南京大学

研 究 生 毕 业 论 文

（申请硕士学位）

论文题目：P2P 流媒体系统的若干关键属性的优化方案

作　　者：李振华

院　　系：计算机科学与技术系

专　　业：计算机软件与理论

研究方向：对等网络

指导教师：陈贵海 教授

二OO八年五月

Optimizing Several Critical Properties of P2P Streaming Systems

by

Zhenhua LI

supervised by Prof. Guihai Chen

A dissertation submitted to

the Graduate School of Nanjing University

for the Degree of **Master of Engineering**

Department of Computer Science and Technology

Nanjing University

May, 2008

摘要

P2P流媒体系统在过去几年里用户群急速膨胀、应用面不断拓宽。典型的P2P流媒体系统有Skype、PPLive、PPStream等，其中无论网络音频、网络视频领域均存在数十家公司激烈竞争，这种激烈竞争的局面不仅促进了工业界的技术进步，也为学术界的研究工作提供了价值保证。根据中国互联网络信息中心于2008年1月17日最新发布的第21次中国互联网发展状况统计报告，中国内地网民观看网络视频的几种主要方式中，通过P2P流媒体软件下载的比率达到29.91%，接近三分之一，由此可见，P2P流媒体是一个拥有巨大用户群的领域，也是一个充满研究潜力与意义的领域。

P2P流媒体系统值得研究的属性有很多，从用户体验的角度看有播放连续度、播放数据率、启动时延、源切换时延、带宽利用率、系统容错性等等，从网络设计者的角度看还有定位效率、端到端时延、数据吞吐量、拓扑一致性、负载均衡、系统维护开销、控制开销、可扩展性等等。本文对P2P流媒体系统的3个关键属性（1）播放连续度、（2）源切换时延、（3）系统容错性，提出、设计并实验测试了对应的优化方案。本文的创造性研究成果主要有：

（1）设计了一个具有高播放连续度的P2P流媒体系统ContinuStreaming，采用基于分布式散列表（DHT）的数据预取方法来弥补Gossip协议传播数据的缺陷，从而保证流媒体系统能保持高播放连续度。对ContinuStreaming系统的播放连续度进行了理论分析并将理论分析的结果与模拟实验的结果进行了比较。在多幅真实P2P网络拓扑上所做的大量模拟实验结果表明：相比于当前具有代表性的基于Gossip协议的P2P流媒体系统CoolStreaming而言，ContinuStreaming系统能以低于4%的额外开销带来接近1.0的高播放连续度。

（2）通过对本质特性与关键参数的分析给P2P流媒体系统中数据发布源切换过程建立了数学模型，从而将源切换问题形式化为一个数学优化问题，然后推导出此数学优化问题的最优解。鉴于实际系统情况的复杂性与网络环境的动态性，提出了一个称为快速源切换算法的实用贪心算法，它通过交错旧源与新源的数据传递来趋近理论上的最优解。在多个真实P2P覆盖网拓扑上做的大量模拟实验的结果显示：快速源切换算法相比传统源切换算法能节省20%-30%的切换时间，同时不会带来额外的通信开销，并且切换时间的减少比例随着系统规模的增大而趋于增加。

（3）提出分点这一概念来描述P2P流媒体系统的底层覆盖网的拓扑关键点，然后基于分点的概念，设计了一套简单、有效、分布式的分点检测和消除方法来优化覆盖网拓扑结构。模拟实验的结果表明：设计的方法能有效地检测并消除分点，使系统对覆盖网分割的抵抗力得到本质的增强；分点消除后，系统查询成功率有所上升，在高动态性网络环境下系统的容错性得到明显提高。

关键词：P2P网络，P2P流媒体系统，Gossip协议，关键属性，播放连续度，数据调度算法，数据预取算法，源切换时间，快速源切换算法，底层覆盖网，分点，拓扑优化，系统容错性

Abstract

In recent years, P2P streaming systems have been widely used by a great number of Internet users. Typical P2P streaming systems include Skype, PPLive, PPStream and so on. Both in the areas of Internet audio and Internet video, there exist tens of companies competing severely. The competition situation is not only accelerating technology development in IT industry, but also providing guaranteed value for the research work in academia. According to the 21st statistics report of the development status of China Internet lately published by the China Internet Network Information Centre, among the several primary ways in which China Internet users watch Internet video programs, downloading through P2P streaming software occupies a ratio up to 29.91%, which is close to 1/3. Therefore, P2P streaming has become an area which owns a huge user base and an area filled with research potential and significance.

P2P streaming systems have many properties that are worth research work. From the point of user experience, there are some properties such as playback continuity, playback data rate, startup delay, source switch time, bandwidth utilization, system fault tolerance, etc. From the point of system designer, there are some properties such as location efficiency, end-to-end delay, data throughput, topology awareness, load balance, system maintenance overhead, control overhead, scalability, etc. This paper focuses on three critical properties of P2P streaming systems: (1) playback continuity, (2) source switch delay, (3) system fault tolerance. It proposes, designs and evaluates the performance of corresponding optimization schemes for the above three critical properties. The main creative research results of this paper can be summarized as follows:

(1) This paper designs ContinuStreaming, a gossip-based P2P streaming system with high playback continuity. With the help of DHT, data segments which are likely to be missed by the gossip-based data scheduling can be quickly fetched by the on-demand data retrieval so as to guarantee continuous playback. We discuss the results of both theoretical analysis and comprehensive simulations on various real-trace overlay topologies to demonstrate the effectiveness of our system. Simulation results show that ContinuStreaming can increase the playback continuity very close to 1.0 with less than 4% extra overhead.

(2) This paper models the source switch process in P2P streaming systems and formulates it into an optimization problem. Then we propose a practical greedy algorithm that can approximate the optimal solution by properly interleaving the data delivery of the old source and the new source. The simulation results on various real-trace overlay topologies show that our proposed algorithm outperforms the normal source switch algorithm by reducing the source switch time by 20%-30% without bringing extra communication overhead, and the reduction ratio tends to increase when the network scale expands.

(3) This paper suggests the concept of partition node to describe the topologically-critical nodes of the underlying overlay network of P2P streaming systems. The failure of a partition node may potentially lead to overlay partitioning. Then we propose a simple, effective and distributed

method to detect and avoid partition nodes, so as to optimize the overlay topology. The results of simulation show that our method can essentially enhance the system's resilience to overlay partitioning and remarkably improve the fault tolerance of the overly network under a dynamic environment.

Key Words: P2P network, P2P streaming system, Gossip protocol, Critical properties, Playback continuity, Data Scheduling Algorithm, Data Pre-fetch Algorithm, Source switch time, Fast source switch algorithm, Underlying overlay network, Partition node, Topology optimization, System fault tolerance

目录

图目录

表目录

第1章 绪论

1.1 引言：从P2P网络说起

P2P（Peer-to-Peer），中文译为对等网络或对等计算，在不到 10 年的时间里迅速发展成为 Internet 上最新潮的思想、最流行的技术和最具影响力的应用。我们可以随意举出几个在网民中耳熟能详的 P2P 网络软件名称：BT、迅雷、电骡、Skype、PPLive 等等，可以说其中的每一个都曾经或者正在深刻影响着人们上网的方式和体验。在百度公司最新发布的“百度软件排名榜”和 Google 公司最新发布的“Google 电脑软件热搜榜”排名前 50 的软件中，P2P 网络软件分别占 13 和 11 个，其应用领域包括文件共享、网络音频、网络视频、协同计算、虚拟社区等等。从而可以看出：P2P 网络技术已经渗透到绝大多数 Internet 应用领域中来，并且 P2P 网络技术在其中很多领域占据支配性的地位。几年前就有来自多个国家的测量报告指出，P2P 网络流量已占据当前 Internet 超过一半的带宽资源，成为名副其实的“改变 Internet 的新一代网络技术”。

P2P 的思想起源很早，我们用“Google 学术搜索”（http://scholar.google.com）找到最早提及 P2P 的文献发表于 1956 年，从那以后几乎每年都有 P2P 相关的文章，但一直未成为学术界研究热点。任何一种思想、理论或技术的流行通常都需要一个杀手级应用（Killer Application），以一种征服性的力量冲击人类的传统思维。P2P 的杀手级应用正是出现于 1999 年的世界上第一个应用性 P2P 网络 Napster，它最初只是一个 18 岁的美国学生为了方便自己和朋友共享音乐文件而制作的小软件，却创造了在半年时间里拥有 5000 万用户的网络奇迹，向整个世界传达了 P2P 优秀的思想、展现了 P2P 巨大的潜力。

学术的脚步常常先于应用踏入某个领域，又往往在应用之后成为热点，P2P 和 Napster 的关系正是如此。在 Napster 之后，是一系列广泛流行的 P2P 网络软件：Gnutella、KaZaA、BT、电驴/电骡、Skype 等等。P2P 应用的成功，促使学术界于 2001 年开始真正关注和重视 P2P 网络的研究，代表性的事件是关于 Chord、CAN 两大 P2P 覆盖网模型的两篇论文发表于国际网络通信领域顶尖会议 SIGCOMM’01 上。从那以后，全世界的网络、通信、系统会议或期刊上都开始发表 P2P 相关的论文，而 P2P 网络的研究者逐年递增，P2P 网络成为学术界不可忽视的重要研究领域。

作为P2P网络的一个应用子领域，P2P流媒体系统在过去几年里用户群急速膨胀、应用面不断拓宽。典型的P2P流媒体系统有Skype、PPLive、PPStream、QQ直播、新浪UUSee等，其中无论在网络音频还是网络视频领域均则存在数十家公司激烈竞争，这种激烈竞争的局面不仅促进了工业界的技术进步，也为学术界的研究工作提供了价值保证。

根据中国互联网络信息中心（简称CNNIC）于2008年1月17日发布的第21次中国互联网发展状况统计报告，中国内地网民观看网络视频的几种主要方式中，通过P2P流媒体下载软件的比率达到29.91%，接近三分之一，由此可见，P2P流媒体是一个拥有巨大用户群的领域，也是一个充满研究潜力与意义的领域。我们仅以最新的两例来证实P2P流媒体系统的研究为学术界所重视：例一，2007年微软研究院的研究者采用P2P技术辅助MSN视频点播的设计方案和模拟测试结果，发表在SIGCOMM’07会议上；例二，2008年国内PPLive公司和香港中文大学讯息工程系及计算机系合作的关于PPLive视频点播系统的设计方案和真实测量结果，将发表在SIGCOMM’08会议上。

1.2 P2P流媒体系统的概念

流媒体系统（Media Streaming System，或简称 Streaming System）由来已久，其基本含义就是将流媒体数据（主要是音频和视频数据）从系统中一个结点传播到其它一个、多个或所有结点，所以流媒体系统也常常被称为“网络多播系统”。从功能上看，流媒体系统一般可分为两类：实时流媒体系统（Live Streaming），和流媒体点播（常称为视频点播，即 VoD：Video on Demand）。从工作方式上看，传统流媒体系统基本采用客户/服务器方式（如 YouTube 一类的播客网站）或树状多播，近几年出现的基于 Gossip 协议的网状多播是随 P2P 网络的兴起而出现的，可谓独辟蹊径，一跃成为流媒体系统最新颖独特和最具影响力的第三种工作方式。

现存 P2P 流媒体系统按其工作方式大致可分为两类：基于树状多播，和基于 Gossip 协议（基于 Gossip 协议也可称为基于网状多播）。

基于树状多播的 P2P 流媒体系统将网络中所有结点组织成一棵多播树，如图 1.1 左图所示，树的根结点是媒体发布源，数据分片总是从多播树的父结点向其子结点传播直到叶结点。基于多播树的方法可以最小化系统中多余的数据传播，并能保证每个数据分片能传播到系统中每个结点，但它有一个严重的缺陷：除叶结点以外任何一个系统结点的失效都将导致多播树分裂为两棵，而其中一棵在分裂后不能得到任何数据。因此，多播树极易分裂且维护多播树的开销巨大，造成树状多播方法不适合于高动态性的因特网环境。另一方面，树状多播方法的带宽利用率一般较低，原因有两方面：（1）多播树的叶结点只下载不上传，是纯粹的带宽消费者，对系统没有贡献；（2）多播树的父节点限制了其所在子树的最大输入带宽，因此多播树中带宽瓶颈结点到处存在。

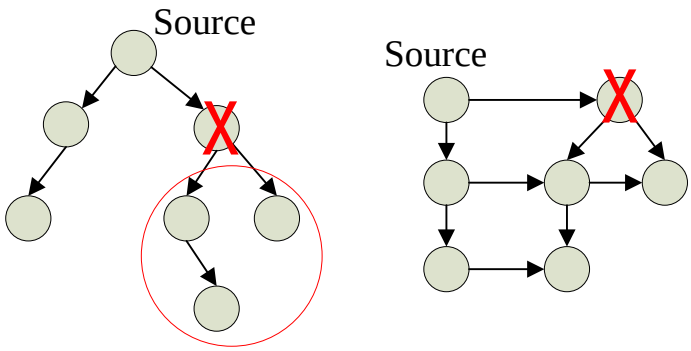


图 1.1 P2P 流媒体系统的两种工作方式：
基于树状多播（左图），基于 Gossip 协议（右图）

从本质上看，基于树状多播的方法实际上属于 P2P 流媒体系统沿袭传统流媒体技术的一种过渡。近年来，基于 Gossip 协议的 P2P 流媒体系统已成为 P2P 流媒体系统的主流。如图 1.1 右图所示，基本的 Gossip 协议算法让系统中每个结点维护一定数量的邻居信息，当一个结点发出或收到一个数据分片后将该数据分片随机地传播给部分邻居结点。邻居维护的灵活性与数据传播的随机性使得基于 Gossip 协议的 P2P 流媒体系统不会因结点失效而导致显著的性能下降，从而良好地适应了高动态性的因特网环境。在实际应用中，由于基本的 Gossip 协议所采用的随机“推”数据的方法会带来大量多余的数据传播，对于高带宽要求的流媒体传播极其不利，所以大多数基于 Gossip 协议的 P2P 流媒体系统，如 PPLive、PPStream、CoolStreaming、PeerStreaming、AnySee、PRIME 等，都采用更为明智的“拉”数据方法来传播数据：每个结点和其邻居周期性地交换数据可用性信息，通过分析数据可用性信息有选

择地从其邻居获取数据。这种“拉”数据方法称为改进的 Gossip 协议，也是本文所主要研究的 Gossip 协议。

相比树状多播方法，基于 Gossip 协议的网状多播方法的优势主要表现在高容错性与高带宽利用率两方面。因此，基于 Gossip 协议的 P2P 流媒体系统更适用于大规模、高动态的 Internet 网络环境，凭借本质的技术优势成为 Internet 流媒体领域的主流。表 1.1 总结了基于树状多播和基于 Gossip 协议的 P2P 流媒体系统的优缺点。

表1.1 基于树状多播和基于Gossip协议的P2P流媒体系统的优缺点

	基于树状多播的P2P流媒体系统	**基于Gossip协议的P2P流媒体系统**
拓扑结构	树状	网状，无固定形状
邻居数目	子结点数，或子结点数+1	不确定，通常有一上限
传播方式	父节点传播给其子结点	邻居间按需传播
容错性	单点失效，容错性低	灵活健壮，容错性高
多余数据	无	有
带宽利用	利用率低，瓶颈结点到处存在，将近一半的叶结点对系统无贡献	利用率高，不存在瓶颈结点，所有结点对系统都有贡献
适用环境	静态或低动态网络	高动态网络
系统规模	中、小规模	大规模
发展情况	逐渐被淘汰	领域主流

1.3 P2P流媒体系统的发展历程

（1）Narada端系统多播系统

P2P流媒体系统最早可追溯到2000年发表于ACM SIGMetrics会议的Narada端系统多播系统（End System Multicast）[CR00]。卡内基梅隆大学的研究者Y. Chu等人提出端系统多播方案的初衷是为克服IP多播的固有缺陷、为Internet多播提供另一条可能的出路，而事实上，这一实验性的设计思想后来被证明不仅是Internet多播可能的出路、而且是Internet多播主导的方案，直接驱动了工业界的大规模开发和学术界的密集性研究。

