子节点视频接收质量将严重下降，这种影响将一直传递到底层的叶子节点。第三，树形结构对节点动态性容忍性差，当内部节点失效时，其所有的子节点都会受到影响。因此，单棵树的结构只适合小规模的应用层组播场景。针对

单棵树缺点，研究人员提出了多棵树的重叠网络。如图 3-5 所示，在多棵树结构中，源节点将视频数据分割成多个子流如 MDC，每棵树上转发一个流。节点在一棵树中是内部节点，承担数据转发任务，在其他树中是叶子节点接收数据。与单棵树相比，多棵树结构能够利用叶子节点的带宽，同时采用了 MDC 编码方式降低了视频速率，低带宽节点也能够参与数据转发。最后节点从多棵树中获取数据提高了对动态性的鲁棒性。但是相比较单棵树结构，维护多棵树覆盖网络的开销要大的多，另外高级视频编码技术（分层编码和 MDC）还不能够应用到实际系统，这些都影响了多棵树结构流媒体的部署。

§3.2.2 “拉”数据调度机制

与“推”方案不同，“拉”调度机制是基于接收端的，接收者根据自己的需要从邻居节点请求数据。其基本流程如下：节点与多个邻居节点建立连接，彼此交换缓冲区映像，然后根据自己的需求和邻居节点的缓存情况，向邻居节点请求数据，邻居节点收到请求后，向该节点发送请求的数据块。“拉”数据调度方案使得节点之间的形成了一种网状结构，如图 3-6 所示 。

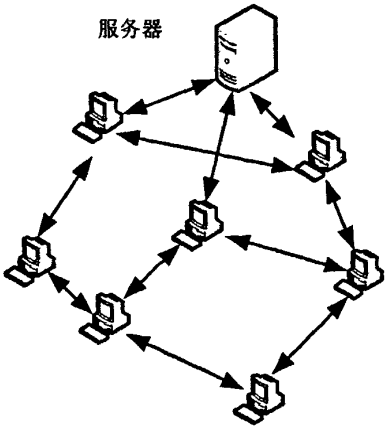


图 3-6 “拉”数据调度

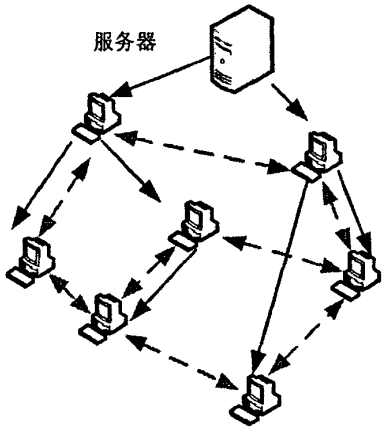


图 3-7 “推拉”结合方式

“拉”数据调度机制的优点很明显：第一，节点可以从多个邻居节点获取数据，提高了对动态性的鲁棒性；第二，节点根据自己的需求请求数据，避免了重复数据块的可能性；第三，节点之间相互合作传输数据，充分利用了节点带宽。正是由于这些优点，目前实际部署的流媒体系统，如 CoolStreaming, PPLive 等都采用了“拉”数据传输机制。但是发送节点需是根据在接收到接收端的请求后发送数据，相比较“推”分发机制，“拉”数据传输延时比较大，另外每次传输都

要交互缓冲区映像和发送请求消息，控制消息开销比较大。

§3.2.3 “推拉”相结合方式

混合模式的提出是为了结合前述两个数据传输方案的优点：“推”方式消息开销小，数据传输延时小；“拉”方式鲁棒性好，能够很好地适应了节点的动态性。mTreebone[33]提出了树形-网状混合覆盖网络。如图 3-7 所示，数据先沿着树结构自顶向下传输，树中传输丢失的数据块通过“拉”方式获取。因为大部分数据都是采用“推”方式沿着树的路径进行传递，传输延时小，同时网状结构可以辅助节点获取丢失的数据块，利用了叶子节点的带宽，提高了对节点动态性的鲁棒性。为了降低树中内部节点动态性的影响，可以选择稳定的节点作为树中内部节点，不稳定节点作为边缘节点，如何识别稳定节点是一个值得研究的问题。

§3.3 典型的 P2P 流媒体系统分析

从最初的卡内基梅隆大学的 ESM [34]，到香港中文大学开发的 CoolStreaming，再到目前广泛部署的 PPLive 和新兴的点播系统 GridCast，P2P 流媒体技术得到了长足的发展。由于设计目标和应用场景差异，这些流媒体系统采用了不同的节点组成员管理和数据调度方式。本节将介绍一些具有里程碑意义的 P2P 流媒体系统。

§3.3.1 实时流媒体系统

§3.3.3.1 ESM(End System Multicast)

A．节点组成员管理

端系统组播 ESM 是最早提出的应用层组播方案，其核心是应用层组播协议 Narada。Narada 基本思想是：首先将组播组的成员组织成一个网状结构(Mesh)，每个成员都维护所有组成员的列表，在网状覆盖网络的基础上，构建基于源节点的数据转发树。每个节点维护其他所有组成员信息，并周期性地交互状态信息，如果在一定时间内没有收到某邻居节点的更新消息，就认为这个邻居节点失效。

Narada 采用了集中式组成员管理方式，聚集点(rendezvous point)即中心服务器维护了系统中所有节点的信息。新节点加入时，首先从聚集点获取部分成员信息，新节点向这些成员发送加入请求消息。此后，新节点与邻居节点周期性交互状态消息，在流言协议的帮助下，新节点信息很快传播到系统中其他成员，在此

过程中新节点也了解到系统中其他节点信息。节点之间的这种邻居关系就形成了一种网状结构，网状结构提高了节点之间的连通度，避免了由节点动态性导致的网络分割，因此，Narada 协议能够很好的适应节点的高度动态性。但是由于采用了集中式的节点管理方式，以及节点需要维护所有组成员信息，Narada 协议不适合大规模的 P2P 应用。

B. 数据调度

在网状重叠网络的基础上，Narada 协议运行距离矢量路由协议来构建数据转发树，采用了“推”的数据传输方式。在图 3-8 中，节点和带箭头的逻辑链路构成组播树，而所有的逻辑链路构成的是网状网。当网状网发生变化时，每个成员上运行的组播算法自动计算新的组播树。Narada 协议还对数据转发树进行了优化，树中子节点可以根据需求调整自己的父节点。当节点需要更换父节点时，节点会向自己成员列表中的节点发送探测请求，收到探测请求的节点返回自己的性能信息：1）自己与源节点的延时；2）能否接收新的子节点；3）自己与探测节点的关系，是否是探测节点的子节点。根据这些信息，节点判断切换父节点后是否能够带来性能上的提升，如距离源节点的延时变小或者吞吐量的提高。如图 3-9 所示，节点 F 切换父节点，断开与节点 H 的连接，重新选择节点 E 为父节点，节点 E 距离源节点更近，因而减小了延时。

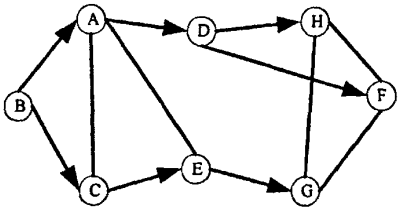


图 3-8 ESM 系统网状结构和数据转发树

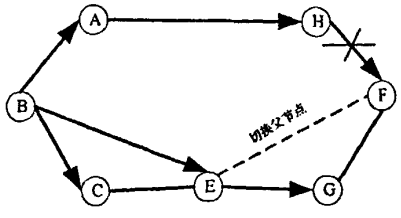


图 3-9 更换父节点

§3.3.3.2 CoolStreaming

CoolStreaming 是第一个基于接收端数据驱动的 P2P 流媒体系统，节点周期性地与邻居节点交互缓冲区映像信息，并根据自己的需求和邻居节点的缓冲状态向邻居节点发送数据请求，即“拉”数据传输方式。CoolStreaming 的系统结构[8]如图 3-10 所示，它主要包括 3 个模块：成员管理(Member Manager)，负责维护系统中部分对等节点的信息；伙伴管理(Partnership Manager)，用于和其它对等节点建立合作关系，节点从这些伙伴交换数据；数据调度(Scheduler)，负责具体的数据调度过程。

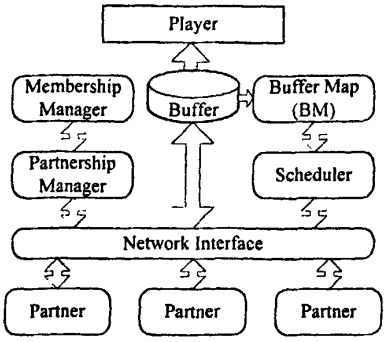


图 3-10 CoolStreaming 系统结构图

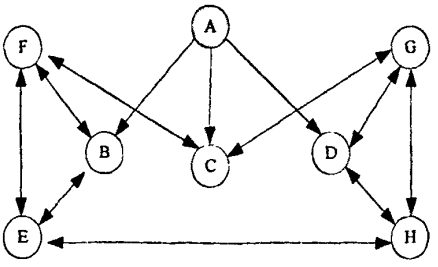


图 3-11 CoolStreaming 数据调度

A. 节点组成员管理

CoolStreaming 是分布式的节点组成员管理方式，采用了 SCAM (Scalable Gossip Membership protocol)流言协议来维护和更新节点状态信息。SCAM 具有较好的扩展性和鲁棒性，能够以较少消息开销实现了节点状态信息的快速分发。

CoolStreaming 中的每个节点都维护一个保存组中其他部分成员的节点列表 mCache。新节点首先与源节点取得联系，源节点从它的 mCache 中随机选择一个代理节点，将新节点的加入请求重定向到代理节点。代理节点发送自己的 mCache 给新节点。新节点把收到节点列表填入自己的 mCache 中，并与这些节点建立连接，从而加入系统。

针对 P2P 网络节点的高度动态性，每个节点周期性地生成宣布自己存活的成员消息。成员消息的格式为<seq_num, id, num_partner, time_to_live>，seq_num 指明该节点发送消息的序号，num_partner 表示该节点连接的节点个数，time_to_live 表示该消息的生存期。成员消息由 SCAM 流言协议负责在整个网络中分发。当节点收到这样的成员消息，如果 mCache 中已存在此节点的消息，则更新相应条目，否则就创建一个新的表项。mCache 中的每项的格式为<seq_num, id, num_partner, time_to_live, last_update_time>，前四项是收到的消息中的拷贝，最后一项 last_update_time 为上次更新该条目的时间。节点周期性地检查自己 mCache 中的每一项，计算该项的剩余生存时间。如果剩余生存时间小于等于零，则将它从 mCache 中删掉。

B. 数据调度方案

CoolStreaming 采用了“拉”数据调度方案，节点之间周期性地交互缓冲区映像(Buffer Map，表示了节点缓冲区拥有哪些数据块)来了解彼此的缓存情况，然后根据自己的需求和邻居节点的缓存情况发送数据请求。如图 3-11 所示，节点之间的数据流传输路径是非确定的、不定向的。相比之下，“推”数据传输方案中数据流传输路径是自顶向下的，节点之间的关系是明确的。

CoolStreaming 设计了一个稀有性优先的数据调度方案。基本流程如下：在交互缓冲区映像后，节点可以计算每个需求数据块的节点数，并根据历史信息估计每个父节点在当前的调度周期可以提供的带宽。如果某个数据块只有一个父节点有，直接向这个父节点请求；如果多个父节点拥有这个数据块，优先向可用带宽最大的父节点请求。相比较其他数据块，稀有数据块很有可能得不到及时传输，这种稀有性优先调度方案的好处是保证稀有数据块的传输，避免了由于稀有数据丢失造成的视频中断。实际上，在“拉”数据调度方案中，节点需要根据自己的需求和邻居节点的缓存情况发送数据块请求，为了满足流媒体传输的实时性要求，必须保证每个数据块在播放之前达到，在节点带宽异构的情况下，设计最优数据调度方案是一个 NP 难解问题。

§3.3.2 P2P 点播系统

与实时流媒体不同，P2P 点播系统 VOD(Video-on-demand)允许用户在任何时间收看时候视频的任何部分，支持 VCR 操作，具有更强的灵活性和方便性。这个特点也给 VOD 系统的设计带来了极大的挑战性，因此，相比较实时流媒体系统的迅猛发展，P2P VOD 系统开发相对缓慢。本节阐述华中科技大学开发的 GridCast 系统，GridCast 系统主要包括三个部分：源节点服务器(source server)，服务器上存放了视频的完整拷贝，视频以数据块的方式存储，服务器帮助用户节点获取数据块。目录服务器(tracker server)，目录服务器等价于集中式节点组成员管理的中心服务器，目录服务器维护了系统中所有节点信息，节点周期性地向目录服务器更新自己的状态，如播放位置和存活消息。黄页服务器(web portal)，维护了所有视频目录，用户可以通过网络浏览器选择视频播放。图 3-12 表示了 GridCast 系统结构和基本交互过程，有向线段 1，2 表示用户与黄页服务器连接，选择自己喜欢的视频，黄页服务器返回与相关的目录服务器信息。有向线段 3 和 4 说明用户与目录服务器建立连接，目录服务器返回视频服务器信息和其他正在播放视频的节点信息。节点收到这些信息后，与服务器和邻居节点建立连接，加入网络并请求自己需要的视频数据。

A. 节点组成员管理方案

在实时流媒体中，节点之间的播放是半同步的，播放延时差距小，节点组成员管理的重点在于优化覆盖网络，如考虑节点的邻近性和异构性，满足实时性要求。在 VOD 系统中，由于 VCR 操作，节点可以收看任何部分的视频，播放差异大，一个正在收看视频初始部分的节点不能为一个收看视频结尾部分的节点提供任何帮助。因此，在构建 VOD 覆盖网络时，考虑更多的是节点的播放位置。

GridCast 采用了集中式的组成员管理，目录服务器维护了所有节点的信息，包括存活性，节点的播放位置。新节点加入时，首先与目录服务器联系，目录服务器返回一部分节点列表，新节点发送加入请求消息，从而加入网络。当节点进行 VCR 操作时，也需要与目录服务器联系，并报告自己当前的播放位置，目录服务器将为节点重新选择邻居节点。除了集中式的组成员管理，节点还采用了流言协议来维护和更新自己的邻居节点列表。表 3-1 说明了 GridCast 中节点维护的成员信息。节点根据播放位置判断邻居节点与自己的关系。播放位置相近并且缓冲区有重叠的邻居节点属于伙伴，节点可以从伙伴处获取自己需要的数据。播放位置相近但是缓冲区不重叠的邻居节点属于邻居，节点与这些邻居节点周期性交互控制消息，目的是发现可能存在的伙伴。播放位置相差较远并且缓冲区没有重叠的邻居节点属于成员，节点维护这些成员节点的目的是保持较好的连通度，避免可能出现的网络分割。

表 3-1 节点成员信息

节点集	控制消息（共享）	数据块（共享）	转换标准	目的	成员最大数
成员	否	否	播放位置	备用节点集	100
邻居	是	否	播放位置	流言协议	50
伙伴	是	是	播放位置	数据交换	25

B. 数据调度方案

VOD 系统的调度方案与实时流媒体差别不大。为了保证视频播放的连续性，需要优先下载即将播放的数据块，为了提高对节点动态性的鲁棒性，需要从多个父节点进行调度获取数据。因此，撇开 VCR 操作，VOD 的数据块调度与实时流媒体没有什么区别。GridCast 的数据调度类似与 CoolStreaming，节点周期性地交互缓冲区映像，根据自己的需求和伙伴节点的缓存状况，采用稀有性优先策略发送数据块请求。

为了支持 VCR 操作，Gridcast 提出了一种基于锚定点的预取算法。图 3-13 是基于锚定点预取算法示意图。整个视频文件被分割为多个锚定点，在满足正常播放的前提下，节点利用剩余带宽优先预取锚定点处数据。当节点进行 VCR 操作时，将跳跃播放点重定位到距离最近的锚定点，由于预先获取了锚定点的数据，这样就可以降低节点的 VCR 操作延时，同时减少了对服务器的数据请求。如何有效地设置锚定点是该算法的关键。

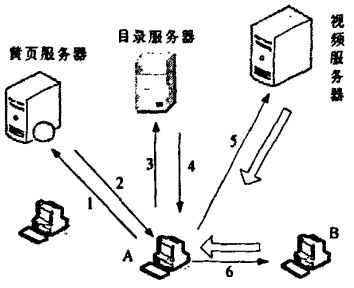


图 3-12 GridCast 系统结构

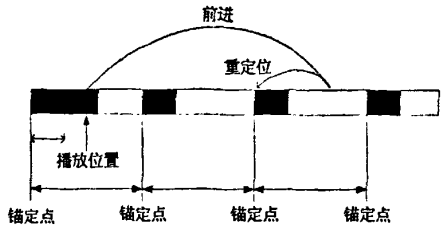


图 3-13 基于锚定点的 VCR 操作

§3.3 本章小结

组成员管理机制和数据调度是 P2P 流媒体系统两个关键问题。组成员管理维护和更新系统中节点信息，帮助新节点加入网络。组成员管理方式主要分为集中式和分布式两种。集中式组成员管理机制实现部署简单，由于维护了所有节点信息，支持对覆盖网络优化，但存在着“单点失效”和可扩展性问题。分布式的组成员管理不存在中心服务器，依赖流言算法传播节点信息，可扩展性强。源节点通过覆盖网络分发数据，数据调度方式依赖与覆盖网络结构。数据调度方式可分为“推”、“拉”和“推拉”混合模式。“推”和“拉”数据传输方式各具优缺点，“推拉”混合模式结合了这两种方式的优点。最后，本章详细介绍了几种典型的 P2P 流媒体系统，从节点组成员管理方式和数据调度两个方面进行了详细的讨论。

第四章 基于优先级的数据调度算法研究

在第三章我们介绍了 P2P 流媒体系统两个关键组成：覆盖网络构建和数据块调度。覆盖网络构建和数据调度是紧密关联的。一方面，构造与底层拓扑匹配的覆盖网络可以提高整个 P2P 流媒体系统性能：缩短数据传输路径，降低数据传输延时，减少跨 ISP 的流量。另一方面，在已有覆盖网络基础上设计快速高效的数据调度算法可以进一步的提高用户的观看体验，如降低启动延时，频道切换延时，提高播放的连续性和视频质量。本章深入分析了现有的数据调度方案的优缺点，并在此基础上提出了一种基于优先级的数据调度算法。

§4.1 已有的数据调度方案

§4.1.1 树形覆盖网络的数据调度方案

在 P2P 流媒体研究的初期阶段，研究人员借鉴了传统 IP 组播的思想，把参与会话的节点在应用层上组织成组播树的结构。树的根节点是流媒体服务器，同一棵树上的节点共享一个频道流出的数据。树形网络通常采用了"推"数据分发方案，节点之间首先确定父子关系，父节点收到数据后直接传递给子节点，整个数据分发过程由父节点控制。树形重叠网络的代表有 ESM 和 NICE[35]。

"推"方案最大优点是数据分发延时小，父节点接收到数据后，直接转发给子节点，数据分发效率高。但是缺点也很明显，首先树形结构并不适合高度动态性的 P2P 网络，一个父节点退出，会影响到其下游所有的子节点，当退出节点靠近源节点时，受影响的节点更多。其次树形结构并不能够利用叶子节点的带宽，而叶子节点占据了整个系统的大部分，因此系统的带宽利用率不高，不适合高带宽的流媒体应用。

针对树形重叠网络的缺点，研究人员提出了各种改进方案，典型代表有 SplitStream。但是这些方案主要集中在重叠网络的构建，用多棵树的重叠网络代替单棵树结构，每个节点同时加入到多个数据转发树，节点在一棵数据转发树中充当内部节点转发数据，而在其他数据转发树中是叶子节点，可以充分利用节点带宽。同时由于从多个父节点接收数据，多棵树结构能够更好地适应 P2P 网络的动态性。另外由于这些改进方案依然沿用了"推"数据方案。相比单棵树，维护多棵树重叠网络需要很大的控制消息开销。

§4.1.2 网状覆盖网络的数据调度方案

为了克服树形覆盖网络的缺点，研究人员提出了网状覆盖网络，目前广泛部署的 P2P 流媒体系统基本上都采用了这个结构，典型代表有 CoolStreaming 和 PPLive。网状覆盖网络采用了“拉”的数据块分发方案。节点根据自己的需求，周期性地从多个父节点获取数据。具体过程是：在每个调度周期，节点之间交换缓冲区映像；节点根据自己的需求和邻居节点缓冲状况，带宽向各个邻居节点分配请求数据块。邻居节点根据节点的请求发送数据块。与“推”方案相比，“拉”方案中节点从多个邻居节点获取数据，因此，能够很好地适应 P2P 网络节点动态性，充分利用网络节点带宽。但是缺点也很明显，在实际传输数据之前，节点之间需要交换一系列的控制报文，发送节点只有在收到接收节点的请求才发送数据，数据传输延时大，控制消息开销大。对 P2P 流媒体的实际测量表明[36][37]，实时流媒体中节点之间存在较大的播放延时。

[38][39][40]对 P2P 流媒体系统数据调度算法进行了建模，理论分析了各种数据调度算法性能，如数据分发延时和速率。[38]提出了雪球 snow-ball 算法来实现数据的最快分发，降低节点的启动延时。为了方便计算分发延时，时间被划分为调度周期，同时对节点带宽和数据块大小进行归一化，即在单个调度周期内，具有单位带宽的链路只能传输一个数据块。假设系统中节点总数为 N，在同构情况下，即所有节点为单位带宽，推出将单个数据块分发到全网最短时间为：

$$D_{\min} = 1 + \lceil \log_2 N \rceil \quad (1)$$

证明如下，在第一个调度周期，源节点将数据块 C 随机地分发给节点 P1，将数据块注入到 P2P 网络后，源节点不参与数据分发。第二个调度周期末，网络中最多有两个节点拥有数据块 C，依次类推，n 个调度周期后，系统中拥有数据块 C 的节点数为 2^{n-1}，即拥有数据块的节点数成几何级数的增长。

与单个数据块不同，流媒体中源节点不停地向 P2P 网络注入新的数据块，假设没有节点波动，能否保证每个数据块都能够在最短时间分发给每个节点？作者提出了一种雪球(snow-ball)算法并证明了其可以在同构的网络环境下实现每个数据块的最快分发。其核心思想是每个节点不能同时拥有多于一个正在分发的数据块，源节点总是将新的数据块注入到下一个调度周期没有分发任务的节点，这样就可以充分利用节点的带宽，避免了“内容瓶颈”，保证了每个数据的最快分发。雪球算法是集中式的算法，源节点需要与所有节点建立连接并负责具体的数据分发过程，这在 P2P 网络中是不实际的。图 4-1 是雪球算法示意图。系统中有 8 个节点，6 个子图表示了数据分发的连续 6 个调度周期，节点旁边的白色方块表示节点已经拥有这个数据块，节点之间的有向线段表示数据传输的方向，黑

色方块代表服务器注入到网络中的新数据。在第一个调度周期，节点 0 向节点 4 传输数据块 0，同时服务器将数据块 1 注入到节点 1。第二个调度周期，已经有 4 个节点拥有数据块 0，节点 1 传输数据块 1 给节点 3，源节点向节点 2 注入新数据块 2。根据等式 1，在第三个调度周期末，系统中所有节点都有数据块 0，一半节点拥有数据块 1，2 个节点拥有数据块 2。拥有数据块的节点数成几何级数增长，每个数据块都能够在三个调度周期内传播到系统中所有节点。