在Narada出现以前的10多年中，IP多播一直被认为是Internet多播的必然形式，原因在于IP层提供了最基本的数据路由功能（实际上相当于数据单播功能），并且大多数IP层路由器在制造时就考虑了IP多播功能的扩展或支持。尽管如此，IP多播在经历10多年的研究开发后仍然停滞不前，其主要原因有：（1）IP多播需要让路由器维护每个多播组的状态信息，这个需求与IP层最初的无状态设计架构是矛盾的，它给IP层带来了巨大的设计复杂性与扩展困难性，实际上是要求Internet工业界替换已铺设的硬件网络、重新构建IP层；（2）由于IP层提供的功能一直被界定为尽最大努力传输数据、而并不提供传输质量保证，因此IP多播也只能是尽最大努力传输数据的服务，Internet多播对传输质量的要求是相对较高的，如果要求IP多播提供诸如可靠性保证、拥塞控制、流控制、安全性保证等服务质量保证，从实现上看开发难度太大、成本太高。

Narada对“IP多播是Internet多播的必然形式”这一命题提出质疑，建议将Internet多播的层次提高，从IP层提高到应用层，从路由器提高到端系统，端系统指的是Internet用户主机（Hosts），端系统之间的边对应于连接两个端系统的一条IP路径。实际上，端系统正是P2P覆盖网中的一个普通结点，“端系统多播”这一说法后来也逐渐被“应用层多播”、“覆盖网多播”和“P2P流媒体系统”这样的名称所取代，这几个名称在本文中是等价的。

端系统多播相比于IP多播存在几个明显的优势：（1）不需要改变IP层的设计与路由器的

制造，受Internet工业界欢迎；（2）将传输质量保证提高到网络应用层来实现，难度大大降低，因为应用层的开发是灵活的、轻量级的，不会对Internet基础架构产生任何影响，并且应用层可以通过调用其下层（主要是传输层）提供的功能轻松地实现诸如可靠传递、拥塞控制、流控制、安全性保证等服务机制。另一方面，端系统多播相比IP多播也存在其明显的劣势：由于端系统之间的一条边对应于一条IP路径，所以端系统多播必然带来大量多余的数据包流量和明显高于IP多播的通信时延，从而导致端系统多播的通信开销要远远高于IP多播。绝大多数新技术相比于老技术都不可能处处占优，总有一个权衡（tradeoff）存在，在优劣共存的情况下，新技术是否能取代老技术，取决于新技术所带来的最明显的优势是否能远远超出其劣势。端系统多播，也就是后来的P2P流媒体系统，从2000年提出到本文的写作不到8年的时间里，不仅取代了IP多播，更成为Internet流媒体传输的主流，正是源于上面所述的优势远远超出其通信开销增大的劣势。

Narada系统的设计是比较朴素的，它以自组织、纯分布式的方式进行多播组的管理，主要是检测和更新结点、边的状态，消除覆盖网的分割。组成员信息周期性地在组内广播，假设N为组成员数目，每个周期多播组维护的通信开销是$O(N^2)$，这个开销是很高的，所以Narada系统的定位是中小规模的多播系统，通常在数十到数百个结点之间。Narada系统分为两层来构建，底层称为“状态层”，使用网状多播协议维护系统成员状态信息，上层称为“数据层”，基于状态层的信息构建多播树作为流媒体数据的真正传递路径，多播树的维护和优化也是通过状态层信息的更新来实现的。总体上看，Narada系统的设计方案并不适合Internet流媒体传输，而且包含大量诸如多播树这样的传统机制。Narada的主要意义在于端系统多播这一革新思想的提出。

（2）Overcast覆盖网多播系统

Overcast是2000年发表于OSDI会议上的一个先驱性的覆盖网多播系统 [JG00]，其名称Overcast可以解读为Overlay + Multicast，即覆盖网+多播。Overcast系统也分为“状态层”与“数据层”两层，但其状态层的构建方式与Narada系统有着明显的不同：Overcast状态层使用树状多播协议维护系统成员信息。在状态层之上，数据层构建多播树传递流媒体数据。在状态层和数据层都采用树状多播协议，带来了一定程度上的功能重复，而且树状多播的容错性较低，因此Overcast的影响力有限，很少有后继工作。

（3）ALMI应用层多播系统

ALMI系统，英文全称为Application Layer Multicast Infrastructure，是2001发表于USITS会议上的一个应用层多播系统 [PS01]。ALMI的系统状态维护是集中式的，通过称为Session Controller的中央控制单元来监控和更新整个系统中每个结点的状态，因此其扩展性较差，只支持数十个结点的中小规模系统。ALMI系统通过Session Controller来统一构建多播树传递流媒体数据，与Narada及Overcast系统中的数据层多播树不同的是，ALMI多播树的每条边都是双向的，不过每条边上的两个结点会协商一种临时的父子关系，这种父子关系的存在是为了消除出现循环路径，并不决定数据流向。ALMI以集中方式维护整个系统状态信息的做法缺乏可扩展性，所以影响力不及最早的Narada与Overcast。

（4）Bayeux系统

发表在NOSSDAVA’01会议上的Bayeux系统最早将P2P覆盖网技术应用到Internet数据多播中 [ZZ01]。与前面讲过的Narada、Overcast系统类似，Bayeux也维护了状态层和数据层的双层结构，但其状态层基于著名的P2P覆盖网Tapestry [ZK01]，通过Tapestry提供的高效的路由和定位操作维护结点状态信息，因此能支持大规模的流媒体系统，并且其容错性、系统维

护开销相比先前的几个系统要优越的多。Bayeux的数据层由多棵多播树组成，共享相同的数据流的系统结点形成一个组（group），与组标识（groupID）最接近的组内结点（一般是媒体发布源）成为多播树的根结点，想加入该组的新结点首先通过Tapestry路由定位到根节点，根节点再通过Tapestry路由定位到新结点并随之建立从根结点到新结点的多播树路径，由于Tapestry的路由是非对称的，也就是说从结点*A*到结点*B*的路由路径和从*B*到*A*的路由路径可能不一样，所以上述结点加入和建立多播树路径的过程被称为“非对称加入和树路由”，这是Bayeux系统的一大特色。

为了增强多播树的容错性，尤其是防止多播树分裂为多棵，Bayeux系统采用了复制根结点的方法，将一棵多播树的根节点复制成多个子根结点，每个子根结点负责多播树的一部分成员信息维护，形成一棵子树。同时，Bayeux系统根据实际测量到的网络时延将一些ID相近的结点集簇到一起，提高了覆盖网与底层IP网络的拓扑匹配性。

（5）Scribe系统

Scribe系统 [CD02] 与Bayeux系统有诸多相似之处。首先，两者的状态层都构建在P2P覆盖网之上，Bayeux构建在Tapestry之上，Scribe构建在Pastry [RD01] 之上，而Tapestry和Pastry实际上都是由一个更早的P2P覆盖网模型Plaxton Mesh [PR97] 扩展来的。其次，两者都构建多播树来传递数据流，都将共享相同数据流的结点组成一棵多播树。再次，Bayeux也采用了复制根结点的方法，将一棵多播树的根节点复制到多个临近结点以防止根结点失效。

Scribe与Bayeux的不同之处主要表现在构建多播树的方向，Bayeux是从根到新结点来构建多播树路径，而Scribe则是从新结点到根来构建多播树路径。Scribe的设计者通过模拟实验表明他们的多播树路径构建方案能够带来更低的路由时延。

（6）NICE系统

发表在SIGCOMM’02会议上的NICE系统（意为：NICE is Internet Cooperative Environment）[BB02] 通过将覆盖网结点层次化地集簇来大幅度降低系统维护开销，同时降低多播树的数据流传播时延，图1.2是层次化集簇的一个3层示例图。NICE系统并不显式地建立多播树，它所构造的层次化集簇隐式地包含了数据传递路径。具体说来，NICE在保持每个结点常数度的前提下将最坏情况下的结点状态维护开销降低到*O(logN)*，*N*为系统结点总数，通过有拓扑意识的集中式算法将系统延展性（Stretch，即在一次路由过程中的IP层跳数同覆盖网跳数的比值）降低到接近*O(logN)*。NICE系统的主要价值在理论上，其后继工作并不多。

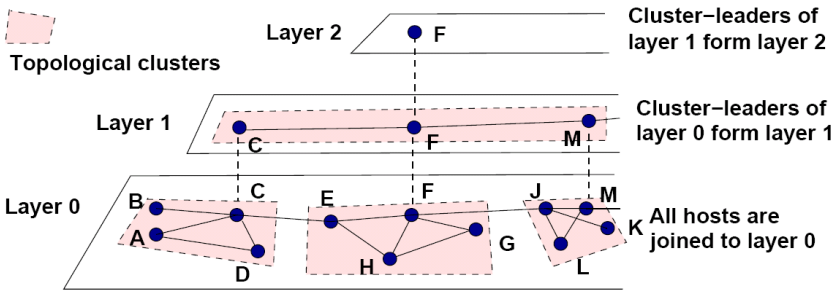


图1.2 NICE系统的结点层次化集簇示例图

（7）ZIGZAG系统

ZIGZAG系统 [TH03] 相当于NICE系统的改进版。假设N为系统结点总数，k是对结点层次化集簇时的一个重要常数，那么ZIGZAG系统能保证其中多播树的高度始终为$O(\log_k N)$，结点度始终为$O(k^2)$，最坏情况下的结点状态维护开销为$O(\log_k N)$，而平均情况下的结点状态维护开销为$O(k)$。相比NICE系统，ZIGZAG系统最大的优点表现在：当需要做错误修复时，ZIGZAG的修复操作只发生在局部区域的常数个结点上，不会给数据源结点带来任何负担。而上述优势的取得，关键在于ZIGZAG在组织多播树时采用了一条与NICE不同的规则：任何一个集簇的父结点只能选择外部集簇的头结点，见图1.3。

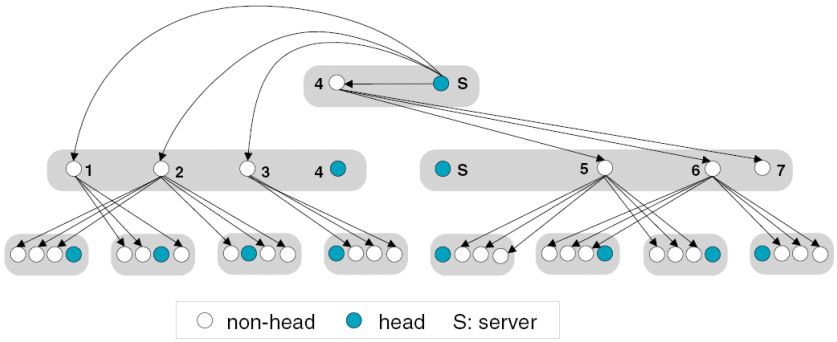


图1.3 ZIGZAG系统的多播树组织示例（k=4）

（8）SplitStream系统

上节总结过多播树方法的优点：（1）可以最小化系统中多余的数据传播；（2）能保证每个数据分片传播到系统中每个结点。同时多播树方法至少有3个方面的缺点：（1）除叶结点以外任何一个系统结点的失效都将导致多播树分裂为两棵，而其中一棵在分裂后不能得到任何数据；（2）多播树的叶结点只下载不上传，是纯粹的带宽消费者，对系统没有贡献；（3）多播树的父节点限制了其所在子树的最大输入带宽，因此多播树中带宽瓶颈结点到处存在。针对上述特点，从克服缺点、保留优点的角度出发，M. Castro等人在SOSP’03会议上发表了SplitStream系统的设计方案 [CD03]。

SplitStream将系统结点组织成k棵多播树形成一个多播森林，每个结点在一棵多播树中作为内部结点，在剩下的若干棵树（≤k-1）中作为叶结点，克服了缺点（2）和（3）。SplitStream将数据流切成k个更细的数据子流分配到k棵多播树中以平衡负载，并且任何一个结点的失效最多导致一棵多播树分裂、一个数据子流暂停，不影响其它多播树正常工作，克服了缺点（1）。如果将SplitStream系统与网络冗余编码结合起来，能够保持应对结点失效的高容错性，当然，系统的设计和实现必然更为复杂。

多播森林的方法虽然保持了多播树方法的优点、克服了多播树方法的缺点，却带来了维护多棵多播树的过量开销和系统实现上更高的难度，所以一直没有真正被应用。

（9）CoolStreaming系统

X. Zhang 等人发表在 INFOCOM’05 会议上的 CoolStreaming 系统 [ZL05b] 在 P2P 流媒体系统的发展历程上具有重要的现实意义。早在 2004 年 5 月欧洲杯期间，CoolStreaming 原型系统就已在 Planet-Lab 平台上试用获得成功，在 3 台服务器上并发用户迅速积累到 25 万人，奠定了 P2P 直播技术进入商业运作阶段的基础。在 CoolStreaming 之前，基于 Gossip 协议的 P2P 流媒体系统理论和设计已经相当完善，但始终缺乏现实的说服力，而 CoolStreaming 则通过其实践证实了这种说服力。实际上，CoolStreaming 的发明人在发表其论文后不久就

进入工业界创办了光芒国际传媒网络技术有限公司（http://www.roxbeam.com/），其经营产品主要包括视频 CDN、P2P 流媒体系统和光芒国际流媒体运营平台。

基于 Gossip 协议的 P2P 流媒体系统历史很短。就我们目前所知，Kermarrec 等人最先给基于 Gossip 协议的可靠多播算法提供了理论支持 [KM03] ：他们证明了在一个包含 N 个结点的网络中，如果平均每个结点随机给($logN+K$)个其它结点发送一条消息 M，那么最终网络中每个结点都收到消息 M 的概率将趋近于 $e^{-e^{-K}}$ 。这一理论分析结果提供了对实用化的基于 Gossip 协议的 P2P 网络多播技术 [GK03] 的指导。然而， $e^{-e^{-K}}$ 这一接近 1.0 的消息覆盖率仅仅是理想化的理论结果，并没有考虑到带宽、时延等限制性因素。

CoolStreaming 使用上面提到的基于 Gossip 协议的 P2P 网络多播技术 [GK03] 构建了一个灵活、实用的 P2P 流媒体系统，其设计者也称之为 DONet（Data-driven Overlay Network，数据驱动的覆盖网络）。CoolStreaming 系统的核心操作非常简单：每个结点和其邻居周期性地交换数据可用性信息，通过分析数据可用性信息有选择地从其邻居获取数据。为了提高 P2P 流媒体系统的关键属性之一：播放连续度，CoolStreaming 设计了“稀缺优先”的数据调度算法，每个结点优先获取对它来说稀缺的数据分片，稀缺意味着能提供该数据分片的邻居数目很少。总体上看，CoolStreaming 最大的优点在于其设计的简单、灵活、有效、易于实现，可以说这正是 CoolStreaming 模式后来能够在工业界广泛推广的关键性原因。

（10）GridMedia系统

M. Zhang 等人观察到 Gossip 协议采用“拉”数据的方法虽然比“推”数据要节省带宽，却带来了更大的数据传播时延，因此他们设计了一个“推拉结合”的系统 GridMedia [ZL05a]。他们将 P2P 流媒体系统中的数据分片分成两类：一类数据分片只在被请求获取时才传播，称为“拉数据”；另一类数据分片一旦结点收到就立即传播给邻居，称为“推数据”。通过推拉结合的方法，GridMedia 能获得相比一般的基于 Gossip 协议的 P2P 流媒体系统更低的数据传播时延。GridMedia 实际系统层被用来在全球 Internet 上以 300kbps 的数据率实时转播 2005 年 CCTV-1 的春节联欢晚会，共有约 50 万用户参与了这一过程。

（11）P2P 流媒体系统现状

目前在 Internet 上被广泛应用的 P2P 流媒体系统有 Skype、UUCall、PPLive、PPStream、QQ 直播、新浪 UUSee、沸点网络电视、TvAnts 等，其中无论在网络音频还是网络视频领域均存在数十家公司激烈竞争，其中占据最大份额的几家是 Skype、PPLive 和 PPStream。

由于绝大多数商业系统的架构、技术都是对外保密的，研究者只能通过测量或抓包分析来大致推断其系统结构。根据 PPLive 公司首席架构师 Y. Huang 在 2007 年 SIGCOMM 的 IPTV workshop 上所作的关于 PPLive 系统的报告，PPLive 目前拥有约 7500 万安装用户、约 2000 万活跃用户、超过 600 个实时转播的电视频道。从图 1.4 可以看出，PPLive 系统分两层来组织：超结点层（SN）和用户层（C）。超结点层由性能稳定的超结点或虚拟超结点组成，超结点之间参照 Chord 覆盖网模型来组织，目前正试图以更灵活的 Kademlia 覆盖网模型 [MM02] 来替换 Chord。对 PPLive 实时流媒体系统来说，每个超结点负责若干个频道的数据发布；而对于 PPLive 视频点播系统来说，每个超结点负责一定范围的区域。每个超结点带领其下属的用户形成一个多播组，不同的多播组可以根据组内结点的特性采用不同的多播协议，如多播树或 Gossip 多播。