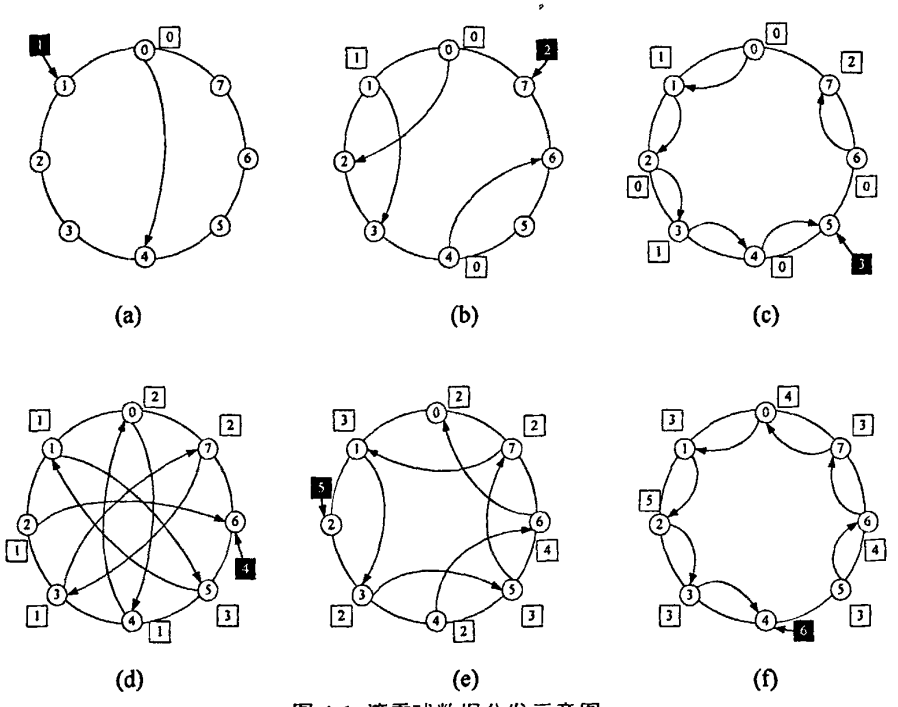


图 4-1 滚雪球数据分发示意图

针对雪球算法集中式数据调度的缺点，本章提出了一种基于优先级的数据调度方案(Distributed Priority-based Chunk) DPC。DPC 算法是分布式“拉”数据调度方案，节点根据邻居节点信息做出数据转发决策，在每个调度周期初始，节点与邻居节点交互控制消息获取邻居节点状态。为了降低控制消息开销，在实际应用中需要对邻居节点数进行了限制。在同构网络环境下，我们理论证明 DPC 算法能够实现最快数据分发，在一般异构的网络环境下，我们同样给出了最快分发理论界限，仿真结果表明了 DPC 算法能够很好的逼近理论值。

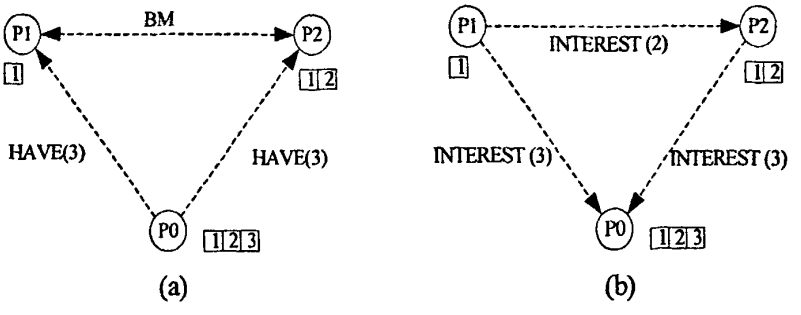
§4.2 DPC 算法

§4.2.1 数据分发延时模型和消息符号定义

我们首先建立数据分发延时模型，推导数据分发延时理论最小值，为算法设计提供理论依据。为了便于计算，时间被划分为调度周期，视频数据流被划分为数据块并归一化。每个数据块被标上唯一的序列号。与数据块传输延时相比，传播延时可以忽略，因此在一个调度周期内单位带宽链路上可以传播一个数据块。源节点只负责分发一份数据到 P2P 网络，并不参与后续的数据转发过程。我们假设一个全连通的网络拓扑结构。在介绍算法之前，首先说明节点之间的交互消息和符号定义。

- $p_i, i=0,...,N$：参与组播会话的节点，P_0表示源节点。
- $U_i, i=0,...,N$：节点的上传带宽。
- have(k)：源节点通知所有节点自己拥有数据块 k。
- BM：Buffer map，节点之间相互交换的缓冲区映像。
- interest(k)：如果节点P_i拥有数据块 k，而节点P_j没有这个数据块。节点P_j向节点P_i发送 interest 消息表示自己希望获得数据块 k。
- offer(k)：如果节点P_j运行节点P_i下载数据块 k，节点P_j将发送 offer 消息。
- decline(k)：节点P_i向节点P_j发送 decline 消息，表明不会从P_j处获取数据块 k。
- request(k)：节点P_i向节点P_j请求数据块 k。
- LS：Largest Sequence，节点拥有数据块的最大序列号，也就是最新的数据块。

图 4-2 是节点之间消息交换示意图。源节点 0 通知节点 1 和 2 自己拥有数据块 3，节点 1 和节点 2 交互 BM 信息(a)。节点 1 和 2 向源节点 0 发送 interest (3)消息，另外节点 1 向节点 2 发送 interest (2)消息，表明自己对数据块 2 感兴趣(b)。源节点随机选择节点 1 发送 offer(3)，同时节点 2 向节点 1 发送 offer (2) (c)。节点 1 向节点 2 发送 request(2)消息请求传输数据块 2，同时发送 decline(3)消息，表示自己不请求数据块 3(d)。



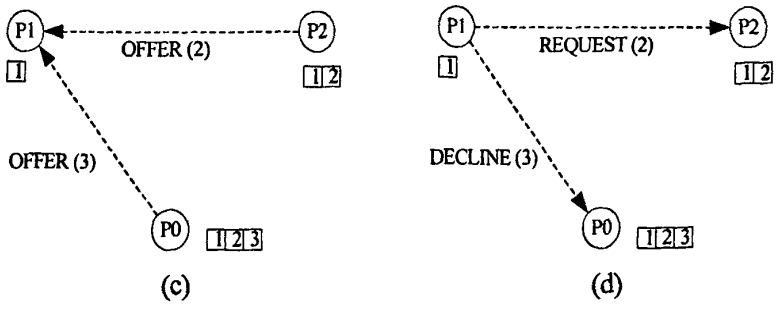


图 4-2 节点之间消息交换示意图

§4.2.2 基本算法

在同构环境下，所有节点具有相同的带宽，不失一般性，我们假设所有节点包括源节点的带宽为 1。流媒体速率为 1 个数据块/调度周期。DPC 算法中对节点和数据块进行了优先级的划分，优先传输优先级高的数据块和响应优先级高的节点请求。优先级规则有以下两种：

规则 1：在 P2P 流媒体系统中，数据具有不同的优先级，距离播放时间越近的数据越重要，如果没有及时接收，就会造成视频中断或者跳跃，从而影响播放质量。我们认为较小序列号的数据块有较高的优先级，序列号越小的数据块距离播放时间越近。

规则 2：为了充分利用节点带宽，避免内容瓶颈。节点的 LS 越小，表明其已经落后于其他节点，自己没有其他节点需要的数据。我们将这样的节点设置为高优先级，目的是帮助落后节点，利用落后节点的带宽。

同构环境下的 DPC 算法分为两个部分：发送端调度和接收端调度。作为一个数据提供者，节点会接收到多个邻居节点 interest 消息。节点需要选择邻居节点和数据块进行发送 offer 消息。发送端调度基本流程：发送节点对发送 interest 消息的邻居节点根据规则 2 进行排序，节点的 LS 越低，优先级越高。发送节点向优先级最高的节点（LS 最低的节点，优先级相同的多个节点随机选择）发送 offer 消息。如果发送节点接收到选定节点发送的 decline 消息，则选择下一个优先级最高的节点发送 offer。图 4-3 是发送端调度伪代码。

作为一个数据接收者，节点可能收到多个 offer 消息，节点必须选择提供者和数据块发送 request 消息。接收端调度基本流程：接收节点对发送 offer 消息的节点进行排序，首先是根据规则 1，选择序列号最小的数据块，再根据规则 2 在拥有这个数据块的节点中选择 LS 最小的节点进行请求，避免选择具有较大 LS 的节点，这样做的好处是 LS 大的节点可以向其他节点提供新的数据块（具有较大 LS），保证了新数据块的及时分发。由于视频速率为单个数据块/调度周期，

我们限制节点在每个调度周期只能获取一个序列号大于自己LS的数据块。图4-4是接收端调度伪代码。

```
while ( P_s.interest-list ≠ φ )
{
 m← the INTEREST message with the lowest sender's LS, break tie randomly;
 P_i ← the peer sending m;
 k← the sequence number of the chunk that P_i interests;
 Send OFFER(k) to P_i ;
 m' ← the message replying from peer P_i ;
 if m'==REQUEST(k):
    send chunk k to P_i ;
    return;
 else m'==DECLINE(k):
    delete m from interest-list;
}
```

图4-3 发送端调度算法伪代码描述

```
count=0;
while ( P_i.offer-list ≠ φ )
{
 M← the offer set providing the lowest chunk sequence;
 K← the lowest chunk sequence number;
 m← the offer message with the lowest sender's LS in M, break tie randomly;
 P_s ← the peer sending m;
 Send REQUEST(k) to P_s ;
 Send DECLINE(k) to other peers providing OFFER(k);
 Delete M from offer-list;
 if k > P_i.LS:
    count+=count+1;
 /* a peer can download at most one chunk whose sequence number > its LS */
 if count > 0:
    break;
}
```

图4-4 接收端调度算法伪代码描述

图4-5为DPC算法的简单示意图，我们用连续的5个调度周期说明数据分发过程。系统中有5个节点，有4个数据块需要分发到网络中。初始时节点没有任何数据块。源节点每调度周期上传一个数据块。黑色方块表示节点拥有了相应数据块。第一个调度周期，节点1从源节点处获得数据块1。第二个调度周期，源节点0上传数据块2给节点5。同时节点1给节点2传输数据块1。依次类推，拥有某个数据块的节点成几何级数增长，在第四个调度周期，所有节点都将拥有数据块1，数据块2在第五个调度周期分发到所有节点，数据块按序地分发给所

有节点，这表明 DPC 算法不仅可以最快的分发数据，同时按序递交特性非常适合流媒体应用连续性要求。

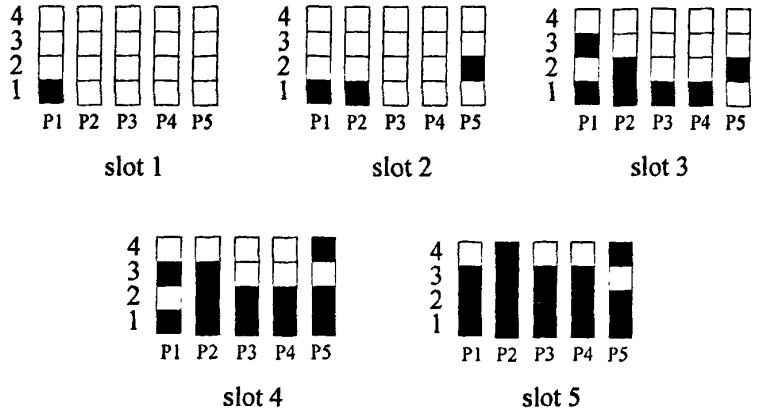


图 4-5 DPC 数据分发示意图

真实的网络环境下节点的带宽是异构的。为了将 DPC 算法扩展到异构网络环境下，需要对发送端和接收端调度进行一些修改。对发送端来说，节点可以在一个调度周期里传输多个数据块，因此发送节点可以在一个调度周期里响应多个邻居节点请求。对接收端来说，节点可以在一个调度周期内下载多个序列号大于自己 LS 的数据块。与同构环境下相同，每个调度周期序列号大于自己 LS 的数据块不应超过视频速率。

§4.2.3 算法最优性证明

首先证明 DPC 算法可以在同构环境下实现数据块的最快分发。根据等式(1)，可以推出将 M 个数据块分发到拥有 N 个节点的网络需要的时间为：

$$T = M + \lceil \log_2 N \rceil \tag{2}$$

定理 4.1：忽略控制消息开销，在全连通的网络中，DPC 算法实现数据最快分发。

证明：源节点带宽为 1，源节点在调度周期 k 上传数据块 k。没有分发到所有节点的数据块为分发数据块，S_k 表示当前拥有分发数据块 k 的节点集合。$|S_k(i)|$ 表示在第 i 个调度周期末，拥有分发数据块 k 的节点数。最快数据分发算法应满足充分条件Δ[38]：

$$|S_k(i)| = \begin{cases} 2^{i-k} & k \le i < k + D_{\min} - 1 \\ N & i \ge k + D_{\min} - 1 \end{cases} \tag{3}$$

我们分别在 $N = 2^{D_{\min}-1}$ 和 $2^{D_{\min}-2} < N < 2^{D_{\min}-1}$ 两种情况下归纳证明 DPC 算法满足

这个充分条件。

1)：$N=2^{D_{min}-1}$

初始条件：根据发送端调度，空节点具有最高的优先级。在第D_{min}调度周期之前，系统中有足够的空节点。源节点和非空节点总是响应空节点的数据请求，同时根据接收端调度，节点每个调度周期只能下载一个序列号大于自己LS的数据块。因此，在第D_{min}调度周期前，可以证明充分条件Δ成立，集合$\{S_k,k>0\}$是不相交的。