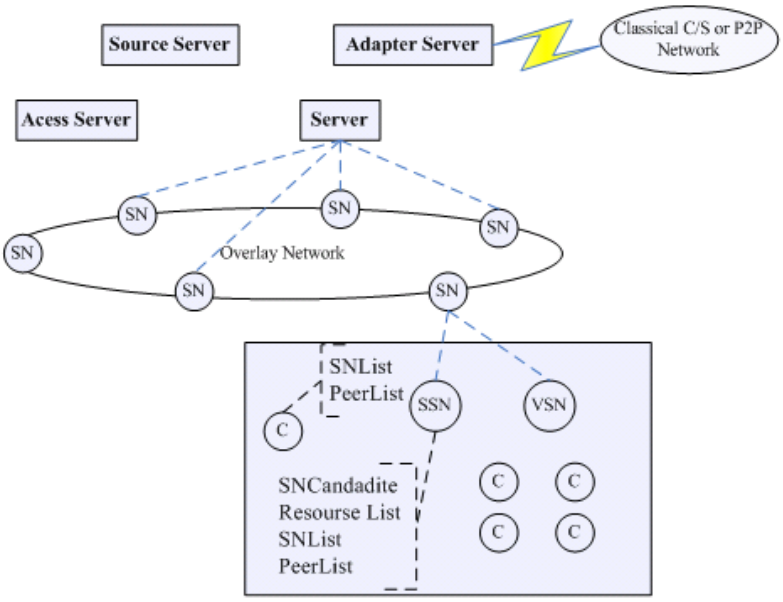


图 1.4 PPLive 系统架构图

1.4 P2P流媒体系统的关键属性

P2P流媒体系统值得研究的属性有很多，从用户体验的角度看有播放连续度、播放数据率、启动时延、源切换时延、带宽利用率、系统容错性等等，从网络设计者的角度看还有定位效率、端到端时延、数据吞吐量、拓扑一致性、负载均衡、系统维护开销、控制开销、可扩展性等等。在表1.2中，我们列举了P2P流媒体系统的各项属性及其描述。

表1.2 P2P流媒体系统的各项属性

分类	属性	描述
用户体验的角度	播放连续度	从流媒体开始播放后，在时间段T中，能连续播放流媒体的时间为T'，那么播放连续度就定义为$\frac{T'}{T}$。此属性越高则观看越连续。
	播放数据率	连续播放时，每秒钟播放流媒体的数据量，通常在200kbps—1Mbps左右。此属性越高则画质越好，但播放数据率受制于物理带宽。
	启动时延	从用户发出流媒体播放请求后，到流媒体开始播放的时间间隔。此属性越低则启动播放越快。
	源切换时延	在支持多个流媒体发布源的系统中，当发生旧发布源停止、新发布源启动的事件时，用户所感受到的切换间隔时间。此属性越低则源切换越快。
	带宽利用率	衡量系统结点的带宽是否得到了充分的利用。
	系统容错性	“容错性”是一个含义很广的概念，粗略地讲它表示系统对其成员错误行为的健壮程度、受影响程度。
网络设计者的角度	定位效率	定位任意一个结点或者数据对象所需要走过的覆盖网路由跳数，或理解为任意一条消息从源结点出发要到达目的结点所走过的覆盖网跳数。

	端到端时延	消息从一个端用户发送到另一个端用户所需要的时间。
	数据吞吐量	在较长的一段时间中，系统或某个结点的输入输出数据量之和。
	拓扑一致性	P2P覆盖网与物理网之间拓扑结构匹配的程度。
	负载均衡	数据对象在网络结点中分布得是否均匀，以及每个结点为其它结点消息路由所承担的负载是否平衡。对以P2P流媒体系统来说，主要指后者。
	系统维护开销	随着新结点不断加入、旧结点不断离开或失效，为了维护系统体系结构和保持结点状态更新所需要的消息通信开销。
	控制开销	对于Gossip协议的P2P流媒体系统，每个调度周期中，结点和其邻居交换数据可用性信息，控制开销就定义为交换数据可用性信息的通信开销占实际流媒体数据传输的通信开销的比例。
	可扩展性	当系统规模增大、结点增多时，系统是否还能正常工作，其它各项属性发生的变化是否仍然合理、可以接受？可扩展性是衡量一个分布式系统性能的重要属性，在系统规模剧增时尤其明显。

实际上，表 1.2 中的各项属性是有所交叉的，比如播放连续度就受播放数据率、带宽利用率、定位效率、端到端时延等多项属性的混合影响。经过对大量文献的分析和作者自身的研究经验，我们认为 P2P 流媒体系统的关键属性如下：

播放连续度、启动时延、源切换时延、系统容错性、可扩展性。

它们的共同特点是：直接影响用户体验和系统宏观评价，人为因素的影响较小，仍然存在较大的可优化空间。

1.5 本文的工作

用一句话来说，本文对P2P流媒体系统的3个关键属性（1）播放连续度、（2）源切换时延、（3）系统容错性，提出、设计并实验测试了对应的优化方案，如图1.5所示。

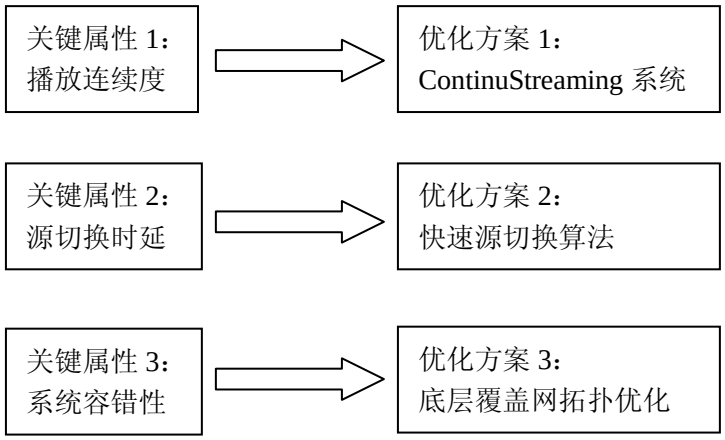


图1.5 本文优化的3大关键属性及其优化方案示意图

本文第2章设计了一个具有高播放连续度的P2P流媒体系统ContinuStreaming，采用基于分布式散列表（DHT）的数据预取方法来弥补Gossip协议传播数据的缺陷，从而保证流媒体系统能保持高播放连续度。系统中每个结点不断预测哪些数据分片很可能被数据调度算法遗漏，如果有认为遗漏的数据分片，就启动数据预取算法通过DHT快速查找并获得它们，从而保证连续播放。我们对ContinuStreaming系统的播放连续度进行了理论分析并将理论分析的结果与模拟实验的结果进行了比较。在多幅真实P2P网络拓扑上所做的大量模拟实验结果表明：相比于当前具有代表性的基于Gossip协议的P2P流媒体系统CoolStreaming而言，本文设计的ContinuStreaming系统能以低于4%的额外开销带来接近1.0的高播放连续度。

本文第3章通过对本质特性与关键参数的分析给P2P流媒体系统中数据发布源切换过程建立了数学模型，从而将源切换问题形式化为一个数学优化问题，然后推导出此数学优化问题的最优解。鉴于实际系统情况的复杂性与网络环境的动态性，我们提出了一个称为“快速源切换算法”的实用贪心算法，它通过交错旧源与新源的数据传递来趋近理论上的最优解。我们在多个真实测量的P2P覆盖网拓扑结构上做了大量模拟实验来证实快速源切换算法的有效性，模拟实验的结果显示：我们提出的快速源切换算法相比传统源切换算法能节省20%-30%的切换时间，同时不会带来额外的通信开销，并且切换时间的减少比例随着系统规模的增大而趋于增加。

本文第4章首先提出分点这一概念来描述P2P流媒体系统的底层覆盖网的拓扑关键点，然后基于分点的概念，设计了一套简单、有效、分布式的分点检测和消除方法来优化覆盖网拓扑结构。模拟实验的结果表明：我们的方法能有效地检测并消除分点，使系统对覆盖网分割的抵抗力得到本质的增强；分点消除后，系统查询成功率有所上升，在高动态性网络环境下系统的容错性得到明显提高。

上述各章都以研究该问题的背景与动机作为开始，其次是对该问题的国际研究现状进行综述，再次是我们在研究这个问题上所做的工作，最后以小结和进一步的工作作为结尾。此外，本文还在第5章结束语中简单总结了本文的主要工作，并提出了P2P流媒体系统领域一些值得研究的问题。

最后，本文在附录部分对本人攻读硕士学位期间参加科研情况、撰写论文情况进行了总结。

第2章 具有高播放连续度的P2P流媒体系统的设计

2.1 背景与动机

上文已经讲过，基于 Gossip 协议的 P2P 流媒体系统已成为 P2P 流媒体系统的主流。虽然改进的 Gossip 协议能更为明智地获取数据，但它并不能够克服 Gossip 协议固有的缺点：数据传播的随机性与不确定性，因此，很难保证从媒体数据源发布的每个数据分片能按时传播到每个结点，导致当前基于 Gossip 协议的 P2P 流媒体系统普遍存在着播放连续度较低的问题。播放连续度一般被认为是一个流媒体系统最重要的属性，不连续的播放效果往往要比低数据率造成的低画质效果要更损害观看体验。

为了提高 P2P 流媒体系统的播放连续度，目前提出的方法大多数着眼于设计一个良好的数据调度算法，以使得每个结点从其邻居结点中获得尽可能多的“好”数据分片，这里数据分片的“好”可以由其紧迫性、稀缺性等属性来衡量。设计一个良好的数据调度算法对提高播放连续度是必要的，但并不足够，比如当某个结点 A 迫切需要获得某个数据分片 d 时，可能出现如下 3 种情况：1）A 的邻居结点都没收到 d（见图 2.1 左图）；2）A 的某个邻居 B 收到过 d 但 d 已经被 B 播放过并且从 B 的数据缓存中移除（见图 2.1 中图）；3）A 的某个邻居 C 收到过 d 且 d 在 C 的数据缓存中，但 C 没有足够的可用带宽发送 d（见图 2.1 右图）。对于上述情况中的任何一种，数据调度算法都不起作用。所以说数据调度算法不能从本质上克服 Gossip 协议随机性的缺点，也就不能从本质上保证基于 Gossip 协议的 P2P 流媒体系统保持高播放连续度。

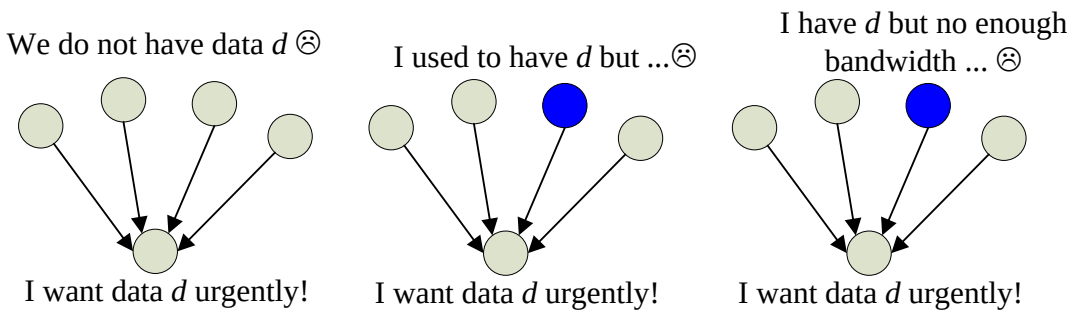


图 2.1 数据调度算法不起作用的 3 种情况示例

2.2 国际研究现状

基于 Gossip 协议的 P2P 流媒体系统历史很短。就我们目前所知，Kermarrec 等人最先给基于 Gossip 协议的可靠多播算法提供了理论支持 [KM03]：他们证明了在一个包含 N 个结点的网络中，如果平均每个结点随机给$(logN+K)$个其它结点发送一条消息 M，那么最终网络中每个结点都收到消息 M 的概率将趋近于 $e^{-e^{-K}}$。这一理论分析结果提供了对实用化的基于 Gossip 协议的 P2P 网络多播技术 [GK03] 的指导。然而，$e^{-e^{-K}}$ 这一接近 1.0 的消息覆盖率仅仅是理想化的理论结果，并没有考虑到带宽、时延等限制性因素。

CoolStreaming [ZL05b] 使用上面提到的基于 Gossip 协议的 P2P 网络多播技术 [GK03]

构建了一个灵活、实用的 P2P 流媒体系统。为了提高播放连续度，CoolStreaming 设计了“稀缺优先”的数据调度算法，每个结点优先获取对它来说稀缺的数据分片，稀缺意味着能提供该数据分片的邻居数目很少。我们在设计 ContinuStreaming 系统的数据调度算法时，综合考虑了数据的稀缺性与紧迫性（一个将要被播放的数据分片具有高紧迫性），这比 CoolStreaming 仅考虑数据稀缺性更为合理。

Xu 等人考虑了 P2P 流媒体系统中的数据率分配问题 [XH02]。P2P 流媒体系统中每一个结点都连有若干带宽不等的邻居，Xu 等人设计了一个名为 OTS_{P2P} 的数据率分配算法，能够保证当前结点具有最小的数据缓冲时延，从而获得高播放连续度。然而，OTS_{P2P} 算法成立的前提非常苛刻，它要求系统结点的带宽满足严格的倍数关系，实际的 P2P 流媒体系统是无法满足该前提的。

Zhang 等人观察到 Gossip 协议采用“拉”数据的方法虽然比“推”数据要节省带宽，却带来了更大的数据传播时延，因此他们设计了一个“推拉结合”的系统 GridMedia [ZL05a]。他们将 P2P 流媒体系统中的数据分片分成两类：一类数据分片只在被请求获取时才传播，称为“拉数据”；另一类数据分片一旦结点收到就立即传播给邻居，称为“推数据”。GridMedia 系统主要的设计目标是减少数据传播时延，它间接地提高了播放连续度。然而，推数据的方法必然带来相当大的通信开销，而且也不能从本质上保证高播放连续度。类似地，P2P 流媒体系统 AnySee [LJ06] 采用覆盖网间的优化方法（Inter-overlay Optimization）来减少数据源到接收端的时延，通过时延的减少来间接提高播放连续度。

自适应的层次化 P2P 流媒体系统 PALS [RO03] 采用接收者驱动的方法实现层次化编码流媒体的质量可调节播放。PALS 使用可调节播放质量的缓存机制来最大化每个结点的通信吞吐量，并使用滑动窗口机制来阻止发送者发送已经过期的数据分片。在播放连续度方面，PALS 是采用调节流媒体编码层次（编码层次越多，数据率越高，流媒体画质就越好）来获得较高的播放连续度。我们在 ContinuStreaming 系统中设计用来提高播放连续度的方法，其适用面比 PALS 要广很多，因为 PALS 仅限于层次化编码的流媒体。

2.3 ContinuStreaming系统

本文设计了一个具有高播放连续度的 P2P 流媒体系统 ContinuStreaming，采用基于分布式散列表（DHT）的数据预取方法来弥补 Gossip 协议传播数据的缺陷，从而保证流媒体系统能保持高播放连续度。DHT 的作用在于提供大规模网络中高效、分布式的数据放置与定位 [SM01] [RD01]。基于 DHT 的定位功能，从媒体源发布出来的每一个数据分片都会被分布式地存储在 k 个网络结点中，k 是一个系统常量。系统中每个结点不断预测哪些数据分片很可能被数据调度算法遗漏（即通过数据调度算法很可能无法按时获得），如果有认为遗漏的数据分片，就启动数据预取算法通过 DHT 快速查找并获得它们，从而保证连续播放。

我们对 ContinuStreaming 系统的播放连续度进行了理论分析并将理论分析的结果与模拟实验的结果进行了比较。在多幅真实 P2P 网络拓扑上所做的大量模拟实验结果表明：相比于当前具有代表性的基于 Gossip 协议的 P2P 流媒体系统 CoolStreaming 而言，本文设计的 ContinuStreaming 系统能以低于 4%的额外开销带来接近 1.0 的高播放连续度。