递推：假设充分条件Δ在调度周期D_{min}成立，DPC算法能够保证Δ在调度周期$D_{min}+1$仍然成立。在调度周期D_{min}初始，系统中有$N/2$个节点拥有数据块1。根据接收端调度规则，序列号低的数据块优先级高，在调度周期D_{min}没有数据块1的节点将接收来自S_1的OFFER(1)。因此，在调度周期D_{min}末，所有节点都会有数据块1。同时根据发送端调度，集合S_1中的节点比集合$S_2 \cup ... \cup S_{D_{min}-1}$的节点具有更高的优先级，因此集合$S_1$节点将会接收来自集合$S_2 \cup ... \cup S_{D_{min}-1}$和源节点的共$N/2$ OFFER。因此充分条件Δ在时隙D_{min}成立。类似的，我们可以把调度周期$D_{min}+1$的数据块2等价于调度周期D_{min}的数据块1，因此，充分条件Δ在调度周期$D_{min}+1$成立。

综上所述，DPC算法可以在情况1实现数据最小分发延时。

2)：$2^{D_{min}-2}<N<2^{D_{min}-1}$

2)与 1)不同之处在于集合$\{S_k,k>0\}$存在相交。一些节点可能同时拥有两个分发数据块。在图4-5，节点2和5在第四个调度周期完成时拥有两个分发数据块，节点2有数据块2和3，节点5有数据块2和4。但是这种情况并没有违背充分条件Δ。根据接收端调度，节点收到多个提供相同数据块的OFFER消息时，节点将向LS最小的提供者请求数据。因此图4-5中在第五个调度周期，节点1会从节点3，4和5收到OFFER(2)。但是节点1选择节点3或者4传输数据块2，因为节点3和4的LS比节点5要小。与情况1的中的证明类似，我们可以得到充分条件在情况2成立。

在异构环境下，节点具有不同的上传带宽。我们用$u_{max}=\max\{u_i,i=0,1...,N\}$表示系统中带宽最大的节点。推出单个数据块分发到网络中的最小延时为：

$$D^*_{min}=1+\left\lceil \log_{(U_{max}+1)}\frac{N}{U_0}\right\rceil \tag{4}$$

证明：源节点在第一个调度周期上传数据块给u_0个节点。假设这些节点具有最大的上传带宽u_{max}。在第二个调度周期完成时，至多有$(u_{max}+1)u_0$节点拥有这个数据块。依次类推，在第i调度周期完成时最多有$(u_{max}+1)^{i-1}u_0$节点拥有数据块。

根据等式(4)，我们可以推出将 M 个数据块分发到所有节点至少需要的调度周期

$$T^* = \frac{M-1}{U_0} + D^*_{\min} \tag{5}$$

§4.3 仿真与分析

我们首先在不同的网络环境下验证 DPC 算法性能。整个视频文件划分为 100 个数据块，源节点将这些数据块注入到 P2P 网络。实际的 P2P 网络拥有大量的参与节点，构建一个全连通的网络拓扑是不实际的。为了仿真实际的网络环境，我们对节点的邻居节点数进行了限制。每个节点从系统中随机选择 L(L<20)个邻居节点。由于在具体的数据分发之前，节点之间需要交互控制消息，限制邻居节点数可以降低控制消息开销。针对“拉”数据调度方案数据传输延时大，启动延时大的缺点，我们给出了两个评价指标。

- 数据分发延时，数据分发到整个网络所需要的时间；
- 丢包率，在一定的启动延时下，错过播放时间的数据块的比例。启动延时越大，节点的丢包率越小，我们希望在较小的启动延时下能够达到较低的数据丢包率。

§4.3.1 性能分析

实验 1 的目的是验证 DPC 算法在同构网络条件下能够很好的逼近理论延时最小值。同构网络中所有节点的带宽均为 1，源节点每个调度周期注入一个数据块到系统。图 4-6 显示了 DPC 算法和稀有性优先算法分发 100 个数据块到不同数量的节点需要的时间，可以看出 DPC 算法相比稀有性优先算法能够更快将数据分发到整个网络，并接近理论延时最小值。理论延时最小值由等式(2)得到。

在实验 2 中，我们比较 DPC 算法和稀有性优先算法在异构网络环境下的性能。源节点的带宽为 $u_0 = 4$，流媒体视频速率为 4 个数据块/调度周期。节点的带宽随机分布在 2 和 6 之间，系统平均带宽为 4 个数据块/调度周期，等于流媒体视频速率。从图 4-7 从我们可以得到类似实验 1 的结论，理论延时最小值是根据等式(5)获得。

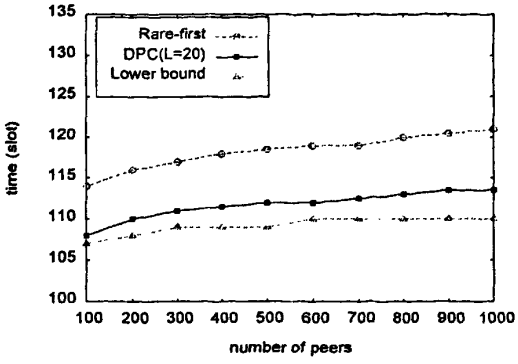


图 4-6 同构环境下 DPC 算法

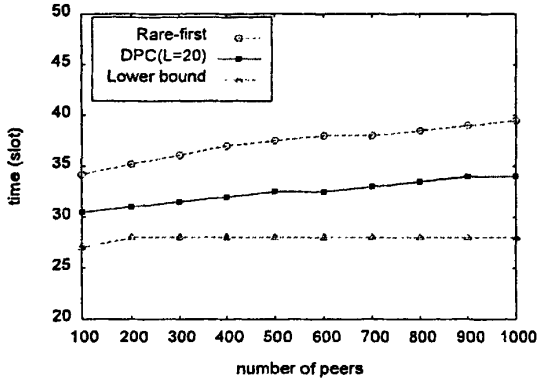


图 4-7 异构环境下 DPC 算法

在 P2P 流媒体中，每个数据块都有相应的播放时间，节点接收到的数据块超过了其播放时间，这就会影响视频播放的连续性。在实验 3 中我们计算在一定启动延时下，迟到的数据块在总数据块中的比例。启动延时是节点从收到第一个数据块到实际开始播放需要等待的时间。为了保证播放的连续性，在开始播放前设置较大的启动延时可以缓存足够数据，而较小的启动延时意味着缓存数据少，增加了播放不连续的可能性。用户总是希望当他们选择一个频道后，能够及时的观看视频而不是长时间的等待。实验中节点数为 1000，其他参数设置等同实验 2，从图 4-8 可以看出 DPC 算法相比稀有性优先算法在相同启动延时下具有更低的数据丢包率。这是由 DPC 算法的特点导致的，一方面 DPC 算法能够较快的分发数据到整个网络，获取等量的缓存需要更短的时间；另一方面，DPC 算法能够将数据块近似按序地发送到每个节点，具有较好的数据分发连续性。相比之下，稀有性优先算法强调数据块的可用性，数据块分发连续性并不好。

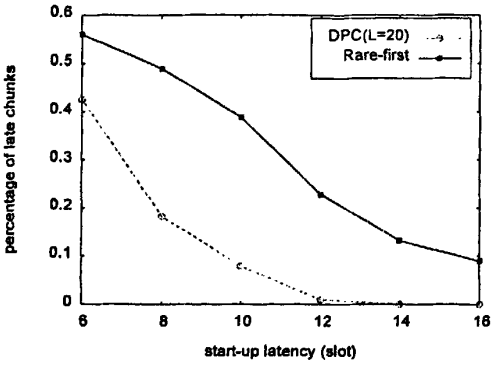


图 4-8 丢包率和启动延时

§4.4 算法的扩展和讨论

本节是上一节内容的扩展和改进，在 4.3 节中我们提出了一种数据分发最小延时算法，基本思想是根据节点的缓存状态和数据块序列号分别对节点和数据块划分优先级，优先响应优先级高的节点请求和传输优先高的数据块，严格按照优先级进行数据分发保证了数据的几何级数增长，充分利用了每个节点的带宽，从而最小化传输延时。但是严格的优先级需要节点之间交换大量的控制消息开销并引入了消息交换延时。本节深入分析 DPC 算法的控制消息开销和消息交换延时，并进一步提出了同构环境下基于“推”的 DPC 数据调度方案。

§4.4.1 消息开销和交换延时

在 DPC 算法中每个调度周期节点之间都要交互大量的控制消息，4.2 节描述了具体的控制消息，这些控制消息保证了数据块传输严格按照优先级进行。由于拥有数据节点数量的几何级数增长，即等式(1)，在第一个数据块分发到所有节点后的每个调度周期，都有一个数据块分发到所有节点，我们称系统处于稳定状态，并在稳定状态下分析消息开销和交换延时，

根据等式(3)，容易推导当系统中有 N 个节点时，稳定状态下每个调度周期内分发的数据块数为 $D_{\min}-1$，需要承担数据传输任务的节点满足以下条件：

$$|\phi_i(k)|=\begin{cases}2^{i-k-1} & i-D_{\min}+2\le k<i\\ N-2^{D_{\min}-2} & k=i-D_{\min}+1\\ 0 & k\le i-D_{\min}+1\end{cases} \tag{6}$$

$\phi_i(k)$ 表示在调度周期 i 需要传输数据块 k 的节点集合，易得 $\sum_{k=1}^{i-1}|\phi_i(k)|=N-1$，并

且集合$\{\phi_i(k),\ i-D_{\min}+1\le k\le i-1\}$不相交。

定理 4.2：当$N=2^{D_{\min}-1}$时，消息传输总延时 5l≤L≤N+3。

证明：当$N=2^{D_{\min}-1}$系统处于稳定状态 i 时根据等式(6)，N/2 个节点拥有数据块$i-D_{\min}+1$，N/4 个节点拥有数据块$i-D_{\min}$，一个节点有数据块 i-1，一个节点在当前调度周期没有数据传输任务，其 LS 也是最小。根据 DPC 调度算法，

- 在第一个单向传输延时，节点之间相互发送 interest 消息；
- 第二个单向传输延时，有数据传输任务节点都向 LS 最小的没有数据传输任务的节点发送 offer 消息；
- 第三个单向传输延时，这个节点选择一个能够提供数据块$i-D_{\min}+1$的节点发送 request 消息，同时向其余节点发送 decline 消息；
- 第四个单向传输延时，属于$\phi_i(i-D_{\min}+1)$的节点（除去向 LS 最小节点发送数据的）向集合$\{\phi_i(i-D_{\min}+2)\cup\ldots\cup\phi_i(i-1)\}$发送 offer 消息，同时集合$\{\phi_i(i-D_{\min}+2)\cup\ldots\cup\phi_i(i-1)\}$和源节点向集合$\phi_i(i-D_{\min}+1)$中的节点发送 offer 消息，在理想情况下发送节点集合和接收节点集合中的节点一一匹配，相互发送 offer 消息；
- 第五个单向传输延时，收到 offer 消息的节点发送 request 消息请求数据传输。

综述所述，消息传输总延时最小值为 5l。根据 DPC 发送端调度规则，当节点 LS 相同时随机选择发送 offer 消息，因此在最坏情况下，在第四个单向传输延时，属于$\phi_i(i-D_{\min}+1)$的节点（除去被选中的）都向集合$\{\phi_i(i-D_{\min})\cup\ldots\cup\phi_i(i-1)\}$中的同一个节点发送 offer 消息，同时集合$\{\phi_i(i-D_{\min})\cup\ldots\cup\phi_i(i-1)\}$和源节点也都向集合$\phi_i(i-D_{\min}+1)$中的同一个节点发送 offer 消息。依次类推，直到每个集合中只剩下一个节点完成。这个过程共需要 N 单向传输延时。因此最坏情况消息传输总延时为 N+3。

定理 4.3：当$2^{D_{\min}-2}<N<2^{D_{\min}-1}$时消息传输总延时为 $5l\le L\le 2^{D_{\min}-1}+1$。