总体来说，本文对领域的贡献可以总结为如下 3 点：

（1）据我们目前所知，本文是 P2P 流媒体领域第一次尝试采用基于 DHT 的数据预取方法来弥补 Gossip 协议数据传播随机性的缺陷，从而设计出一个具有高播放连续度的 P2P 流媒体系统。更重要的是，该系统带来的额外开销非常的低。

（2）设计了一套称为“紧迫界线”的机制让每个结点动态、自适应地预测可能被数据调度算法遗漏的数据分片。这套机制可以有效地避免不必要的数据预取操作。

（3）通过理论分析和基于真实 P2P 网络拓扑的模拟实验证实了 ContinuStreaming 系统的有效性。

2.3.1 系统概览

ContinuStreaming 系统主要由 3 部分组成：（1）P2P 覆盖网管理；（2）数据调度算法；（3）数据预取算法。这一节概要讲述这 3 个部分，每一部分的细节描述将在下一节中。

基于 Gossip 协议的 P2P 流媒体系统通常采用无结构 P2P 网络作为其底层结构，因为无结构 P2P 网络具有灵活松散、维护开销低的特点。如前文所提到的，由于运行在无结构 P2P 网络上的 Gossip 协议固有的随机性，不能保证高播放连续度，我们设计了一个轻量级的混合式 P2P 覆盖网，组合了无结构 P2P 覆盖网和结构化 P2P 覆盖网，结构化 P2P 覆盖网也就是用来实现高效数据定位的 DHT。之所以称此混合式 P2P 网络是轻量级的，因为它所支持的 DHT 是松散组织的，并且其中结点状态的更新主要依靠监听经过的路由消息，所以维护开销很低。

数据调度算法工作在无结构 P2P 覆盖网之上，它从建立连接的网络邻居中周期性地获取数据可用性信息，从而规划数据分片的获取过程。我们为 ContinuStreaming 系统设计的数据调度算法综合考虑了数据分片的稀缺性与紧迫性，根据一个分片的稀缺性与紧迫性计算出获取它的优先权，优先权越高的数据分片越早获得，从而有效减少不能按期播放的数据分片数目。

数据预取算法工作在结构化 P2P 覆盖网即 DHT 之上。依靠 DHT 所提供的高效定位功能，从媒体源发布出来的每一个数据分片都会被分布式地存储在 k 个网络结点中，k 是一个系统常量，一般是较小的整数。系统中每个结点不断预测哪些数据分片很可能被数据调度算法遗漏，如果有认为遗漏的数据分片，就启动数据预取算法快速查找并获得它们，从而保证连续播放。

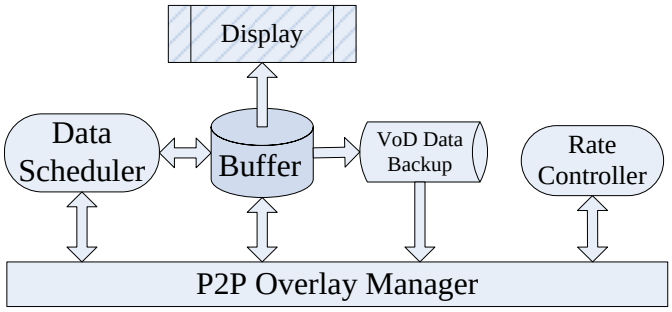


图 2.2 ContinuStreaming 系统结点的软件架构

图 2.2 描画了 ContinuStreaming 系统结点的软件架构，主要包含了几个模块：（1）P2P 覆盖网管理器（P2P Overlay Manager）是结点同覆盖网间的通信接口，负责维护和更新结点状态；（2）数据调度器（Data Scheduler）从建立连接的网络邻居中周期性地获取数据可用性信息，从而安排到哪里去取哪些数据分片；（3）按需数据备份（VoD Data Backup）存储当前结点负责备份的数据分片，只要当前结点正常工作其它结点就能从当前结点的按需数据备份中获取它负责存储的数据分片；（4）流量控制器（Rate Controller）监控和估测来自每一个邻居结点的数据流量。

2.3.2 底层覆盖网

ContinuStreaming 系统中每个结点维护一张称为“Peer Table”的表，如图 2.3 所示，它由 3 个部分组成：

（1）连接邻居（Connected Neighbors）包含无结构 P2P 覆盖网上 *M* 个邻居。当前结点与这 *M* 个邻居建立 TCP 连接，周期性的数据交换仅发生在当前结点和其连接邻居之间。如果当前结点发现某个邻居已失效或者给自己提供的数据很少，就从监听结点中选取时延最小的结点来替换这个邻居。

（2）DHT 结点（DHT Peers）包含 *logN* 个按层（Level）排列的 DHT 结点。*N* 是覆盖网最多能容纳的结点个数。对于某个结点 *n* 来说（*n* 是该结点的 ID），它的第 *i* 层 DHT 结点唯一需要满足的条件是必须落在区间 $[n+2^{i-1}, n+2^{i})$ 中，所有结点的 ID 均是模 *N* 后的结果。可见结点 *n* 在选择它的 DHT 结点时拥有很高的灵活性，所以这里设计的 DHT 是松散组织的。所有的 DHT 结点周期性地用监听结点来更新。

（3）监听结点（Overheard Nodes）包含 *H* 个最近监听到的网络结点。根据我们的模拟实验经验，*H*=20 通常就足够了。每个结点不断地监听经过自身的路由消息，用路由消息中的结点信息更新监听结点的信息。

由上可见，连接邻居和 DHT 结点都是根据监听结点来更新的，而监听结点又是通过本地监听来更新，本地监听操作不需要额外的通信开销，因此我们所设计的混合式 P2P 覆盖网维护开销非常低。

Connected Neighbors				
No	**PeerID**	**IP address**	**Latency**	**Recent supply rate**
1	369	210.29.131.34	5ms	50kbps
…	…	…	…	…
M	672	165.93.100.77	15ms	25kbps
DHT Peers				
Level	**PeerID**	**IP address**	**Latency**	
1	442	102.119.32.30	25ms	
…	…	…	…	
logN	891	219.45.128.49	30ms	
Overheard Nodes				
No	**PeerID**	**IP address**	**Latency**	
1	453	107.124.33.58	15ms	
…	…	…	…	
H	120	61.173.76.44	100ms	

图 2.3 Peer Table 结构

如果结点 *A* 想要加入覆盖网，*A* 首先需要联系“会合服务器”（Rendezvous Server），会合服务器的 IP 地址是公开的，它维护系统结点的列表。会合服务器给 *A* 分配一个独一无二的 ID，并给 *A* 发送一张很短的列表，其中包含着与 *A* 的 ID 最相近的若干现存结点的信息。假定 *A* 收到的列表是{*B*,*C*,*D*,*E*}，*A* 首先使用 UDP 数据报给 *B*、*C*、*D*、*E* 发送 *PING* 消息来探测哪个结点离 *A* 最近且在线，结点间的距离（也就是时延）通过 $RTT/2$ 来近似估计，*RTT* 是探测消息的往返时间（Round Trip Time）。如果 *B*、*C*、*D* 都在线而 *E* 已失效，且 *B* 是其中离 *A* 最近的结点，那么 *A* 就获取 *B* 的 Peer Table 作为 *A* 的 Peer Table 的基础，同时通知 *B*、*C*、*D* 它的加入，并且通知会合服务器 *E* 已失效。

我们设计的 P2P 覆盖网支持 ContinuStreaming 系统所需要的单播与多播功能。单播功能，也就是定位一个结点或者一个数据分片，通过 DHT 路由算法来实现。DHT 路由算法是一个

简单的贪心算法：对于路由过程中经过的每一个结点，都选择此结点 DHT 结点中离目标顺时针最近的结点作为下一跳，直到再也找不到更近的结点为止。我们证明了一次 DHT 定位的路由跳数上限是 $\frac{\log N}{\log(4/3)} \approx 2.41 \times \log N$，如下所示。

定理 1：对于我们所设计的 DHT，一次 DHT 定位的路由跳数上限是 $\frac{\log N}{\log(4/3)} \approx 2.41 \times \log N$。

证明：假设结点 *S* 想要定位结点 *T*，那么路由消息的传递顺序可能是 $S \to H_1 \to H_2 \to \cdots$。从 *S* 到 *T* 的顺时针距离 $dist(S,T) \in [2^{d-1}, s^d]$，*d<logN*。*S* 首先选择它的顺时针最近邻居 H_1 作为下一跳，H_1 应该是 *S* 的第 *d* 层或(*d-1*)层邻居，所以有两种情形：

（1）H_1 是 *S* 的第 *d* 层邻居，那么 $dist(H_1,T) \le 2^{d-1}, dist(S,H_1) \ge 2^{d-1}$；

（2）H_1 是 *S* 的第(*d-1*)层邻居，那么 $dist(H_1,T) < 2^{d-1} + 2^{d-2} = 3 \times 2^{d-2}, dist(S,H_1) \ge 2^{d-2}$。

为了推导出路由跳数的上限，我们一直采取最坏的情形即情形 2 来进行每一跳计算，通过下面的步骤可以推出 $dist(H_1,T) < \frac{3}{4} \times dist(S,T)$：因为 $dist(S,T) = dist(S,H_1) + dist(H_1,T)$，所以 $\frac{3}{4} \times dist(S,T) - dist(H_1,T) = \frac{3}{4} \times dist(S,H_1) - \frac{1}{4} \times dist(H_1,T) = \frac{3 \times dist(S,H_1) - dist(H_1,T)}{4}$；又因为 $dist(H_1,T) < 3 \times 2^{d-2}, dist(S,H_1) \ge 2^{d-2}$，所以 $\frac{3}{4} \times dist(S,T) - dist(H_1,T) > 0$，也就是 $dist(H_1,T) < \frac{3}{4} \times dist(S,T)$。

类似地，$dist(H_2,T) < \frac{3}{4} \times dist(H_1,T) < (\frac{3}{4})^2 \times dist(S,T)$，$dist(H_i,T) < (\frac{3}{4})^i \times dist(S,T) < (\frac{3}{4})^i \times N$。让 $dist(H_i,T) = 1$，也就是说 H_i 一定能够找到目标 *T*，那么 $(\frac{3}{4})^i \times N > 1$，$i > \frac{\log N}{\log(4/3)}$，这正是路由跳数的上限。■

此外，我们还对具有不同的 *N* 和 *n* 的 DHT 覆盖网进行了模拟实验，实验结果在图 3 中，*N* 是覆盖网最多可容纳结点数目，*n* 是覆盖网中现有结点数目。从图 2.4 可以看出，DHT 定位所需的平均路由跳数接近 $\frac{\log N}{2}$，查询成功率即使在覆盖网结点稀疏的情况下也接近 1.0，结点稀疏意味着 *n* 远小于 *N*。

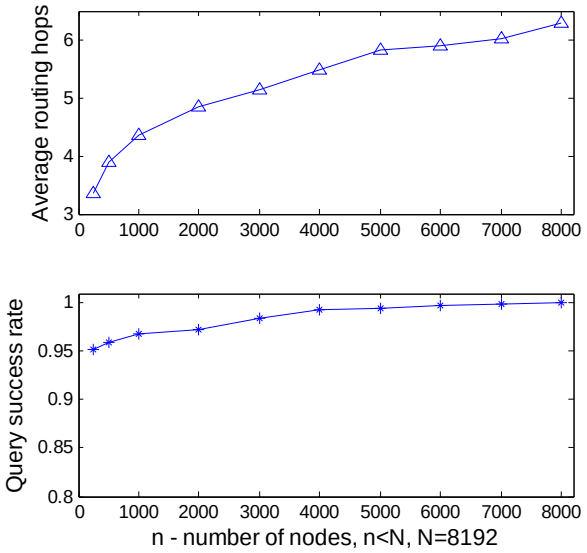


图 2.4 DHT 网络的性能

多播功能，也就是结点与其连接邻居间的数据交换，通过数据调度算法来实现。文献[ZL05b] 中的理论分析表明：在基于 Gossip 协议的多播系统中，距离发布源 d 跳的消息覆盖率近似为 $1-e^{-\frac{M\times(M-1)^d-2}{(M-2)\times n}}$，其中 M 是连接邻居数目，n 是覆盖网中现有结点数目。然而受诸多因素所限，实际的消息覆盖率要远远低于这一理论值，这也正是本文的动机和价值所在。

2.3.3 数据调度算法

每个结点何其连接邻居周期性地交换各自缓存中的数据信息，交换的周期称为“调度周期”，符号τ。每个调度周期中，结点的数据调度器首先查明连接邻居的缓存中有哪些数据分片可用，“可用”指的是这些数据分片是当前结点还没得到的。数据调度过程所涉及的各项参数列举在表2.1中。

表2.1 数据调度算法的相关参数

参数	说明
τ	数据调度周期。
id_i	数据分片 D_i 的 id。
n_i	能够提供 D_i 的邻居数目。
I	当前结点的接收总带宽。
R_{i_j}	从第 j 个提供者那里获取 D_i 的速率。
R_i	数据分片 D_i 的最大获取速率。
id_{play}	当前正在播放的数据分片的 id。
p	每秒播放的数据分片数目。
t_i	数据分片 D_i 的过期时间。
B	缓存大小，即缓存中能放多少分片。
p_{i_j}	数据分片 D_i 在第 j 个提供者的缓存中的位置。缓存采用先进先出的替换策略，位置指的是到缓存尾部的距离。
$urgency_i$	数据分片 D_i 的紧迫性。
$rarity_i$	数据分片 D_i 的稀缺性。
$priority_i$	数据分片 D_i 的获取优先权。

综合考虑每个数据分片的紧迫性与稀缺性，数据调度器通过公式(1)到(3)计算出每个数据分片的获取优先权。首先计算数据分片 D_i 的紧迫性：

$$R_i=\max\{R_{i_1},R_{i_2},\cdots,R_{i_n}\}$$

$$t_i=\frac{id_i-id_{play}}{p}-\frac{1}{R_i}\text{，那么 }urgency_i=\frac{1}{t_i} \qquad (1)$$

我们将数据分片 D_i 的稀缺性理解为 D_i 在其所有提供者的缓存中都被替换掉的概率，这比过去的文献中普遍将 D_i 的稀缺性理解为 $rarity_i=\frac{1}{n_i}$ 要更为合理：

$$rarity_i = (\frac{p_{i_1}}{B}) \times (\frac{p_{i_2}}{B}) \times \cdots \times (\frac{p_{i_{n_i}}}{B}) \qquad (2)$$

数据分片 D_i 的获取优先权 $priority_i = \max\{urgency_i, rarity_i\}$ (3)

计算出每个数据分片的获取优先权后，数据调度器将需要获取的数据分片按优先权降序排列，比如排列成形如 $D_1, D_2, \cdots, D_m$。对一个数据分片 D_i 来说，可能有多个邻居能提供 D_i，通常会选择能够最快传送 D_i 的邻居作为 D_i 的提供者。然而，这样的选择方式可能会出现冲突，比如当两个数据分片选择了同一个邻居作为提供者时，其中一个数据分片必须等待或者重选提供者。因此，为数据分片选择合适的提供者可表述为如下调度问题：如何为每一个数据分片选择一个合适的提供者，以使得过期或被替换的数据分片数目最少？实际上，即使是这个问题的一个简化版的特例（并行机调度问题 [CL90]）都已被证明是 NP 难的，因此我们使用算法 2.1 所示的贪心调度算法来试图尽早取得高优先权的数据分片，而不必追求总体最优的效果。

算法2.1 数据调度算法

输入：（1）按照获取优先权降序排列的数据分片 $D_1, D_2, \cdots, D_m$；
（2）每个数据分片的提供者集合 $S_1, S_2, \cdots, S_m$；
（3）结点 *j* 的发送数据率 *R(j)*；
（4）结点 j 处的预期排队时间 $\tau(j)$，初始时 $\tau(j)$=0。

输出：每个数据分片 D_i 的提供者 $supplier_i$。

算法：

1 计算在此调度周期内最多能接收的数据分片数目：$min(m, I \times \tau)$；
2 **for** *i*=1 to $min(m, I \times \tau)$ **do**
3 设置 D_i 的最早获取时间 $t_{\min} = \infty$；
4 假设 S_i 中包含 *k* 个提供者 $S_{i_1}, S_{i_2}, \cdots, S_{i_k}$；
5 **for** *j*=1 to *k* **do**
6 计算从 S_{i_j} 处获取 D_i 的预期时间 $t_{trans} = \frac{1}{R(S_{i_j})}$；
7 **if** $t_{trans} + \tau(S_{i_j}) < t_{\min}$ **and** $t_{trans} + \tau(S_{i_j}) < \tau$
8 $t_{\min} \leftarrow t_{trans} + \tau(S_{i_j})$； $supplier_i \leftarrow S_{i_j}$；
9 **if** $supplier_i \neq null$
10 $\tau(supplier_i) \leftarrow t_{\min}$；

2.3.4 数据预取算法

在每个调度周期中，数据调度器预测哪些数据分片很可能被数据调度算法遗漏。因为实际上不可能精确预测到数据分片被遗漏的可能性，所以我们设计了一套称为“紧迫界线”的机制让每个结点动态、自适应地预测。图 2.5 描画了紧迫界线机制的基本工作原理。缓存中的数据分片被紧迫界线（Urgent Line）分成两部分，左边的部分是紧迫的，右边的部分并不急于获取，灰色的部分是已获得的数据，白色部分尚未获得。因此，仅是紧迫界线左边的白

色数据分片被预测为可能遗漏。数据预取算法相关的参数列举在表 2.2 中。

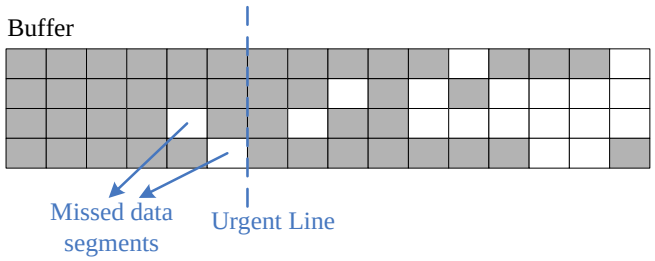


图 2.5 紧迫界线机制基本原理

表 2.2 数据预取算法的相关参数

参数	说明
α	缓存的紧迫比率，需动态计算。
B	缓存大小，即缓存中能放多少分片。
id_{head}	缓存头部的数据分片的 id。
id_{urgent}	紧迫界线对应的数据分片的 id。
N_{miss}	被预测为遗漏的数据分片数目。
k	每个数据分片需要备份在 k 个结点中。
l	数据预取算法在一个调度周期中最多能获取 l 个遗漏分片。

对照表 2 和图 4，可以得出紧迫界线机制的基本公式：$id_{urgent}=id_{head}+\alpha\times B$。可以看出 α 是紧迫界线机制的关键性参数，我们将在这一小节的最后部分详细讲述参数 α 的计算。所有尚未获得的 id 小于 id_{urgent} 的数据分片都被预测为遗漏数据，它们的总数目为 N_{miss}，根据 N_{miss} 与参数 l 的关系，数据预取算法采取不同行为：

情形 1：如果 $N_{miss}=0$，不需要启动数据预取算法；

情形 2：如果 $N_{miss}\le l$，启动数据预取算法并行获取所有遗漏的数据分片；

情形 3：如果 $N_{miss}>l$，不启动数据预取算法，因为遗漏分片太多，预取将带来过大开销；

对一个结点 n，假设其 Peer Table 中最近的结点是 n_1，那么 n 最终需要在其按需数据备份中保存 id 符合如下公式的数据：$hash(id\times i)\%N\in[n,n_1),i=1,2,\cdots k.$，$hash()$是一个普通的安全散列函数，%代表取模操作，这样每个数据分片最终会备份在 k 个不同的结点中。我们使用 $id\times i$ 来散列是为了平衡负载，如果使用 $id+i$ 来散列，那么连续的数据分片可能会聚集在同一个结点上导致该结点负载过重。如果结点 n 想离开系统，它应首先找到逆时针方向离 n 最近的结点 n'，然后将自身数据备份中的数据移交给 n' 来保存。如果结点 n 离开系统是突然的，数据移交就不可能发生，然而考虑到随着时间的流逝，n 所备份的数据变得越来越陈旧和无意义，所以 n 的突然离开并不会给系统带来多少影响。

算法 2.2 描述了数据预取算法的流程。假设被预测为遗漏的数据分片是 $D_1,D_2,\cdots,D_m$，对每个数据 D_i，首先需要定位到 k 个（有可能不足 k 个）备份了 D_i 的结点，然后选择能以最大发送速率传送 D_i 的结点作为 D_i 的预取提供者。当结点 n 向一个应当备份 D_i 的结点 n_0 索取 D_i 时，有可能 n_0 也没有获得 D_i，这个概率记为 P_{fail}，一般认为 n 和 n_0 有等价的概率获得 D_i，

所以平均来说 $P_{fail}=\frac{1}{2}$，那么 n 从 k 个应当备份 D_i 的结点处都不能获得 D_i 的概率是 $(\frac{1}{2})^k$，这个概率是相当低的，所以绝大多数情况下遗漏数据的预取操作都能成功。

算法 2.2 数据预取算法

输入：预测遗漏的数据分片 $D_1,D_2,\cdots,D_m$，按 *id* 升序排列；
输出：每个数据分片 D_i 的预取提供者 *on-demand supplier*$_i$；
算法：

```
1  for i=1 to m do
2    为获取 D_i，并行发送 k 条路由消息去寻找 k 个目标结点 n_1,n_2,...,n_k；
3    设置 D_i 的最大接收速率 R_i = 0；
4    for j=1 to k do
5      对于目标结点 n_j，最终将会找到逆时针方向离 n_j 最近的结点 n_j'；
7      if n_j' 已备份了 D_i and n_j' 的发送速率大于 R_i
8        on-demand supplier_i ← n_j'；
```

现在讲述数据预取算法的一个关键性参数 α 的计算。α 决定了缓存中有多大比例的数据是紧迫的，而由于流媒体系统工作的网络环境一直在变化，所以根据预取算法的执行效果来动态计算 α 是比较合理的。假设 t_{hop} 是覆盖网上路由一跳的平均时间，一般通过实验来估测，n 是覆盖网中预期的现存结点数，那么预取一个数据分片的平均时间将是 $t_{fetch}=t_{locate}+t_{reply}+t_{request}+t_{retrieve}\approx\frac{\log n}{2}\times t_{hop}+3\times t_{hop}=(\frac{\log n}{2}+3)\times t_{hop}$。

在一个调度周期 τ 中，数据预取算法必须为按期获得遗漏分片的过程分配足够的时间，也就是说必须满足 $\alpha\times B>p\times\tau$ 和 $\alpha\times B>p\times t_{fetch}$，即 $\alpha>\frac{p}{B}\times\max(\tau,t_{fetch})$，这就是 α 的下限和初始值。随着时间的推移，在出现下面两种情况时 α 需要被动态更新：

情形 1——数据过期：如果发现有预取的数据获得时已经过期，说明为预取操作分配的时间不足，因此必须增加缓存紧迫数据的比例，$\alpha\leftarrow\alpha+\frac{p\times t_{hop}}{B}$；

情形 2——数据重复：如果发现有预取的数据和数据调度算法按期取得的数据重复，说明预测为遗漏的数据比例过大，因此必须减少缓存紧迫数据的比例，$\alpha\leftarrow\alpha-\frac{p\times t_{hop}}{B}$。

α 的增量设置为 $\frac{p\times t_{hop}}{B}$，这是一个很小的增量，以使得 α 变化平滑。对于情形 2 来说，为了能够发现数据预取算法与数据调度算法所获得的重复数据，每个预取的数据分片都有特殊的标志以便于识别。

2.3.5 性能评价

a) 理论分析

这一小节从理论上分析 ContinuStreaming 系统的播放连续度。我们使用泊松过程作为基于 Gossip 协议的流媒体系统中数据分片的到来模型，因为泊松过程具有独立、平稳增量，这正是 Gossip 过程也具有的重要属性。具体来说，令 $N(t)$代表时间段 t 中到达一个结点的数据分片数目，根据泊松分布的定义有 $P\{N(t)=n\}=e^{-\lambda t}\frac{(\lambda t)^n}{n!}$，其中 λ 是一个常量。不难证明 $E[N(t)]=\sum_{n=0}^{\infty} n\cdot P\{N(t)=n\}=\sum_{n=0}^{\infty} n\cdot e^{-\lambda t}\frac{(\lambda t)^n}{n!}=\lambda t$，这正是参数 λ 被称为泊松过程到达速率的原因。对于基于 Gossip 协议的流媒体系统，将结点的接收带宽 I 近似当作数据分片的到达速率 λ。

为了让结点能连续播放流媒体数据，$\lambda>p$ 必须成立，其中 p 是播放速率。考虑一个调度周期中数据预取算法被启动的概率。在一个调度周期 τ 中，如果到来的数据分片数目 $N(t)$小于一个调度周期播放掉的分片数目，数据预取算法就很可能需要被启动，从而得出 $P\{\text{数据预取算法启动}\}=P\{N(\tau)\le p\tau\}=\sum_{n=0}^{p\tau} e^{-\lambda t}\frac{(\lambda t)^n}{n!}$，预期遗漏分片数目如下式所推导

$$N_{miss}=\sum_{n=0}^{p\tau-1}(p\tau-n)P\{N(\tau)=n\}=p\tau\sum_{n=0}^{p\tau-1}P\{N(\tau)=n\}-\sum_{n=0}^{p\tau-1}nP\{N(\tau)=n\}=p\tau\sum_{n=0}^{p\tau-1}e^{-\lambda t}\frac{(\lambda t)^n}{n!}-\sum_{n=0}^{p\tau-1}ne^{-\lambda t}\frac{(\lambda t)^n}{n!}$$

。

在 4.3 小节我们已估测出在预取过程中一个特定结点不能获得一个特定数据分片的概率为 $(\frac{1}{2})^k$，而在一个调度周期中预取遗漏的分片数目为 N_{miss}，所以在一个调度周期中能成功获取所有遗漏分片的概率是 $(1-(\frac{1}{2})^k)^{N_{miss}}$。从而我们可以计算出不使用数据预取算法的播放连续度 PC_{old}、使用预取算法的播放连续度 PC_{new} 以及它们之间的差值 $\Delta=PC_{new}-PC_{old}$，PC 代表 Playback Continuity。

$$PC_{old}=1-P\{N(\tau)\le p\tau\},\quad PC_{new}=1-P\{N(\tau)\le p\tau\}(1-(1-(\frac{1}{2})^k)^{N_{miss}}),$$

$$\Delta=PC_{new}-PC_{old}=P\{N(\tau)\le p\tau\}(1-(\frac{1}{2})^k)^{N_{miss}}。$$

为了检验上述理论分析结果的合理性，我们将理论结果与 1000 个结点规模的模拟实验结果进行比较，比较的结果列举在下表中。模拟实验的细节将在 5.2 小节讲述，几个主要的参数是：播放速率 p=10，结点平均接收带宽 I=15，调度周期 τ=1 秒，每个数据分片备份在 k=4 个结点中。同构指的是所有结点的接收带宽相等，异构指的是结点间有不同的接收带宽。

模拟环境	PC_{old}	PC_{new}	$\Delta=PC_{new}-PC_{old}$
理论分析结果，λ=15	0.8815	0.9989	0.1174
理论分析结果，λ=14	0.8243	0.9975	0.1732
同构、静态网络	0.8748	0.9979	0.1231
同构、动态网络	0.8520	0.9803	0.1283
异构、静态网络	0.8431	0.9726	0.1259
异构、动态网络	0.8166	0.9537	0.1371

从上表可以看出，模拟实验的结果介于 λ=15 和 λ=14 的理论分析结果之间，因为 I=15

的接收带宽中有很少的一部分被用作预取开销。动态或异构环境下模拟实验的 PC_{new} 一般要略小于理论分析结果，这主要归因于实验网络的搅动和其它一些限制条件。

b) 实验环境

我们在 30 幅真实 P2P 网络拓扑上对 ContinuStreaming 系统进行了模拟实验，这些拓扑数据来自于 http://dss.clip2.com 上收集的无结构 P2P 网络拓扑测量结果。拓扑数据中包含每个结点的 *ID*、*IP* 地址、端口号、*PING* 时间（从某个测量结点发出 *PING* 消息）等信息，我们仅使用 *ID*、*IP* 地址和 *PING* 时间信息。这 30 幅网络拓扑的结点规模从 100 到 10000 不等，平均结点度从 1 到 3.5 不等，因为这样的平均结点度对流媒体系统来说过低，所以我们在拓扑中随机加入了一些边让每个结点连接 M=5 个邻居。根据模拟实验的经验，M=5 一个比较合适的选择，使用更大的 M 并不能带来更多好处。

在所有现存的基于 Gossip 协议的 P2P 流媒体系统中，CoolStreaming 系统因其简单、实用而最具代表性，因此在模拟实验中我们也编写了 CoolStreaming 的模拟代码，比较在相同的网络环境中 CoolStreaming 与 ContinuStreaming 的性能。流媒体发布速率是 300Kbps，每个数据分片包含 30Kb，因此播放速率 p=300/30=10。每个结点维护包含 B=600 个数据分片的缓存，相当于 60 秒的流媒体数据。结点的接收带宽分布在 300Kbps 到 1Mbps 之间，平均接收带宽为 450Kbps，也就是说 $I\in[10,33]$，平均 I=15。结点的发送带宽和接收带宽具有类似的分布，唯一的例外是媒体源接收带宽为 0 而发送带宽很高，通常达到 I=100。数据调度周期 τ=1 秒。

P2P 覆盖网上两个结点之间的时延是它们在真实拓扑上 *PING* 时间的差值，这样的估测虽然不准确但对于我们的模拟实验是基本合理的。覆盖网上路由一跳的平均时延 $t_{hop}\approx 50$毫秒，因此预取一个数据分片的平均时间 $t_{fetch}\approx(\frac{\log n}{2}+3)\times t_{hop}=8\times 50\text{毫秒}=400\text{毫秒}$，这里 n=1000，n 取其它值时计算方法相同。缓存紧迫比率的初始值 $\alpha=\frac{p}{B}\times\max(\tau,t_{fetch})=\frac{10}{600}\times\max(1\text{秒},400\text{毫秒})=\frac{1}{60}$。此外，每个数据分片备份在 k=4 个结点中，数据预取算法每个周期最多获取 l=5 个数据分片。为了构建一个动态网络环境，每个周期随机让 5%的结点失效、再让 5%的新结点加入，新加入结点不需要获取过去播放过的所有数据，它们只需要获取其邻居结点正在播放或将要播放的数据，也就是说新加入结点是跟随其邻居的播放步伐。

c) 性能指标

我们主要使用下面 3 个性能指标来评价一个 P2P 流媒体系统：

（1）播放连续度：在每个调度周期中，记录下已获得足够多的数据分片来播放的结点比例，我们将这一比例定义为播放连续度。某些文献中使用“连续指标”（Continuity Index）来衡量播放连续度，连续指标指的是一个调度周期中按期到达的数据分片比例。不难发现，高连续指标并不能保证连续播放，但高播放连续度能保证连续播放，因此我们这里采用的定义方式更为合理。

（2）控制开销：在每个调度周期中，结点和其邻居交换数据可用性信息，控制开销定义为交换数据可用性信息的通信开销占实际流媒体数据传输的通信开销的比例。

（3）预取开销：为了预取一个数据分片，结点必须首先定位到 k 个备份该分片的结点，然后从其中选出一个结点作为提供者，这一过程带来大约 $k\times(\frac{\log n}{2}+1)+1$ 条路由消息的通信开销和传送一个数据分片的通信开销，这两方面的通信开销占实际流媒体数据传输的通信开

销的比例就定义为预取开销。

d) 实验结果

（1）播放连续度

我们首先记录下在 1000 个结点的静态网络中 ContinuStreaming 和 CoolStreaming 的播放连续度随时间变化的轨迹，见图 2.6。两个系统通常在 30 秒都可达到稳定状态，所以我们只记录它们前 30 秒的播放连续度。从图 2.6 可以看出，CoolStreaming 在第 26 秒到达稳定状态，稳定后的播放连续度大约 0.83，ContinuStreaming 在第 18 秒就到达稳定状态，稳定后的播放连续度大约 0.97，这证实了我们采用的数据预取方法加快了流媒体系统到达稳定状态的速度，更重要的是将稳定状态的播放连续度提升到接近 1.0 的水平。