证明：在稳定状态下根据等式(5)，拥有较大 LS 数据块的节点数为$2^{D_{\min}-2}$（包括源节点），因此 LS 最小的节点数为$N-2^{D_{\min}-2}+1$，并且$2^{D_{\min}-2}\ge N-2^{D_{\min}-2}+1$。在理想情况下完美匹配，LS 最小的节点请求数据前需要两个消息传输延时，若存在剩余数据需要分发，即$2^{D_{\min}-2}>N-2^{D_{\min}-2}+1$，这些数据块将分配给属于集合$\phi_i(i-D_{\min}+2)$的节点，同理在完美匹配下节点开始请求数据前需要两个消息传输延时。因此在消息传输延时的最小值为 5l。

根据 DPC 算法，最坏情况下 LS 最小的节点请求数据前需要的消息传输延时是$(N-2^{D_{\min}-2}+1)\times 2$，若存在剩余数据需要分发，最坏情况下需要的传输延时为$(2^{D_{\min}-1}-N-1)\times 2$。因此总的最大传输延时为$2^{D_{\min}-1}+1$。

我们用图 4-9 来说明消息传输延时。图 4-9(a)表示了调度周期初始节点的缓

存状况，第一个数据块已经分发到所有节点，在当前调度周期需要完成第二个数据块的分发。假设任意节点之间的单向传输延时为固定值 l，控制消息大小为单位值，忽略节点之间缓冲区映像消息交换。

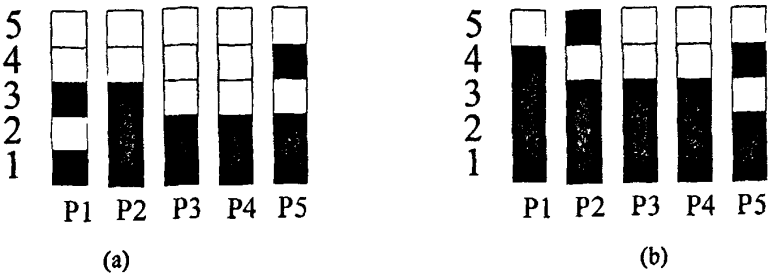


图 4-9 节点缓存数据分布图

- 第一个单向传输延时，节点之间相互发送 interest 消息，如节点 P2 需要向 P5 和源节点发送 interest 消息；
- 第二个单向传输延时，根据 DPC 算法发送端调度规则，节点 P3 和 P4 的 LS 最小，具有最高优先级。其余节点在这两个节点中随机选择发送 offer 消息；P3 和 P4 向节点 P1 发送 offer 消息。理想情况下，节点 P1，P2 分别给 P3，P4 发送 offer 消息。
- 第三个单向传输延时，节点 P3，P4 分别向 P1，P2 发送 request 消息，请求传输数据块 3，同时向其余节点（P5，源节点）发送 decline 消息。节点 P1 随机选择节点 P3 或者 P4 发送 request 消息，请求数据块 2。
- 第四个单向传输延时，理想情况下 P5 和源节点分别向节点 P1 和 P2 发送 offer 消息；
- 收到 offer 消息的 P1 和 P2 在第五个单向传输延时分别发送 request 消息，分别请求传输数据块 4 和 5。

因此，完成最后数据块传输之前，总的消息传输延时至少为 5l。图 4.9(b)为数据传输完成后节点的缓存状况。

§4.4.2 基于“推” DPC 算法

DPC 算法通过交互控制消息，严格按照优先级将数据从邻居节点“拉”过来，牺牲了控制消息开销换取数据分发的几何级数增长。基于“推” DPC 算法不需要节点之间交换控制消息，数据的发送完全由发送端决定。整个算法分为两部分：集合形成和匹配过程。集合形成是在每个调度周期初始，节点之间交换缓冲区映像，根据邻居节点的缓冲区状况生产相应数据块的发送节点集和接收节点集。匹配过程是在发送和接收节点集合形成之后，将两个集合中发送节点与接收节点一一匹配，避免随机发送导致的“发送冲突”，即多个发送节点向同一个接

收节点发送相同的数据块。表 4-1 是算法中的一些符号定义说明，图 4-10 是基于“推”DPC 算法伪代码。

表 4-1 算法中符号定义说明

chunk-list	属于本节点但是没有分发到所有节点的数据块列表
R_i	需要数据块 i 的节点集合；
S_i	拥有数据块 i，并且最小 LS 的节点集合；
F	进入“完成阶段”的节点集合；
A_i	拥有数据块 i 的节点集合；

```
1  while (P. chunk-list ≠ Ø)
2  {
3    if (P is not in Si):
4      continue;
5    else:
6      if (|Ri| ≤ |Si|):
7        recall matching strategy to map senders and receivers;
8        if (P is chose to send chunk i):
9            break;
10     else:
11       if (N/2 < 2*|Ai|):
12           choose peers set F from set Ri, these peers are in finishing status;
13           if ( |F| < |Si|):
14               choose peers from set Ri, these peers has the lowest LS,
15               until |F| == |Si|;
16           recall matching strategy to map senders and receivers;
17       else:
18           merge S sets that satisfy the same condition as Si;
19           if (peers' LS is lower than LS of peers that will be in finishing status):
20               choose set I from the intersection of corresponding R sets, these
                 peers has the lowest LS, and are not in other receiving sets;
21               if ( |I| < |S|):
22                   choose peers from intersection until |I| == |S|, these peers will
                     be in finishing status, and they are not in other receiving sets;
23               recall matching strategy to map senders in S and receivers in I;
24           else:
25               recall matching strategy to map senders in S and receivers in I;
   }
```

图 4-10 基于“推”DPC 算法伪代码。

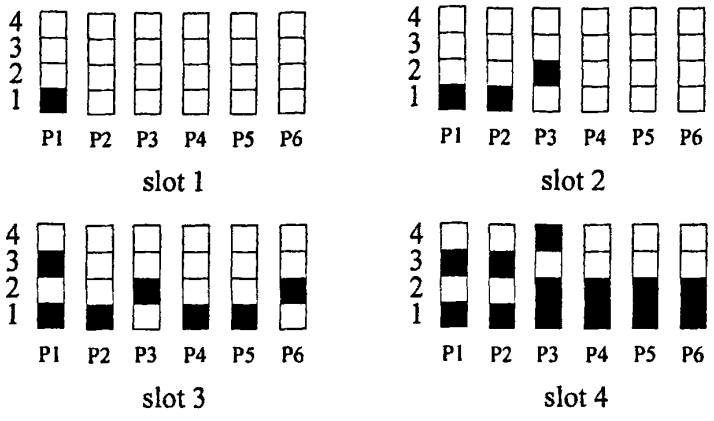


图 4-11 基于“推”DPC 算法示意图

系统有 N 个节点，当 N/2≤|Ai|<N 时，称数据块 i “进入完成阶段”。源节点在每个调度周期将新数据块“注入”系统，其等价于拥有最大数据块序列号的节点。图 4-11 说明改进的算法方案。

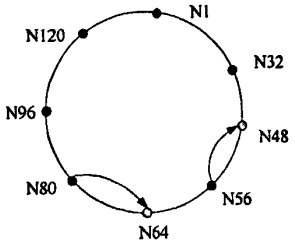
- 在第一个调度周期，系统节点 P1-P6 的 *chunk-list* 列表为空，根据匹配规则，源节点将数据块 1 传输给节点 P1；
- 在第二个调度周期，源节点和节点 P1 分别形成发送节点集合 S2 和 S1，根据伪代码 18-25 行，集合 S1 和 S2 合并成新集合 S，P2-P6 形成新接收节点集合 R，根据匹配规则进行数据传输，节点 P2 和 P5 分别获取一个数据块；
- 第三个调度周期，拥有数据块 1 的节点集|A1|=2，根据伪代码 11-16 行，P3 和 P4 形成接收节点集合 R1，同时源节点和 P5 形成新集合 S，根据伪代码 18-25，数据 1 在本调度周期末进入完成阶段，I={P1, P6}，匹配规则实现一一对应完成数据传输，数据块 1 进入完成阶段；
- 在第四个调度周期，R1={P3，P6}，S1={P2，P4，P5}，根据伪代码 6-9 和匹配规则，数据块 1 在第四个调度周期末分发到所有节点。其他数据块的分发与前述类似。

匹配过程是将发送和接收节点集合进行匹配，避免随机分发带来的“发送冲突”问题。系统中每个节点被分配一个 m 比特标识符 ID。假设两节点 1 和 2 的标识符分别为 ID1 和 ID2，则两节点之间“距离”定义为(ID1 – ID2) mod 2^m。图 4-11 是具体的匹配策略伪代码。

```
1   for i in S_k:
2     for j in R_k:
3         compute the "difference" between peers;
4   end
5   form the "difference" matrix V;
6   while (V≠Ø) :
7     choose the minimal value in the matrix, break tie randomly;
8     V  ← delete both the column and row that the minimal value belongs to;
10  end
```

图 4-11 匹配策略伪代码

具体的匹配策略是：属于发送集合的节点计算自己与每个接收节点的“距离”，形成距离矩阵 V，矩阵中每个值对应于发送节点和接收节点“距离”（伪代码 1-4），从 V 中选取最小距离值并更新矩阵（伪代码 5-7），直到矩阵为空矩阵。图 4-12 是匹配策略简单示意图，m=7，发送节点集合{N80, N56}，接收节点为{N64, N48}。发送节点 N56 和接收节点 N48 距离值 8 是最小值，因此节点 N56 和 N80 分别向 N48 和 N64 传输数据。



V	N80	N56
N64	16	2^7- 8
N48	32	8

图 4-12 匹配过程示意图

§4.5 本章小结

本章首先介绍了现有的数据调度方案的特点，深入分析了“推”和“拉”两种数据传输方式的优缺点：“推”数据分发方案传输延时小，控制消息开销小，节点带宽利用率不高；“拉”数据分发方案能够充分利用节点带宽，能够很好地适应 P2P 系统节点的高度动态性，但传输延时大，控制消息开销大。

针对“拉”数据传输方案传输延时大的缺点，提出了基于优先级的数据分发方案(DPC)。DPC 是一种分布式的数据分发算法，节点根据邻居节点状态信息做出数据分发决策。DPC 算法中对节点和数据块进行了优先级的划分，优先响应具有高优先级节点请求，传输优先级高的数据块。针对同构网络环境和异构网络环境，我们分别分析了数据分发延时最小理论值并证明了 DPC 算法可以实现数据块的最快分发。同时，DPC 算法能够按序地传输数据块，这一点表明 DPC 算

法特别适合流媒体应用连续性要求。最后实验仿真表明，本章提出的 DPC 算法在数据传输延时、启动延时和丢包率方面都优于现有的数据分发方案。

DPC 算法的缺点是控制消息开销要比已有的数据分发方案大，牺牲控制消息开销换取数据块分发几何级数增长，在深入分析了 DPC 算法消息延时开销，提出了同构环境下基于“推”的 DPC 算法改进方案。改进算法是基于发送端的，数据发送完全由发送端控制，大大减少了消息延时开销。基于“推”的 DPC 算法改进方案目前还只是适合于同构网络环境，在异构网络环境下节点具有不同的带宽，数据推向不同带宽的接收节点对直接影响了数据分发延时，如何将改进算法推广到异构网络环境下是我们下一步研究工作。

第五章 P2P 点播系统带宽分配算法研究

相比成熟的 P2P 文件共享和实时流媒体技术，P2P 点播技术还处于发展阶段[41]。P2P 点播系统同样是由覆盖网络和数据调度两个方面组成。但是 P2P 点播具有自己的特点，首先 P2P 点播系统中节点的播放是不同步的，节点可以收看任何时间段的视频。而 P2P 实时流媒体中，节点之间播放是半同步的，播放延时差距小。其次 P2P 点播系统支持用户的 VCR(Video Cassette Recorder)操作，用户可以任意地拖动播放进度条，选择自己喜欢的视频片段，在 P2P 实时流媒体中不存在这种交互性，用户使被动的接收者。这两个显著特点使得 P2P 点播系统对用户有更大的吸引力，同样也使得实时流媒体技术不能简单地应用到点播系统，带来了许多技术挑战。在本章中，我们深入分析了 P2P 点播系统中引起服务器负载的各种因素，并进一步提出了降低服务器负载的带宽分配方案。

§5.1 现有 P2P 点播系统数据分发方案

由于 P2P 点播系统中用户的交互性，节点可以选择自己希望获取的数据，因此基于发送端的“推”数据分发方案不适合 P2P 点播系统。目前广泛部署的 P2P 点播系统都采用了“拉”的数据分发方案，即接收者根据自己的需要向数据提供者请求数据，与实时流媒体系统一样，有效地获取数据满足当前的播放需求也是点播系统的基本要求。