比较了静态网络中的变化轨迹后，我们进一步比较了两个系统在 1000 个结点的动态网络中的播放连续度变化轨迹，见图 2.7。CoolStreaming 在第 27 秒到达稳定状态，稳定后的播放连续度大约 0.78，ContinuStreaming 在第 20 秒到达稳定状态，稳定后的播放连续度大约 0.95。虽然动态网络中的 PC_{new}=0.95 比静态网络中的 PC_{new}=0.97 要小，但动态网络中的播放连续度增幅 0.17 要高于静态网络中的增幅 0.14，从这一点来看，ContinuStreaming 在动态网络中能更多地提升播放连续度。

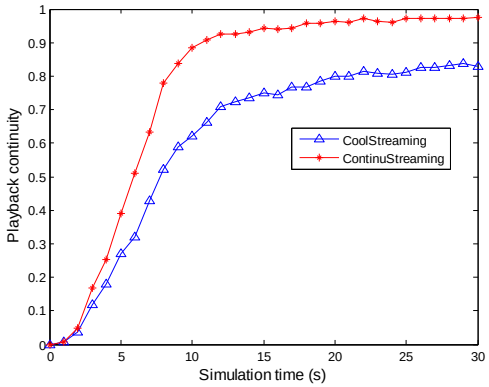


图 2.6 静态网络中的播放连续度轨迹

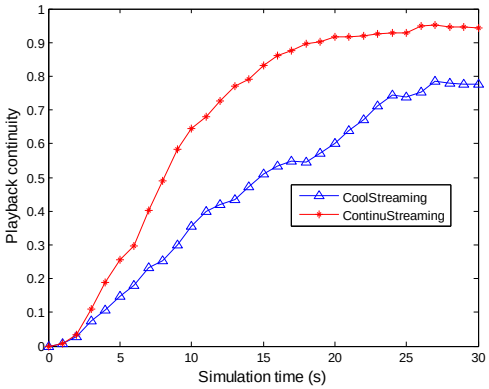
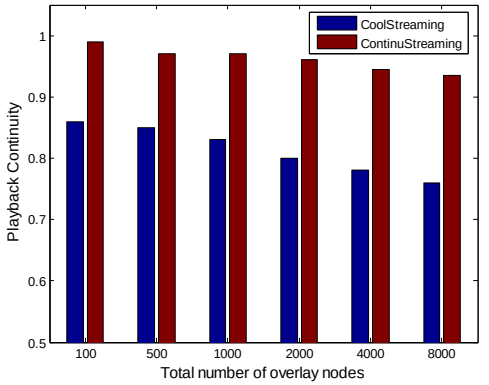


图 2.7 动态网络中的播放连续度轨迹

此外，我们还比较了不同规模网络稳定状态的播放连续度，网络结点数目在 100 到 8000 之间，静态环境和动态环境都做了实验。从图 2.8 和图 2.9 所展示的实验结果来看，随着网络规模的增大，PC_{new} 和 PC_{old} 都在减小，但播放连续度的增量 $\Delta = PC_{new} - PC_{old}$ 在增大，因此 ContinuStreaming 系统对规模越大的网络越有好处。



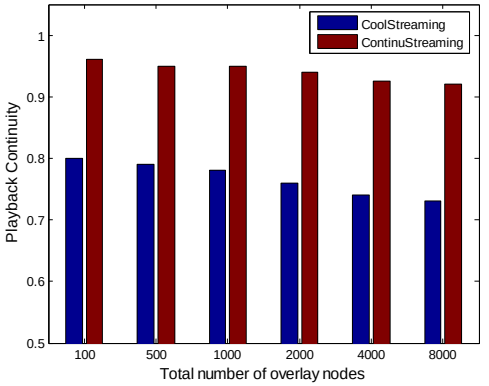


图 2.8 不同规模静态网络的播放连续度　　图 2.9 不同规模动态网络的播放连续度

（2）控制开销

在每个调度周期中，结点和其邻居交换数据可用性信息，控制开销就来自于此。我们记录下不同规模的网络对于不同的连接邻居数 M 的控制开销，见图 2.10。因为结点缓存 B=600 个数据分片，所以我们采用 600 比特的位图来记录缓存的数据可用性信息，比特 1 代表该分片可用，比特 0 代表该分片不可用。缓存的第一个分片需要 20 比特来记录其信息，因为数据源在一天内最多发布 $3600\times10\times24=864000\in(2^{19},2^{20})$个数据分片。因此，获取一个邻居的数据可用性信息总共需要 620 比特的通信开销。每个数据分片包含 30Kb 的数据，假设每个结点每秒能获得 p=10 个数据分片，这意味着每个结点的播放连续度都为 1.0，那么控制开销将是 $\frac{620\times M}{30\times1024\times10}=\frac{M}{495}$。图 2.10 所示的模拟结果比 $\frac{M}{495}$ 要略高，因为实际上很少有结点的播放连续度能保持 1.0。所有网络的控制开销都低于 2%，这是很低的数值，并且 ContinuStreaming 的控制开销同 CoolStreaming 相仿，原因在于两者的数据可用性信息交换机制非常类似。

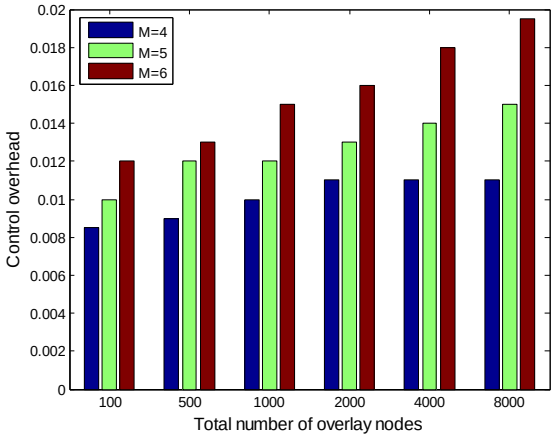


图 2.10 不同规模网络的控制开销

（3）预取开销

为了预取一个数据分片，结点必须首先定位到 k=4 个备份该分片的结点，然后从其中选出一个结点作为提供者，这一过程带来大约 $k\times(\frac{\log n}{2}+1)+1$ 条路由消息的通信开销和传送一个数据分片的通信开销。每条路由消息为 10 字节即 80 比特，当 n≤8000 时，预取一个数据分片的总开销将是 $(4\times(\frac{\log n}{2}+1)+1)\times80+30\times1024\approx33000$ 比特。

我们在图 2.11 中记录下一个包含 1000 个结点的网络在静态和动态环境下的预取开销随时间变化的轨迹。起初预取开销很低，因为大多数结点的预期遗漏分片数目 N_{miss} 高于阈值 l（实际上大多数结点那时还不知道媒体源的存在），因而预取算法没有被启动。几秒之后预取开销上升到略高于稳定状态，因为那时大多数结点已经知道了媒体源的存在。稳定状态的预取开销，对于静态网络是 2.3%，对于动态网络是 3%。

图 2.12 描画了不同规模的网络在静态和动态环境下的预取开销。很明显对于每个网络，动态环境下的预取开销要高于静态环境，这是因为动态环境下会有更多的数据分片被数据调

度算法遗漏。所有网络的预取开销都低于 4%，这表明我们为 ContinuStreaming 系统设计的预取方法仅带来微小的额外开销。

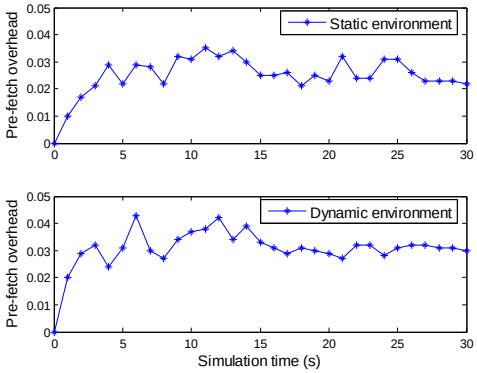


图 2.11 包含 1000 个结点的网络的预取开销轨迹

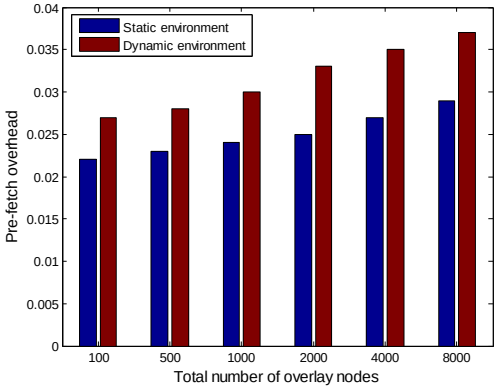


图 2.12 不同规模网络的预取开销

2.4小结与进一步的工作

近几年出现的基于 Gossip 协议的 P2P 流媒体系统因能灵活、高效地工作在动态、异构的因特网环境下而广受关注。然而，现存系统因 Gossip 协议数据传播的随机性缺陷而无法保持高播放连续度。本文采用基于分布式散列表（DHT）的数据预取方法来弥补 Gossip 协议的缺陷，设计了一个具有高播放连续度的 P2P 流媒体系统 ContinuStreaming，并通过理论分析和模拟实验证实了该系统的有效性。下一步我们想在实际的广域网中实现并测试 ContinuStreaming 系统，预期会发现更多值得研究的问题和与模拟实验有所不同的性能情况。

第3章 流媒体发布源的快速切换

3.1 背景与动机

前文已经讲过，基于Gossip协议的P2P流媒体系统已成为P2P流媒体系统的主流。在基于Gossip协议的P2P流媒体系统中可能存在一个或多个流媒体发布源，它们将流媒体源数据不断传播到系统中。对于一个多源系统来说，多个流媒体发布源可能是以“串行”或“并行”的方式之一来工作。比如说，在一个视频会议系统或远程教学系统中，每个成员结点都可以成为发布源，但通常任一时刻系统中仅有一个发布源，我们称这种方式为“串行”；而对于Internet实时网络电视系统来说，对应于多个电视频道的多个发布源通常是同时存在并工作的，我们称这种方式为“并行”。本文中我们考虑有多个流媒体发布源但它们串行工作的P2P流媒体系统，研究的关键问题是如何使得流媒体发布源之间能够快速切换，以尽量减少新发布源的启动时延。

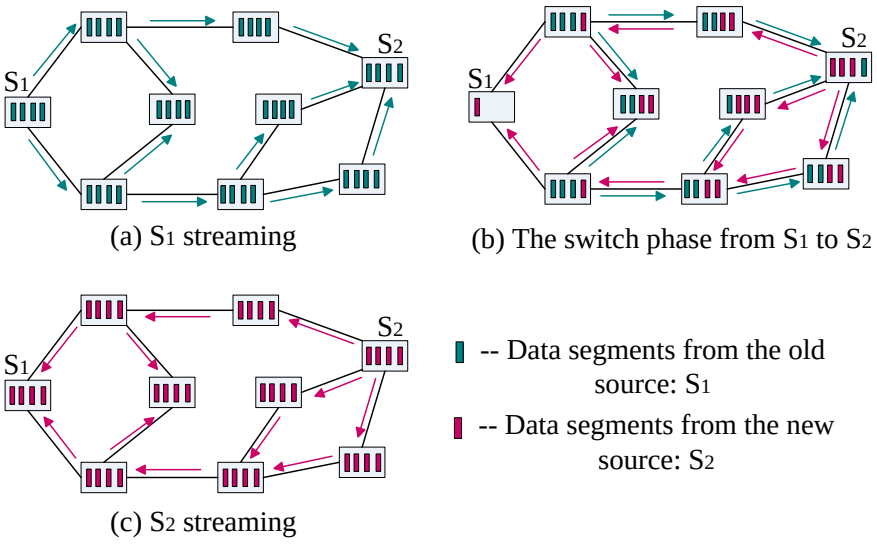


图3.1 源切换过程示例

图3.1显示了一次简单的源切换过程中的数据分布与流向。这一过程可以划分为3个阶段：（a）起初，旧的发布源 S_1 向其邻居结点发布流媒体数据，这些数据被其它结点依次传播到整个系统中；（b）旧发布源 S_1 停止发布数据，新发布源 S_2 开始发布数据，系统中同时存在来自 S_1 和 S_2 的数据流；（c）经过一段时间，系统中每个结点都完成了对来自 S_1 的流媒体的播放，只有来自 S_2 的数据在系统中不断传播。很明显，我们要研究的源切换问题可以归结为如何尽可能地缩短阶段（b）的持续时间。更具体地说，我们需要设计一个合适的源切换算法来最小化每个结点的源切换时间（源切换时间等价于新发布源 S_2 的播放启动时延），而必须遵守的限制性前提是——某个结点可以开始对来自 S_2 的流媒体的播放当且仅当：1）该结点已经完成了对来自 S_1 的流媒体的播放，2）该结点已经接收到了足够多的来自 S_2 的数据（以保证 S_2 的平滑启动）。

3.2 国际研究现状

前文曾经说到过，P2P流媒体系统可以优化的属性有：播放连续度、启动时延、带宽利用率、端到端时延、数据吞吐量、系统容错性等等，本文所要优化的属性是发布源切换时间，也就是新发布源的启动时延，这一属性类似于前文说过的“启动时延”，但比启动时延要更为复杂，因为对发布源切换时间的优化要同时考虑旧发布源和新发布源的播放需求。另一方面，由于我们优化了新发布源的启动时延，也就间接地优化了前文提到的启动时延和带宽利用率，有一举多得的效果。

CoolStreaming 使用基于 Gossip 协议的 P2P 网络多播技术构建了一个灵活、实用的 P2P 流媒体系统 [ZL05b]。它提供了对多个发布源的支持，但并没有描述其发布源切换机制。P2P 流媒体系统 AnySee [LJ06] 采用覆盖网间的优化方法（Inter-overlay Optimization）来减少数据源到接收端的时延，从而减少启动时延，但端到端时延与启动时延都不同于发布源切换时间。

Zhang 等人观察到 Gossip 协议采用“拉”数据的方法虽然比“推”数据要节省带宽，却带来了更大的数据传播时延，因此他们设计了一个“推拉结合”的系统 GridMedia [ZL05a]。他们将 P2P 流媒体系统中的数据分片分成两类：一类数据分片只在被请求获取时才传播，称为“拉数据”；另一类数据分片一旦结点收到就立即传播给邻居，称为“推数据”。GridMedia 系统主要的设计目标是减少数据传播时延，它间接地减少了发布源切换时间。然而，推数据的方法必然带来相当大的通信开销，而且也不能从本质上保证源切换时间的减少比率。

Xu 等人考虑了 P2P 流媒体系统中的数据率分配问题 [XH02]。P2P 流媒体系统中每一个结点都连有若干带宽不等的邻居，Xu 等人设计了一个名为 OTS_{P2P} 的数据率分配算法，能够保证当前结点具有最小的数据缓冲时延，从而减少启动时延。然而，OTS_{P2P} 算法成立的前提非常苛刻，它要求系统结点的带宽满足严格的倍数关系，实际的 P2P 流媒体系统是无法满足该前提的。

3.3 流媒体发布源的快速切换算法

本文首先通过对本质特性与关键参数的分析给数据源切换过程建立数学模型，从而将源切换问题形式化为一个数学优化问题，然后推导出此数学优化问题的最优解。鉴于实际系统情况的复杂性与网络环境的动态性，我们提出了一个称为“快速源切换算法”的实用贪心算法，它通过交错旧源与新源的数据传递来趋近理论上的最优解。快速源切换算法是低开销与纯分布式的，每个结点独立地启动并执行该算法，并且算法执行所依赖的仅仅是本地信息。

我们在多个真实测量的覆盖网拓扑结构上做了大量模拟实验来证实快速源切换算法的有效性，覆盖网结点数目从100到10000不等。模拟实验的结果显示：我们提出的快速源切换算法相比“传统源切换算法”能节省20%-30%的切换时间，同时不会带来额外的通信开销，并且切换时间的减少比例随着系统规模的增大而趋于增加。“传统源切换算法”并不交错旧源与新源的数据传递——它总是优先处理旧源数据的传递。图3.2中的示例比较了两者的不同：当前结点在每个数据调度周期中可以获取7个数据分片，而当前来自旧源 S_1 的可用数据分片是1、2、3、4、5，来自新源 S_2 的可用数据分片是6、7、8、9、10，传统切换算法优先处理 S_1 的数据传递，因此它将获取1、2、3、4、5、6、7这7个数据分片，而快速切换算法根据它特殊的数据获取优先权计算原则交错处理 S_1 、S_2 的数据传递，它将数据分片1、2、3与6、7、8、9的传递交错在一起。

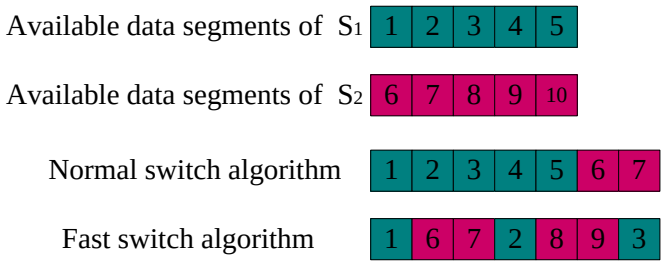


图3.2 快速切换算法和传统切换算法的比较示例

总体说来，本文对领域的贡献可大致总结为如下3个方面：

（1）就我们目前所知，本文是P2P流媒体领域第一个对媒体发布源切换问题做出的深入研究。我们给源切换过程建立了数学模型，从而将源切换问题形式化为一个数学优化问题，并推导出此数学优化问题的最优解。

（2）提出了一个称为“快速源切换算法”的实用贪心算法，它通过交错旧源与新源的数据传递来趋近理论上的最优解。

（3）通过在多个真实测量的覆盖网拓扑结构上进行的大量模拟实验，证实了快速源切换算法的有效性。

3.3.1 源切换过程建模

因为P2P流媒体系统的工作方式是纯分布式的，所以一个结点直到在其邻居结点中发现有来自新发布源的数据分片时才能得知发布源切换过程的开始，也就是说，运行于每个结点的源切换算法不会假设有任何源切换“先验”顺序的存在。当结点发现有新发布源的存在时，它就启动并执行源切换算法，然后在每个数据调度周期再次执行源切换算法，直到完成了旧发布源的所有数据的播放，也就是完成了源切换过程。虽然源切换算法不假设有任何源切换“先验”顺序的存在，但系统中必须存在一种新发布源 S_2 与旧发布源 S_1 之间的同步机制，让 S_2 能够获知 S_1 结束于哪个数据分片，并将 S_1 的最后一个数据分片的*id*加入到 S_2 最初的几个数据分片中以通知系统其它结点。这一同步机制并非本文考虑的重点，所以我们不做详述。

源切换过程建模所需的参数列举在表3.1中。我们使用图3.3来图形化表3.1中的参数以使抽象的参数更加具体。每当收集到来自 S_1 的Q个连续的数据分片时，就播放 S_1，但是为启动 S_2 的播放，则需要获取 Q_S 个数据分片。在流行的实用P2P流媒体系统中，Q_S 通常要比Q大很多以保证新发布源的平滑启动。当前结点的总输入带宽I被被分配到 I_1、I_2 两个部分，以分别获取来自 S_1、S_2 的数据分片。I_1、I_2 由源切换算法动态分配。

表3.1 源切换过程建模的相关参数

参数	说明
S_1	旧发布源。
S_2	新发布源。
Q	每当收集到来自 S_1 的 Q 个连续的数据分片时，就播放 S_1。
Q_1	来自 S_1 的数据分片还有 Q_1 个尚未获取。
Q_S	为启动 S_2 的播放所需获取的数据分片总数。
Q_2	为启动 S_2 的播放，还差 Q_2 个数据分片尚未获取。初始时，$Q_2 = Q_S$。
p	每秒播放的数据分片数目。
I	当前结点的总输入带宽。I 是以每秒的输入数据分片数目来衡量的，它是一

	个常数。
I_1	分配用来获取来自 S_1 的数据分片的输入带宽。I_1 是动态调整的。
I_2	分配用来获取来自 S_2 的数据分片的输入带宽。I_2 是动态调整的。
T_1	获取到来自 S_1 的所有数据分片的预期时间。
T_1'	完成 S_1 播放的预期时间。
T_2	获取到来自 S_2 的最初 Q_S 个数据分片的预期时间。

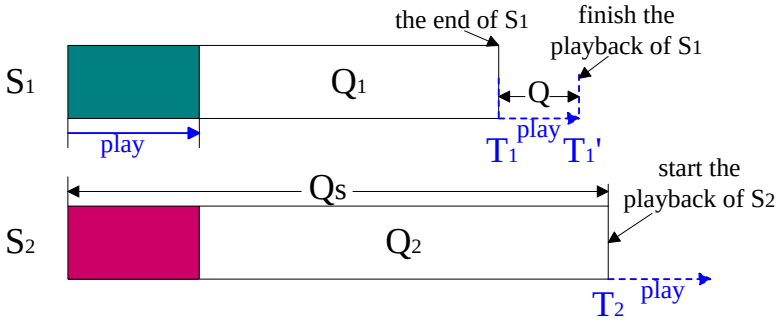


图3.3 源切换过程的时序图

对照表3.1和图3.3，源切换问题可以被形式化成如下所示的数学优化问题：

优化目标：最小化 T_2

限制条件：

$$\begin{cases} I = I_1 + I_2 \\ T_1 = \dfrac{Q_1}{I_1} \\ T_1' = T_1 + \dfrac{Q}{p} \\ T_2 = \dfrac{Q_2}{I_2} \\ T_2 \geq T_1' \end{cases}$$

上面的限制条件可以简化为：

$$\begin{cases} T_1' = \dfrac{Q_1}{I_1} + \dfrac{Q}{p} \\ T_2 = \dfrac{Q_2}{I - I_1} \\ T_2 \geq T_1' \end{cases}$$

从而可以得到不等式：$\dfrac{Q_2}{I - I_1} \geq \dfrac{Q_1}{I_1} + \dfrac{Q}{p}$；

上面的不等式可改写为：$I_1^2 + (\dfrac{p(Q_1 + Q_2)}{Q} - I)I_1 - \dfrac{pIQ_1}{Q} \geq 0$；　　（式3.1）

式3.1实际上是一个关于 I_1 的一元二次不等式，它的数学解为 $I_1 \geq r_1$ 或 $I_1 \leq r_1'$，其中

$$r_1=\frac{I-\frac{p(Q_1+Q_2)}{Q}+\sqrt{(\frac{p(Q_1+Q_2)}{Q}-I)^2+\frac{4pIQ_1}{Q}}}{2}，\quad r_1'=\frac{I-\frac{p(Q_1+Q_2)}{Q}-\sqrt{(\frac{p(Q_1+Q_2)}{Q}-I)^2+\frac{4pIQ_1}{Q}}}{2}。$$

很明显，$r_1'<0$，因此$I_1\geq r_1$是式3.1的唯一合理解。为实现最小化T_2的优化目标，应该让$I_1=r_1$，$I_2=r_2=I-r_1$，这是源切换问题的理论最优解。

3.3.2 快速源切换算法

上一小节得出了源切换问题的理论最优解，但当应用到实际P2P流媒体系统中时，上述理论最优解通常不能完全适用，主要原因在于实际的互联网环境比模型中的环境要复杂的多，并且面临更多的限制性条件。因此，我们需要设计一个能趋近理论最优解的实用算法。

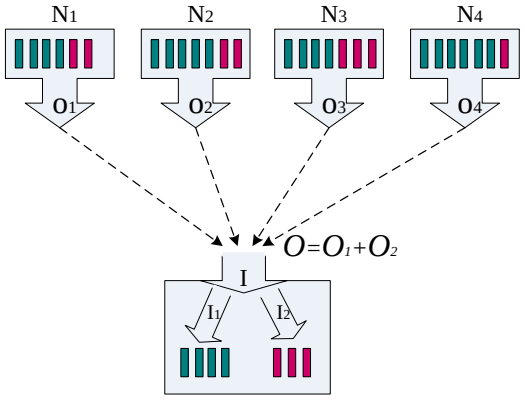


图3.4 结点的局部工作环境示意图

图3.4描画了P2P流媒体系统中一个普通结点的局部工作环境。本地结点有4个邻居：N_1、N_2、N_3、N_4，4个邻居的输出带宽分别是o_1、o_2、o_3、o_4。假设O_1是针对源S_1的总输出带宽，O_2是针对源S_2的总输出带宽，那么上一小节中的数学优化问题可进一步归结为：

优化目标：最小化T_2

限制条件：

$$\begin{cases} I_1+I_2\leq I \\ I_1\leq O_1, I_2\leq O_2 \\ T_1=\dfrac{Q_1}{I_1} \\ T_1'=T_1+\dfrac{Q}{p} \\ T_2=\dfrac{Q_2}{I_2} \\ T_2\geq T_1' \end{cases}$$

在增加了更多的限制条件后，上一小节得出的理论最优解$I_1=r_1$、$I_2=r_2$仅当$r_1\leq O_1$、$r_2\leq O_2$时才成立。因此，当$r_1>O_1$或$r_2>O_2$时，优化的目标将改为最大化本地结点的输入吞吐量，从而现实环境下的源切换问题的最优解是：

情形1：当$r_1\leq O_1$且$r_2\leq O_2$时，$I_1=r_1$、$I_2=r_2$。

情形2：当 $r_1 \le O_1$ 且 $r_2 > O_2$ 时， $I_1 = \min(O_1, I - O_2)$ 、 $I_2 = r_2$ 。
情形3：当 $r_1 > O_1$ 且 $r_2 \le O_2$ 时， $I_1 = r_1$ 、 $I_2 = \min(I - O_1, O_2)$ 。
情形4：当 $r_1 > O_1$ 且 $r_2 > O_2$ 时， $I_1 = O_1$ 、 $I_2 = O_2$ 。

这样，计算源切换问题的最优解的关键是首先计算出 O_1 和 O_2 ，更确切地，从数据分片的微观角度来看，是计算两个集合 O_1 和 O_2 ，其中 $|O_1| = O_1$ ， $|O_2| = O_2$ 。集合 O_1 和 O_2 中的数据分片都按照获取优先权降序排列。计算集合 O_1 和 O_2 所需要的参数列举在表3.2中，实际上这些参数正是我们下面将要设计的快速源切换算法的相关参数。

表3.2 快速源切换算法的相关参数

参数	说明
τ	数据调度周期。
id_i	数据分片 D_i 的 id。
n_i	能够提供 D_i 的邻居数目。
R_{i_j}	从第 j 个提供者那里获取 D_i 的速率。
R_i	数据分片 D_i 的最大获取速率。
id_{play}	当前正在播放的数据分片的 id。
id_{end}	S_1 的最后一个数据分片的 id。
id_{begin}	S_2 的第一个数据分片的 id。我们设定 $id_{begin} = id_{end} + 1$。
p	每秒播放的数据分片数目。
t_i	数据分片 D_i 的过期时间。
B	缓存大小，即缓存中能放多少分片。
p_{i_j}	数据分片 D_i 在第 j 个提供者的缓存中的位置。缓存采用先进先出的替换策略，位置指的是到缓存尾部的距离。
$urgency_i$	数据分片 D_i 的紧迫性。
$rarity_i$	数据分片 D_i 的稀缺性。
$priority_i$	数据分片 D_i 的获取优先权。

在计算一个数据分片的获取优先权时，我们综合考虑了数据分片的稀缺性与紧迫性。首先计算数据分片 D_i 的紧迫性：

$$R_i = \max\{R_{i_1}, R_{i_2}, \cdots, R_{i_n}\}$$

$$t_i = \frac{id_i - id_{play}}{p} - \frac{1}{R_i} \text{ ，那么 } urgency_i = \frac{1}{t_i}$$

我们将数据分片 D_i 的稀缺性理解为 D_i 在其所有提供者的缓存中都被替换掉的概率，这比过去的文献中普遍将 D_i 的稀缺性理解为 $rarity_i = \frac{1}{n_i}$ 要更为合理：

$$rarity_i = (\frac{p_{i_1}}{B}) \times (\frac{p_{i_2}}{B}) \times \cdots \times (\frac{p_{i_{n_i}}}{B})$$

数据分片 D_i 的获取优先权 $priority_i = \max\{urgency_i, rarity_i\}$

得出每个数据分片的获取优先权后，我们设计的快速源切换算法就能计算出集合 O_1 和 O_2 从而安排数据分片的获取过程，见算法 3.1。数据分片按优先权降序排列，比如排列成形

如 $D_1, D_2, \cdots, D_m$，通常来自源 S_1 和源 S_2 的数据分片在这个序列中是交错排列的。对一个数据分片 D_i 来说，可能有多个邻居能提供 D_i，通常会选择能够最快传送 D_i 的邻居作为 D_i 的提供者。然而，这样的选择方式可能会出现冲突，比如当两个数据分片选择了同一个邻居作为提供者时，其中一个数据分片必须等待或者重选提供者。因此，为数据分片选择合适的提供者可表述为如下调度问题：如何为每一个数据分片选择一个合适的提供者，以使得过期或被替换的数据分片数目最少？实际上，即使是这个问题的一个简化版的特例（并行机调度问题 0）都已被证明是 NP 难的，因此我们使用算法 3.1 所示的贪心调度算法来试图尽早取得高优先权的数据分片，而不必追求总体最优的效果。

计算出集合 O_1 和 O_2 之后，I_1 和 I_2 的计算按照前面讲过的最优解的 4 种情形之一来计算，得出 I_1 和 I_2 后，数据分片的获取过程是很直接的。

算法3.1 快速源切换算法

输入：（1）按照获取优先权降序排列的数据分片 $D_1, D_2, \cdots, D_m$；

（2）每个数据分片的提供者集合 $S_1, S_2, \cdots, S_m$；

（3）结点 *j* 的发送数据率 *R(j)*；

（4）结点 j 处的预期排队时间 $\tau(j)$，初始时 $\tau(j)=0$。

算法：

（1）计算集合 O_1 和 O_2：

1 **for** *i*=1 to *m* **do**

2 设置 D_i 的最早获取时间 $t_{\min} = \infty$；

3 假设 S_i 中包含 *k* 个提供者 $S_{i_1}, S_{i_2}, \cdots, S_{i_k}$；

4 **for** *j*=1 to *k* **do**

5 计算从 S_{i_j} 处获取 D_i 的预期时间 $t_{trans} = \dfrac{1}{R(S_{i_j})}$；

6 **if** $t_{trans} + \tau(S_{i_j}) < t_{\min}$ **and** $t_{trans} + \tau(S_{i_j}) < \tau$

7 $t_{\min} \leftarrow t_{trans} + \tau(S_{i_j})$； $supplier_i \leftarrow S_{i_j}$；

8 **if** $supplier_i \neq null$

9 $\tau(supplier_i) \leftarrow t_{\min}$；

10 将 D_i 加入相应的集合 O_1 或 O_2。

（2）安排数据获取：

1 根据 O_1、O_2、r_1、r_2 计算 I_1 和 I_2；

2 获取 O_1 中的前 I_1 个数据分片；

3 获取 O_2 中的前 I_2 个数据分片。

3.3.3 性能评价

a) 实验环境

为了评价快速源切换算法的性能，我们在 30 幅真实 P2P 网络拓扑上对进行了模拟实验。这些拓扑数据来自于 http://dss.clip2.com 上收集的无结构 P2P 网络拓扑测量结果（遗憾的是，

该网站目前已不可用）。拓扑数据中包含每个结点的 *ID*、*IP* 地址、端口号、*PING* 时间（从某个测量结点发出 *PING* 消息）等信息，我们仅使用 *ID*、*IP* 地址和 *PING* 时间信息。这 30 幅网络拓扑的结点规模从 100 到 10000 不等，平均结点度从 1 到 3.5 不等，因为这样的平均结点度对流媒体系统来说过低，所以我们在拓扑中随机加入了一些边让每个结点连接 M=5 个邻居。根据模拟实验的经验，M=5 是一个比较合适的选择，使用更大的 M 并不能带来更多好处。

流媒体发布速率是 300Kbps，每个数据分片包含 30Kb，因此播放速率 p=300/30=10。每个结点维护包含 B=600 个数据分片的缓存，相当于 60 秒的流媒体数据。结点的接收带宽分布在 300Kbps 到 1Mbps 之间，平均接收带宽为 450Kbps，也就是说 $I\in[10,33]$，平均 I=15。结点的发送带宽和接收带宽具有类似的分布，唯一的例外是媒体源接收带宽为 0 而发送带宽很高，通常达到 I=100。数据调度周期 τ=1 秒。