实时流媒体中节点播放是半同步的，播放延时相差不大，但点播系统中节点播放是异步的，用户可以随意地拖动播放进度条收看自己感兴趣的视频片段。一个正在收看视频初始部分的节点不能为一个收看视频结尾部分的节点提供任何帮助，同时由于节点缓存有限，播放到视频结尾的节点早已删除了视频初始数据，刚开始播放的节点也无法从播放到视频结尾的节点获取数据，因此，节点之间的播放不同步减弱了彼此之间相互合作，大大增加了点播系统中服务器的负载。为了提高节点之间的相互合作，进一步降低服务器负载，在 P2P 点播系统中，除了内存中的缓存，节点还必须贡献一部分硬盘空间来弥补缓存空间的不足。节点将自己播放过的数据存放在硬盘空间，这样刚开始播放的节点就可能从播放至结尾的邻居节点处获取视频初始数据。因此，现有的点播系统数据分发方案研究的一个热点是如何有效地利用硬盘空间缓存数据，增加节点之间的相互合作进而减少服务器负载。本节中将结合已有的 Gridcast 和 PPLive VOD 系统说明点播系统数据分发方案。

§5.1.1 复制策略

在点播系统中，为了增加节点之间相互帮助机会，除了内存中的缓存以外，每个节点还贡献一部分硬盘空间缓存播放过的数据。PPLive VOD 将硬盘空间大小设置为 1GB。复制策略研究的是如何有效地利用硬盘空间缓存数据以提高节点之间数据块请求的命中率。

Gridcast 采用了 SVC(single video caching)方式管理硬盘空间。SVC 只是在硬盘上缓存正在播放的影片，当用户切换频道时，原先频道的数据将被删除。PPLive VOD 采用了 MVC(multiple video caching)方式，硬盘上可以缓存多部影片。硬盘空间是有限的，当缓存的数据超过分配的硬盘空间时，需要决策删除一些数据。传统的方案有 LRU(least recently used)和 LFU(least frequently used)，LRU 是指删除最长时间没有被请求的数据块，LFU 是指替换那些被请求次数最少的数据块。PPLive VOD 使用了一个基于权重的删除策略，中心服务器基于以下两个因素对缓存的数据进行评估：1）整个影片视频的完整性；2）影片视频在网络中的需求，也可以说是流行度。假设某部影片被缓存于 c 个节点，当前有 n 个节点正在收看这部影片，则这部影片在网络中的需求为 c/n。实际系统测试表明基于权重的删除策略比 LRU 能够进一步的提高节点数据请求命中率，增加了节点之间的相互合作，降低服务器负载。基于权重删除策略的缺点是中心服务器需要获取影片在网络中需求率，增加了系统控制消息开销。

§5.1.2 数据传输策略

P2P 点播系统中的数据传输策略与实时流媒体类似，目的是保证数据播放的连续性。在流媒体中，每个数据块都有相应的播放时间，如果在规定的播放时间没有接收到数据块，播放器不得不暂停或者跳过这个数据块，从而影响了播放的连续性。传统的数据传输策略主要有贪心策略和稀有性优先策略，贪心策略是指优先请求距离播放时间最近的数据块，从单个节点的角度，贪心策略似乎是最优的，可以降低启动延时，特别适合流媒体应用，但从系统的角度，这种算法降低了数据块在节点分布的多样性，减弱了节点之间的相互合作，从而影响了整个系统的性能。稀有性优先策略优先请求在网络中稀有的数据块，避免了数据块在网络中丢失的可能，具有更好的数据块分布多样性，但是为了获得连续的数据块，稀有性优先的数据分发策略增加了启动延时。

由于 P2P 点播系统支持 VCR 操作，用户可以任意地拖动播放器进度条选择自己喜欢的视频片段。传统的数据分发算法并不能很好的适应这种 VCR 操作，

如何有效地减低 VCR 操作带来的启动延时，提高用户的收看体验是当前的一个研究热点。Gridcast 提出了一种基于锚定点的预取算法[42]。其基本思想是：整个视频文件设置多个锚定点，在满足正常播放的前提下，节点利用剩余带宽优先预取锚定点处数据。当节点进行 VCR 操作时，将节点的跳跃点重定位到距离最近的锚定点，由于预先获取了锚定点的数据，这样就可以降低节点的 VCR 操作延时，同时减少了对服务器的数据请求。图 5-1 描述了 VCR 重定向操作。

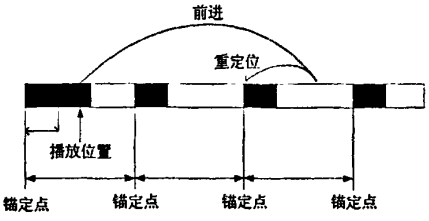


图 5-1 VCR 操作重定向

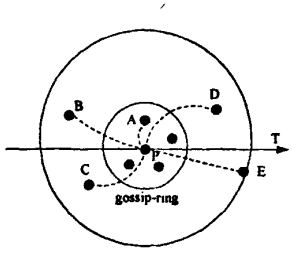


图 5-2 邻居节点环分布

为了支持 VCR 操作，除了在数据调度方案进行相应的改进，点播系统同样修改了覆盖网络的构建方案。Gridcast 采用了一种环形拓扑结构来支持 VCR 操作[43]。基本思想是：根据自己的播放位置，以自己为圆心，将邻居节点分布在逻辑环上。与自己播放位置相差不多并且缓存区存在重叠的节点放置在内环。由于节点与内环里的邻居节点周期性交互流言消息，内环也称为流言环(gossip-ring)。外环上维护的是那些播放位置超前或者落后，缓存区不存在重叠的节点。当节点进行 VCR 操作时，如用户向前拖动播放进度条，外环上的播放超前的节点就可以充当新的父节点，避免了节点的重新选择和向服务器请求数据，有效地降低了 VCR 操作延时和服务器负载，因此外环也称为跳跃环(skip-ring)。图 5-2 说明了节点 P 的邻居节点在环上分布。邻居节点 A 是属于节点，可以从节点 A 获取数据。邻居节点 B，C，D 和 E 属于外环节点，B 和 C 又属于后向节点，D 和 E 属于前向节点。当节点 P 进行前向操作时，节点 D 和 E 不仅可以提供相关数据，还可以帮助节点 P 快速查找系统中播放位置相近的其他节点。

综上所述，由于节点 VCR 操作和播放异步特点，点播系统中节点之间相互合作的机会少，节点带宽不能够得到充分利用，服务器负载压力大。复制策略、数据分发策略以及覆盖网络的构建目的都是为了提高节点之间数据请求命中率，增加节点之间相互合作的机会，降低服务器负载压力。基于此，我们将提出一种基于预测的带宽分配方案 PBA(predictor-based bandwidth allocation)，其核心思想是将数据分发问题转化为带宽分配问题，利用额外带宽帮助落后节点预取数据块，降低离开丢失导致的视频数据丢失，从而减少服务器负载。

§5.2 基于预测的带宽分配方案

§5.2.1 服务器负载分析

我们首先分析点播系统中造成服务器负载的因素，为实际的算法设计提供指导方向。GridCast 系统实际测量表明 P2P 点播系统中服务器负载主要由以下因素造成[44]：

- 带宽瓶颈：邻居节点拥有本节点需要的数据，但是邻居节点由于带宽不足不能够为本节点提供服务，为了保证播放的连续性，本节点必须向服务器请求数据。
- 内容瓶颈：邻居节点有剩余带宽，但是没有本节点需要的数据块。内容瓶颈主要由两个因素引起，一是可能由于缓存空间有限，缓存的数据块已经被邻居节点删除了；二是邻居节点本来就没有获取这样的数据块。
- 新数据块：只有源节点拥有新数据块，只能从源节点处获取。
- 离开丢失：由于节点的动态性，导致了一些数据块在网络中消失，其他节点只有从源节点处获取丢失的数据块。

图 5-3 是离开丢失示意图。S 表示视频服务器，节点 1，2 和 3 参与流媒体会话。视频被划分为数据块，数据块 6-8 仅存在于节点 1。如果节点 1 离开网络，数据块 6-8 将从当前网络中消失，节点 2 和 3 必须向服务器请求这些数据块。

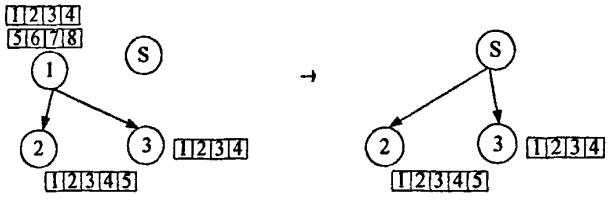


图 5-3 离开丢失示意图

§5.2.2 集中式带宽分配算法

集中式算法根据节点的到达时间和带宽构造数据分发网络，具体的算法过程可分为三个阶段：(1) 等级排序机制；(2) 基本传输要求；(3) 分配额外带宽。

(1) 等级排序机制

根据节点的到达时间和上传带宽对节点进行排序。首先越早加入网络的节点，优先级越高；同一时间到达的节点，带宽越大优先级越高。在不考虑 VCR 操作的情况下，所有节点从视频初始开始请求数据，晚些时候加入系统的节点只

能从较早加入的节点或者服务器获取数据。因此，集中式算法将节点组织成一个类似“树”的覆盖网络，优先级越高的节点距离源节点越近。

(2) 基本传输要求

在流媒体传输中，为了保证视频播放连续性，每个节点的接收速率不能低于视频编码速率。由于节点加入时间的差异，低优先级的节点只能够向高优先级邻居节点请求数据。对系统中 n 个节点进行排序，节点 1 是最早到达的节点，而节点 n 是最新加入节点。节点 i 首先向较早加入的节点 i−1 请求数据，如果节点 i−1 不能够提供与视频编码速率相等的接入带宽，节点 i 必须向更早加入的节点 i−2 请求数据，如此反复，直到节点的接收速率等于视频编码速率为止。

(3) 分配额外带宽

在所有节点获得基本传输速率后，若存在额外带宽，节点自顶向下地贡献额外带宽，帮助子节点预取数据，弥补不同优先级节点在数据缓存上的差异。额外带宽分配策略类似于[45]中的贪心策略。图 5-4 是额外带宽分配算法伪代码。l_k 表示在节点 k 的剩余带宽；g_k 表示节点 k 在满足了基本传输需求后从邻居节点处获得的额外带宽；b_k 表示节点的缓存状况。额外带宽自顶向下的分配给子节点，当父子节点的缓存状况相同时，子节点获取的额外带宽不应超过父节点的额外带宽。

```
surplus =0 ;
for k=1:n-1 do
    surplus=surplus+l_k ;
    if b_k=b_k+1 and g_k < surplus then
        g_k+1= d_k+g_k−d_k+1 ;
    else
        g_k+1= surplus ;
        surplus= surplus- g_k+1 ;
return
```

图 5-4 集中式带宽分配

图 5-5 说明了集中式带宽分配算法。S 是源节点，节点 1 最早加入系统，节点 2 和 3 同时加入，节点带宽分别为 1.5R, 0.75R 和 0.5R，R 是视频编码速率。(a)是没有考虑节点优先级的带宽分配结果，节点 1 优先选择节点 3 传输数据，节点 2 首先从节点 3 获取数据，由于节点 3 的带宽不能满足基本传输需求，节点 2 继续向节点 1 请求数据。在满足基本需求后，节点 1 没有剩余带宽。(b)中根据节点带宽划分了优先级，节点 1 首先满足优先级较高的节点 2 基本需求，由于节点 2 的带宽不能满足节点 3 的基本需求，节点 3 需要向节点 1 请求数据。在满足了所有节点的基本需求后，节点 1 仍然有 0.25R 的剩余带宽帮助节点 2 预取数据。

图 5-5 集中式分配算法示意图

§5.2.3 PBA 算法

相比集中式带宽分配方案，PBA 算法是分布式的，不需要集中式的服务器维护所有节点信息，可扩展性好。其基本思想是：在每个调度周期，节点根据邻居节点的缓存状况和估计带宽请求数据，邻居节点收到请求后根据请求节点的优先级分响应数据请求，分配带宽。PBA 算法是一种“拉”数据分发方案，它包括接收端调度和发送端调度两部分。

(1) 接收端调度

数据调度被划分为调度周期δ，节点在每个调度周期初始与邻居节点交换缓冲区映像，节点缓存可以用一个滑动窗口来表示，滑动窗口记录了节点已接收的数据块和需要请求的数据块，节点根据邻居节点的缓存状况和带宽分配数据块请求。图 5-6 是滑动窗口示意图，已接收到的数据用灰色表示，白色表示相应的数据块没有获取。滑动窗口分为紧急区间和预取区间两部分，紧急区间存放的是即将需要播放的数据，由于流媒体连续性特点，属于紧急区间的数据应当优先请求。预取区间存放的数据在当前的调度周期并不是急需的。播放位置与紧急区间之间的间隔是启动区间，也就是节点选择频道到开始播放的时间间隔。由于流媒体连续性的特点，每个数据块对应相应的播放时间，启动区间没有接收到的数据块（如数据块 14）将“错过”其相应的播放时间，所以节点不请求启动区间丢失的数据块，滑动窗口在每调度周期向前滑动一个紧急区间窗口大小。

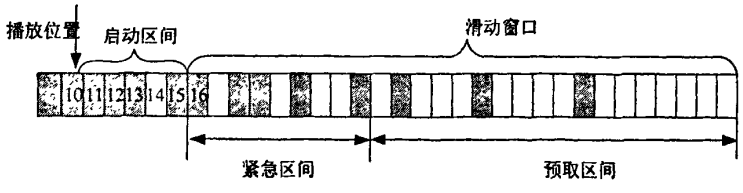


图 5-6 滑动窗口示意图

具体的数据调度过程类似于 CoolStreaming 的稀有性优先算法，其主要思想是节点计算出每个数据块的拥有节点数，并根据历史信息估计每个邻居节点在当前调度周期可以提供的带宽。如果某个数据块只有一个邻居节点拥有，直接向这

个邻居节点请求；如果多个邻居节点拥有这个数据块，优先向可用带宽最大的邻居节点请求。U^1_{ki}，U^2_{ki}，...U^m_{ki}表示在过去 m 个调度周期节点 i 从邻居节点 k 获得数据块数，则当前调度周期邻居节点 k 可以提供的带宽为：$U_{ki}^{m+1}=\theta(\sum_{l=1}^{m}U_{ki}^{l}/m)$，$\theta>1$，调节因子$\theta$影响着带宽估计的精确度。

(2) 发送端调度

发送节点在一个调度周期内收到多个邻居节点的数据请求，因此需要合理地分配带宽。类似于集中式的带宽分配方案，发送节点首先满足所有邻居节点来自紧急区间的数据请求，这等价于集中式方案中的第二步。若存在剩余带宽，发送节点根据邻居节点的当前播放位置和稳定性对邻居节点划分优先级，优先满足高优先级邻居节点来自预取区间的数据请求。我们根据以下标准划分优先级：首先判断节点稳定性，若存在多个稳定节点，根据节点的播放位置区分。单纯地根据播放位置或者稳定性进行优先级划分都是有缺点的：一个有较大播放位置节点很可能是 VCR 操作的结果，若节点不属于稳定节点，即随时可能离开网络，将剩余带宽分配这个节点造成了不必要的浪费；同理，将剩余带宽分配给一个稳定节点，不考虑节点的播放位置，带宽就有可能浪费在帮助播放位置落后稳定节点，影响稀有数据块的分发。表 5-1 是符号定义说明，图 5-7 是发送端调度伪代码。

(3) 节点稳定性预测

实际系统测量表明节点的稳定性与其在系统中存活时间相关。节点的存活时间服从帕累托分布，节点越早加入网络，收看视频时间越长，在网络中继续存活下去的概率越大。因此，我们将存活时间作为节点稳定性判据，当节点存活时间超过门限值Δ，可以认为这个节点是稳定节点，将继续在网络中逗留；否则节点属于不稳定节点，随时都可能离开网络。

表 5-1 符号定义说明

emergence_set	存放来自邻居节点紧急区间的请求
prefetch_set	存放来自邻居节点预取区间的请求
chunk_size	表示数据块大小
SL	具有最大播放位置的稳定节点

```
R_i : set of receivers
U_i : upload bandwidth of supplier i
/* serve emergent requests first, k is chunk sequence number, v is the receiver requesting
chunk k */
for (k,v) in emergency_set:
    if  U_i > chunk_size :
        upload chunk k to receiver v
        U_i -= chunk_size
    if  U_i < chunk_size :
```

```
        break
end
/* serve pre-fetch request, if supplier i has extra bandwidth */
if U_i > chunk_size :
   if (SL≠ Ø):   /* break tie randomly */
      for (i, RL) in prefetch_set
         upload chunk i to receiver SL
         U_i -= chunk_size
         if U_i < chunk_size :
            break
      end
      if U_i > chunk_size  : /* serve pre-fetch requests from others */
         randomly select pre-fetch requests to server until all requests are served or
         U_i < chunk_size
   else   /* there are no stable receivers */
      randomly select pre-fetch requests to server until all requests are served or
      U_i < chunk_size
end
```

图 5-7 是发送端调度伪代码。

§5.3 仿真与性能分析

仿真中视频长度为 1800 秒，每个数据块为 1 秒视频数据，即视频速率为一个数据块/秒，节点到达服从泊松分布，平均间隔为 60 秒，节点带宽是视频速率的 1-4 倍，系统平均带宽为视频速率的 2 倍。节点逗留时间服从帕累托分布，平均逗留时间为 500 秒，数据调度周期为 10 秒。节点根据过去三个调度周期历史记录估计邻居节点在当前调度周期的可用带宽，调节因子 θ=1.2，稳定性门限 Δ=500 秒。

§5.3.1 参数设置

我们首先检测缓存空间的大小对算法性能的影响，在 PBA 算法中节点滑动窗口分为紧急区间和预取区间两部分。紧急区间大小等于一个调度周期时间的视频数据，仿真中其大小为固定值 10 个数据块。图 5-8 表示了不同预取区间大小对服务器负载的影响。预取区间表示为紧急区间的倍数关系，服务器负载用从服务器获取的数据块数表示，服务器负载进一步被分为新数据块，带宽瓶颈（包括带宽和内容瓶颈）和离开丢失三类。随着节点预取区间增大，可以请求的数据范围越广，越有可能获得稀有数据块，从而降低离开丢失现象和服务器负载。但是

过大的预取区间并没有进一步降低服务器负载，6 倍大小的预取区间相比 5 倍大小对服务器负载的改善很小。这是因为在满足了基本需求后，系统中剩余带宽是有限的，限制了节点获得更多稀有数据块。由于每个节点总是优先满足邻居节点的基本需求，预取区间的增大并没有加剧带宽瓶颈现象。

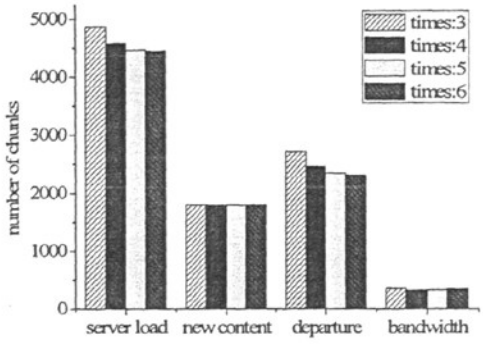


图 5-8 预取区间与紧急区间比例关系

节点的稳定性判断依赖门限值的选取，图 5-9 表示了稳定性门限值对服务器负载的影响。当门限值过大时，只有少数节点被识别为稳定节点，减弱了稀有数据的获取，增加了离开丢失现象；门限值偏小时，一些不稳定节点也被识别为稳定节点，与真正的稳定节点竞争剩余带宽，从而造成了带宽浪费，同样也影响了稀有数据的获取。仿真中我们发现门限值接近平均逗留时间 500 秒时服务器负载最小。

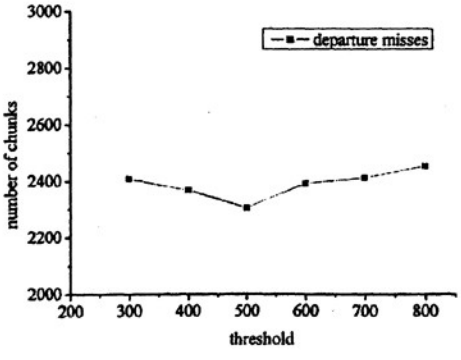


图 5-9 门限值的选取

§5.3.2 性能分析

本节首先检测在不同系统资源下 PBA 算法的性能，PBA 算法采用了 5.3.1 节参数设置中的最佳值，即预取区间是紧急区间的 6 倍，稳定性门限值为 500

秒。图 5-10 表示了在不同系统资源下 PBA 算法的性能，系统平均带宽分别是视频速率的 1.5 倍，2 倍和 2.5 倍。可以看出随着系统资源的增加，PBA 算法能够进一步降低离开丢失引起的服务器负载，这是因为稳定节点可以从父节点处获取更多稀有数据。同时随着节点可用带宽的增加，带宽瓶颈引起的服务器负载也进一步降低。

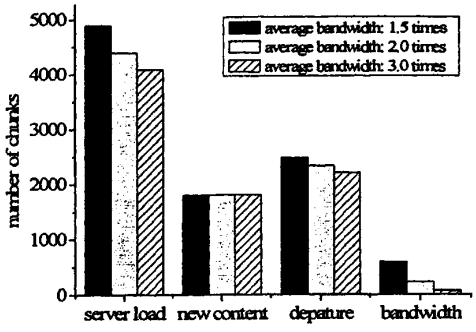


图 5-10 不同系统资源下 PBA 算法的性能

我们都 PBA 算法与集中式、传统的带宽分配方案进行了比较，传统分配方案[42]中没有考虑节点的优先级，带宽近似均匀地分配。PBA 算法同样采用了 5.3.1 节参数设置中的最佳值，即预取区间是紧急区间的 6 倍，稳定性门限值为 500 秒，系统平均带宽是视频速率的 2 倍。图 5-11 表示了三种带宽分配方案的性能。集中式方案的服务器负载最低，与传统分配方案相比，PBA 方案能够减低 8%的服务器负载，PBA 算法降低了离开丢失引起的服务器负载，同时却增加了带宽瓶颈现象。这是因为调节因子θ>1，稳定节点的预取消耗了邻居节点的剩余带宽，增加了邻居节点拥塞的可能性，当节点的基本需求无法得到满足时，只有向服务器请求数据。

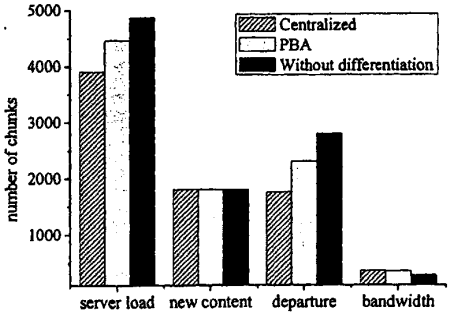


图 5-11 三种带宽分配方案比较

§5.4 本章小结

本章首先介绍了 P2P 点播系统中数据调度策略，与 P2P 实时流媒体不同，点播系统的 VCR 操作和播放异步降低了节点之间的合作，增加了服务器负载，影响了系统的可扩展性。为了降低点播系统服务器负载，深入分析了服务器负载的各种因素。

在此基础上提出了基于预测的带宽分配方案 PBA，其基本思想是将数据分发问题转化为带宽分配问题，利用额外带宽帮助落后节点预取数据块，从而降低离开丢失引起的视频数据丢失，减少服务器负载。PBA 方案包括接收端和发送端调度两个部分，接收节点根据发送节点缓存状况和可用带宽优先请求紧急区间数据块，满足基本的播放要求；发送节点在满足了基本数据传输要求后，若存在剩余带宽，则根据节点的稳定性和播放位置，优先响应稳定节点的预取数据要求。节点的稳定性是根据其在网络中的已存活时间进行判断。

仿真分析表明，本章提出的基于预测的带宽分配方案相比传统的数据分发方案进一步降低了离开丢失引起的服务器负载。但是由于 PBA 算法优先分配剩余带宽给稳定节点，这会影响其他不稳定节点预取数据块的能力，增加了它们 VCR 操作延时。在考虑了带宽的区分分配，如何降低 VCR 操作延时是我们下一步的工作。

第六章 总结和下一步工作计划

§6.1 全文总结

近年来 P2P 技术得到迅速发展，与传统的客户/服务器模式相比，P2P 网络中单个用户既是客户节点，也是服务器节点。P2P 技术利用了闲置在网络边缘用户的资源，提高了系统的可扩展性，使得网络技术向更大规模发展。各种 P2P 文件共享、实时流媒体和点播系统软件大量涌现。相比文件分发，流媒体数据传输具有更高的实时性要求，P2P 流媒体系统需要面对如下三个问题：1) 如何构建有效的覆盖网络将节点组织起来；2) 如何高效地分发视频数据，满足连续性播放要求，提高用户观看体验；3)优化数据调度算法充分利用节点带宽，进一步降低服务器负载，提高系统可扩展性。论文的主体三章针对这三个方面进行展开。

论文第三章深入分析了 P2P 流媒体两个关键组成：节点组成员管理和数据调度机制。节点组成员管理帮助新节点加入 P2P 网络，构建覆盖网络适应节点的高度动态性。在覆盖网络上设计数据分发机制高效地分发数据，满足流媒体应用需求，适应由于节点动态性带来的网络结构变化。然后从这两个方面分析了典型的 P2P 实时流媒体和点播系统。

论文第四章深入分析了现有的 P2P 流媒体数据调度方案，针对“拉”数据调度方案传输延时大的缺点，提出了基于优先级的数据调度方案 DPC。DPC 是一种分布式的“拉”数据调度算法，节点根据邻居节点状态信息做出数据分发决策。DPC 算法由发送端和接收端调度两部分组成，在对节点和数据块划分优先级后，发送节点优先响应优先级高的邻居节点请求，接收节点优先请求优先级高的数据块。这种严格按照优先级的数据传输策略保证了数据分发的几何级数增长，从而最小化数据传输延时。但是严格的优先级传输策略需要节点之间交互大量控制消息，在深入分析了 DPC 算法消息延时开销，提出了同构环境下基于“推”的 DPC 算法改进方案。改进方案是基于发送端的，数据发送完全由发送端控制，大大减少了消息延时开销。

与实时流媒体不同，点播系统中 VCR 操作和播放异步降低了节点之间的合作，增加了服务器负载，影响了系统的可扩展性。论文第五章首先深入分析了 P2P 点播系统中服务器负载因素，在此基础上提出了基于预测的带宽分配方案 PBA，PBA 算法将数据分发等价为带宽分配问题，利用额外带宽帮助落后节点预取数据块，从而降低离开丢失引起的视频数据丢失，减少了服务器负载。

§6.2 下一步工作

作为本论文工作的延伸，有如下内容值得进一步研究。

- 论文第四章提出的基于优先级的数据调度算法 DPC 需要节点之间交互大量的控制消息保证数据严格按照优先级分发，基于“推”的 DPC 算法改进方案目前还只是适合于同构网络环境，在异构网络环境下节点具有不同的带宽，数据推向不同带宽的接收节点直接影响了数据分发延时，如何将改进算法推广到异构网络环境下是我们下一步研究工作。
- 由于 PBA 算法优先分配剩余带宽给稳定节点，这会影响不稳定节点预取数据块的能力，在 P2P 点播系统中存在节点 VCR 操作，由于不稳定节点没有预取足够的将来数据块，VCR 操作后会带来较大的启动延时，从而影响了用户的观看体验。在考虑了带宽的区分分配，降低了服务器负载，如何降低 VCR 操作延时是很值得研究的问题。
- 我们的研究几乎都避开的数据分发网络与底层拓扑之间的匹配问题，覆盖网络与底层拓扑匹配度直接影响了 P2P 流媒体系统性能。在下一步工作中，希望在实际的 P2P 流媒体系统对算法进行性能评估。

参考文献

[1] http://www.pplive.com/

[2] http://www.ppstream.com/

[3] http://tv.qq.com/

[4] http://www.uusee.com/

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

[6] Kangasharju, Jussi.A.T. Internet Content Distribution. PhD thesis, April 2002.

[7] http://www.bittorrent.com/

[8] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Yum Y.-S.P, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming", Proc. of IEEE INFOCOM, 2005

[9] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang, "Towards Cinematic Internet Video-on-Demand" In Proceedings of Eurosys 2008,

[10] S. Li, F. Wu, and Y. Q. Zhang. "Study of a New Approach to Improve FGSVideo Coding Efficiency", ISO/IEC JTC1/SC29/WG11, MPEG99/M5583, Dec. 1999

[11] W. Li. "Bit-Plane Coding of DCT Coefficients for Fine Granularity Scalability", ISO/IEC JTCI/SC29/WG11, MPEG98/M3989, Oct. 1998

[12] Nazanin Magharei, Reza Rejaie, "Adaptive Receiver-Driven Streaming from Multiple Senders", ACM/SPIE Multimedia Systems Journal, Volume 11, Issue 6, pp. 550-567, Springer-Verlag, April 2006

[13] R. Puri, K. Ramchandran, K. W. Lee and V. Bharghavan, "Application of FEC based Multiple Description Coding to Internet Video Streaming and Multicast", In Proc. of Packet Video Workshop, Cagliari, Sardinia, Italy, May 1-2, 2000.

[14] Y. Wang, M. T. Orchard and A. R. Reibman, "Multiple Description Image Coding for Noisy Channels by Pairing Tansform Coefficients", In Proc. of IEEE Workshop on Multimedia Signal Processing, pp. 419-424, June 1997.

[15] M. Castro, P. Druschel, A. M. Kermarrec etal, "SplitStream: High-bandwidth Content Distribution in Cooperative environments", IPTPS'03, Berkeley, CA, Feb. 2003.

[16] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow", IEEE Trans. Inform. Theory, vol. 46, pp. 1204-1216, July 2000.

[17] C Gkantsidis, PR Rodriguez, "Network coding for large scale content distribution", Proceedings of IEEE INFOCOM 2005

[18] PA Chou, Y Wu, K Jain, "Practical network coding", Proceedings of the ANNUAL ALLERTON CONF ON COMMUNICATION, CONTROAL AND COMPUTING, 2003

[19] E. Adar and B. A. Huberman, "Free Riding on Gnutella", *First Monday*, 5(10), October 2000.

[20] A. Habib and J. Chuang, "Incentive Mechanism for Peer-to-Peer Media Streaming", Proc of International Workshop on Quaility of Service (IWQoS) 2004.

[21] Guang Tan, and Stephen A. Jarvis, "A Payment-based Incentive and Service Differentiation Mechanism for Peer-to-Peer Streaming Broadcast", Proceeding of IWQoS, 2006

[22] Zhengye Liu, Yanming Shen, Shivendra S. Panwar, Keith W. Ross and Yao Wang, "Using Layered Video to Provide Incentives in P2P Live Streaming", Proceeding of SIGCOMM,

2007

[23] J.J.D. Mol, J.A. Pouwelse, M. Meulpolder, D.H.J. Epema, and H.J. Sips, "Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems", Proceedings of the Multimedia Computing and Networking, 2008

[24] J Liang, N Naoumov, KW Ross, "The index poisoning attack in P2P file-sharing systems", Proceedings of IEEE INFOCOM, 2006

[25] Grizzard JB, Sharma V, Nunnery C. "Peer-to-Peer botnets: Overview and case study", Proceeding of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots 2007)

[26] 诸葛建伟, 韩心慧, 周勇林, 叶志远, 邹 维, "僵尸网络研究", 软件学报, March, 2008

[27] Xin Sun, Ruben Torres and Sanjay Rao, "Preventing DDoS Attacks with P2P Systems through Robust Membership Management", POSTER in The 8th Annual CERIAS Information Security Symposium, 2007

[28] A. Ganesh, A. Kermarrec and L. Massoulie, "Peer-to-peer membership management for gossip-based protocols", IEEE Transactions on Computers, Vol. 52, No. 2, pp. 139-149, 2003

[29] I. Stoica, R. Morris, D. Karger etal. "Chord: A scalable peer-to-peer lookup service for internet applications". In Proc. of the ACM SIGCOMM'01 Conference, 2001

[30] S. Ratnassamy, P. Francis, M. Handley etal. "A scalable content-addressable network", In Proc. of the ACM SIGCOMM'01 Conference, 2001

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

[32] Y. B. Zhao, J. Kubiatowicz and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing", Technical Report UCB/CSD-O1—1141, University of California at Berkeley, Computer Science Department, 2001.

[33] F. Wang, Y. Q. Xiong and J. C. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast", In Proc of ICDCS, pp. 49-49, 2007

[34] Y. H. Chu, S. G. Rao, S. Seshan, H. Zhang. A case for end system multicast. ACM SIGMETRICS Performance Evaluation Review, Vol. 28, No. 1, pp. 1-12, 2000

[35] S. Banerjee, B. Bhattacharjee, C. Kommareddy. Scalable application layer multicast. ACM SIGCOMM Computer Communication Review, Vol. 32, No. 4, 2002.

[36] Xiaojun Hei, Yong Liu and Keith Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems", IEEE JSAC special issue on advances in P2P streaming, Volume 25, Number 9, December 2007

[37] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu and Keith Ross, "A Measurement Study of a Large-Scale P2P IPTV System", IEEE Transactions on Multimedia, Volume 9, Number 8, December, 2007

[38] Yong Liu, "On the Minimum Delay Peer-to-Peer Video Streaming: how Realtime can it be?", in the Proceedings of ACM Multimedia, September, 2007.

[39] Thomas Bonaldy, Laurent Massoulié, Fabien Mathieuy, Diego Perinoy, Andrew Twigg, "Epidemic Live Streaming: Optimal Performance Trade-Offs", In Proc of SIGMETRIC, 2008

[40] L. Massouli_e, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms", In Proc of INFOCOM, 2007

[41] Yan Huang, Tom Z. J. Fu, Dah-Ming Chiu, John C. S. Luiand Cheng Huang, "Challenges,

Design and Analysis of a Large-scale P2P-VoD System", In Proc of SIGCOMM, 2008

[42] Bin Cheng, Xiuzheng Liu, Zheng Zhang, Hai Jin, Lex Stein, Xiaofei Liao, "Evaluation and optimization of a peer-to-peer video-on-demand system", Journal of Systems Architecture: the EUROMICRO Journal, Volume 54, Issue 7, July, 2008

[43] Bin Cheng, Hai Jin, Xiaofei Liao, "Supporting VCR Functions in P2P VoD Services Using Ring-Assisted Overlays", Proceedings of the IEEE International Conference on Communications 2007 (ICC'07), Glasgow, Scotland, June 24-28, 2007

[44] Bin Cheng, Lex Stein, Hai Jin, and Zheng Zhang, "A Framework for Lazy Replication in P2P VoD", in Proceedings of NOSSDAV 2008

[45] Cheng Huang, Jin Li, Keith W. Ross, "Peer-Assisted VoD: Making Internet Video Distribution Cheap", in Proc. of IPTPS, February 2007

攻读硕士期间发表的论文

1. Zhengjun Chen, Kaiping Xue and Peilin Hong, "A Study on Reducing Chunk Scheduling Delay for Mesh-based P2P Live Streaming", in the Proceeding of IEEE conference on Grid and Cooperative Computing (GCC 2008), October 2008.

2. Kaiping Xue, Peilin Hong, Zhengjun Chen and et al, "KAP-AOT: A Novel Group Key Agreement Protocol Based on Considering Nodes' Average Online Time", in the Proceeding of IEEE conference on Grid and Cooperative Computing (GCC 2008), October 2008.

3. Zhengjun Chen, Kaiping Xue, Peilin Hong and Hancheng Lu, "Differentiated Bandwidth Allocation for Reducing Server Load in P2P VOD", submitted to IEEE conference on Grid and Cooperative Computing (GCC 2009).

致谢

首先感谢我的导师洪佩琳教授。洪老师渊博的学识和严谨认真的治学态度让我受益匪浅。感谢薛开平老师对我研究和工程项目上的帮助。在本实验室的这段经历将是以后引领我向着更高目标前进的宝贵财富。感谢卢汉成教授和刘发林教授对我申请出国所做的帮助，感谢吴忠民老师对我生活上的关心。

已毕业的刘梦娟博士、周晓波博士、张瑞博士、黄冠尧都曾给予我很多鼓励和帮助，在此向他们表示衷心的谢意。尤其要感谢和我有过密切合作的刘梦娟、刘芳林、叶浩、李盼盼、汪玉和朱从州同学，和他们一起讨论给了很大的帮助和启发。感谢我同宿舍的好友，他们让我在科大的生活平添了很多色彩。

感谢实验室所有的同学，他们共同创建了实验室和谐的人际关系，给我留下了美好的回忆，在此向他们表示衷心的谢意。

最后，感谢我的家人一直以来对我默默奉献和无尽关爱，是他们给了我战胜困难的力量。