在每次模拟中，首先让系统运行足够的时间以到达其稳定状态，然后停止旧源 S_1 的数据发布、启动新源 S_2 的数据发布。需要注意的是，在下文的每幅实验结果图中，时刻“0”均指旧源 S_1 停止、新源 S_2 启动的时刻。每当收集到来自 S_1 的 Q=10 个连续的数据分片时，就播放 S_1，但是为启动 S_2 的播放，则需要获取 Q_S=50 个数据分片。

我们比较了快速源切换算法和前文中讲过的传统源切换算法。这里再重复一次传统源切换算法的工作原理：假设本地结点为 n，当 n 的邻居可以向 n 提供来自 S_1 和 S_2 的数据分片时，n 将优先获取来自 S_1 的数据。如果在获取 S_1 的数据之后仍然有剩余输入带宽，就把剩余的输入带宽分配给 S_2 的数据获取。

b) 性能指标

我们主要使用下面 3 个性能指标来评价源切换算法：

（1）新源 S_2 的平均准备时间，即平均源切换时间：所有结点准备足够的 S_2 数据分片来启动 S_2 播放的平均时间；

（2）切换时间减少比率：快速源切换算法比传统源切换算法减少的切换时间比率；

（3）通信开销：在每个调度周期中，结点和其邻居交换数据可用性信息，通信开销定义为交换数据可用性信息的开销占实际流媒体数据传输的开销的比例。

此外，我们还测量了其它一些辅助性的性能指标来帮助更好地理解源切换过程。这些辅助性的性能指标包括：（1）旧源 S_1 的未传递比率（$=\frac{Q_1}{Q_0}$）：当前尚未传递的 S_1 的数据（Q_1）占时刻0未传递的 S_1 的数据（Q_0）的比率；（2）新源 S_2 的已传递比率（$\frac{Q_S-Q_2}{Q_S}$）：已传递的 S_2 的数据（Q_S-Q_2）占启动 S_2 播放所需数据（Q_S）的比率；（3）旧源 S_1 的平均完成时间（$=T_1'$）：所有结点完成 S_1 播放的平均时间。

c) 静态环境的模拟实验结果

我们首先对1000个结点的静态网络环境下的数据传递比率进行跟踪，从而在细致、微观的程度上对比快速源切换算法与传统源切换算法的性能差异。从图3.5的实验结果中可以看出，传统源切换算法能够更快地收集到播放 S_1 所需要的数据，但是在收集启动 S_2 播放所需数据的方面却要慢一些，这与直观上的感觉是一致的，也就是说，快速源切换算法折中了收集 S_1 数据的时间与收集 S_2 数据的时间，从而使整个源切换过程加快。

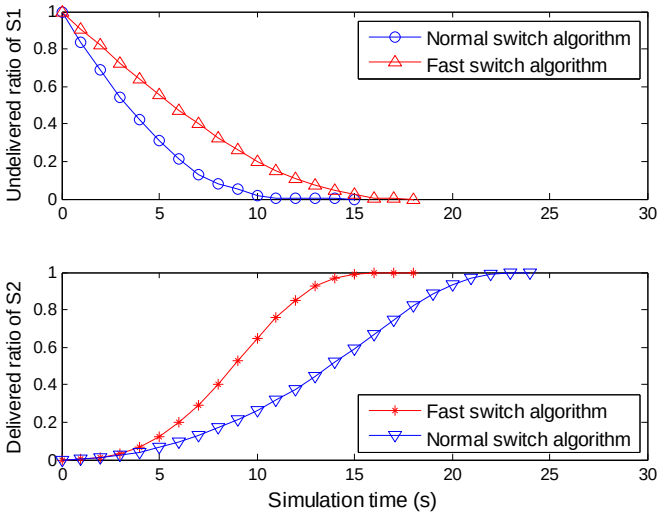


图3.5 1000个结点的静态网络环境下的数据传递比率跟踪

我们进一步测量了静态网络环境下旧源 S_1 的平均完成时间与新源 S_2 的平均准备时间，结点规模从100-8000不等，以使得测量结果更具通用性。图3.6中的柱状图描画了测量结果。对于每个规模的网络环境，从左到右都有4条方柱对应4项指标：1）使用传统源切换算法，旧源 S_1 的平均完成时间；2）使用快速源切换算法，旧源 S_1 的平均完成时间；3）使用快速源切换算法，新源 S_2 的平均准备时间；4）使用传统源切换算法，新源 S_2 的平均准备时间。在每个规模的网络环境下，实验结果都趋于一致：快速源切换算法缩小了旧源 S_1 的平均完成时间与新源 S_2 的平均准备时间之间的差异，从而减少了源切换时间。为了更清晰地描画出快速源切换算法比传统源切换算法减少的切换时间，我们将两种算法各自的切换时间及减少的比率描画在图3.7中。从图3.7可以看出，快速源切换算法相比传统源切换算法能减少约20%-30%的切换时间，并且减少的比率随着网络规模的增大而趋于增加。

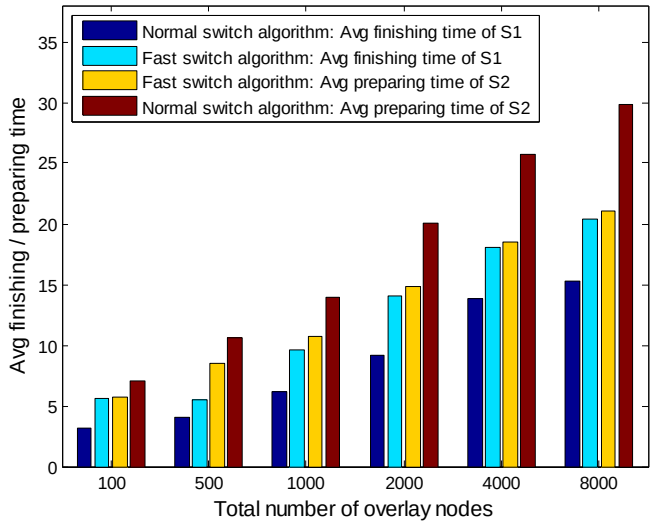


图3.6 静态网络环境下旧源 S_1 的平均完成时间与新源 S_2 的平均准备时间

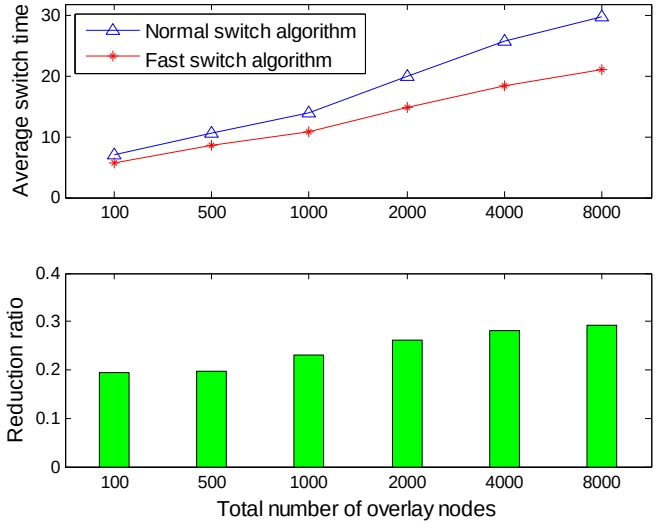


图3.7 静态网络环境下快速源切换算法比传统源切换算法减少的切换时间

此外，我们还测量了两种算法在不同规模的网络中运行的通信开销。结点缓存 B=600 个数据分片，所以我们采用 600 比特的位图来记录缓存的数据可用性信息，比特 1 代表该分片可用，比特 0 代表该分片不可用。缓存的第一个分片需要 20 比特来记录其信息，因为数据源在一天内最多发布 $10\times3600\times24=864000\in(2^{19},2^{20})$个数据分片。因此，获取一个邻居的数据可用性信息总共需要 620 比特的通信开销。每个数据分片包含 30Kb 的数据，假设每个结点每秒能获得 p=10 个数据分片，这意味着每个结点的播放连续度都为 1.0，那么控制开销将是 $\frac{620\times M}{30\times1024\times10}=\frac{5}{495}\approx1\%$，$M$=5。图 3.8 所示的模拟结果比 1%要略高，因为实际上很少有结点的数据获取速率能始终保持数据播放速率的水平。快速源切换算法的通信开销比传统源切换算法略低，因为前者通过缩短源切换时间的努力，间接地提高了带宽利用率。

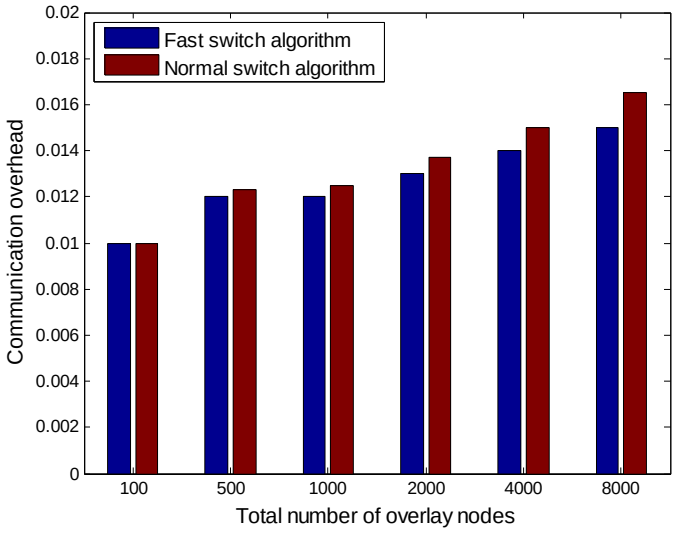


图3.8 静态网络环境下的通信开销

d) 动态环境的模拟实验结果

为了构建一个动态网络环境，每个周期随机让 5%的结点失效、再让 5%的新结点加入，新加入结点不需要获取过去播放过的所有数据，它们只需要获取其邻居结点正在播放或将要播放的数据，也就是说新加入结点是跟随其邻居的播放步伐。

动态网络环境下的实验步骤与静态网络环境下的基本相同，实验结果在图3.9、3.10、3.11和3.12中。总体上看，动态网络环境下的实验结果和静态网络环境下的基本一致，说明我们设计的算法对网络环境具有普适性。

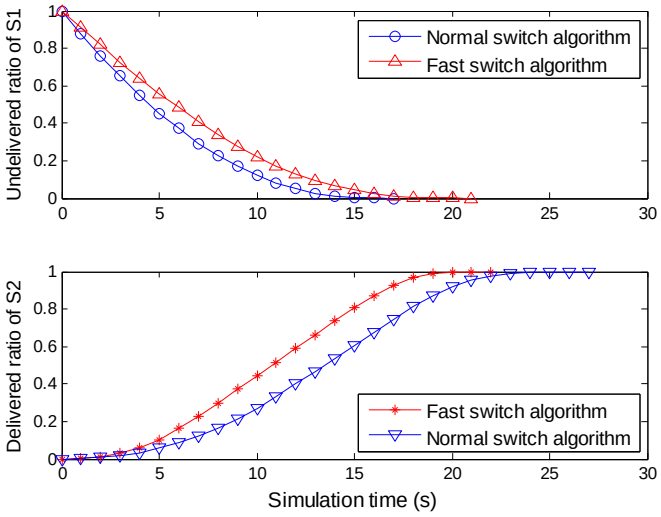


图3.9 1000个结点的动态网络环境下的数据传递比率跟踪

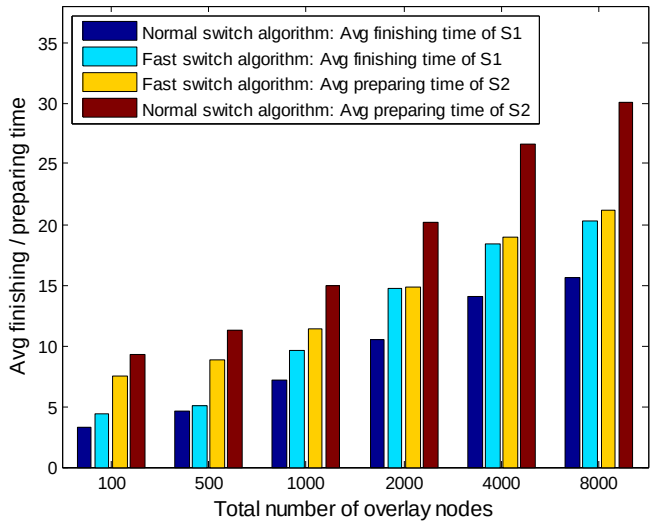


图3.10 动态网络环境下旧源 S_1 的平均完成时间与新源 S_2 的平均准备时间

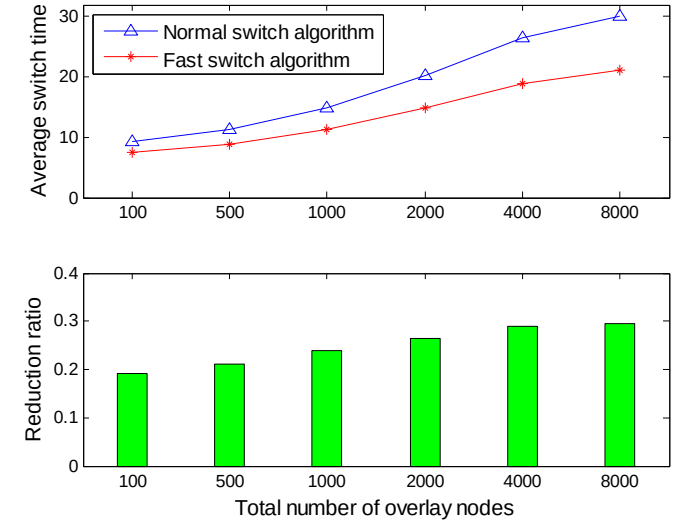


图3.11 动态环境下快速源切换算法比传统源切换算法减少的切换时间

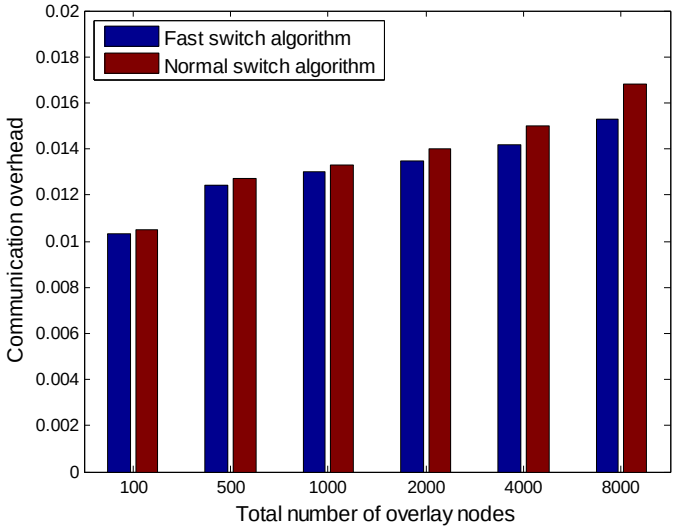


图3.12 动态网络环境下的通信开销

3.4 小结与进一步的工作

本文给P2P流媒体系统的媒体发布源切换过程建立了数学模型，从而将源切换问题形式化为一个数学优化问题，然后推导出此数学优化问题的最优解。鉴于实际系统情况的复杂性与网络环境的动态性，我们提出了快速源切换算法这一实用贪心算法，它通过交错旧源与新源的数据传递来趋近理论上的最优解。模拟实验的结果证实了快速切换算法的有效性。本文的工作针对的是发布源之间串行工作的多源P2P流媒体系统，下一步我们希望能扩展本文的工作到发布源之间并行工作的多源P2P流媒体系统中。

第4章 P2P流媒体系统底层覆盖网的拓扑优化

4.1 背景与动机

前文已经讲过，主流的P2P流媒体系统均采用无结构对等网络作为其底层覆盖网，底层覆盖网的性能对于构架于其上的P2P流媒体系统的性能有重要的甚至是决定性的意义。虽然对等网络中的结点在功能上互相平等，但实际上其中某些结点对于整个网络却有特殊意义。一方面，那些作为两个或多个独立子网之间唯一通道的结点，对覆盖网拓扑结构有重要影响，其失效很可能导致覆盖网被分割，而反过来覆盖网被分割往往可以追溯到它们的失效。另一方面，那些成为系统性能瓶颈的结点，其能力影响到系统在动态环境下工作的多方面性能，如查询成功率、数据（结点）定位跳数、通信拥塞和时延等等。我们称上述特殊意义的结点为“拓扑关键点”。如果能在对等网络中以分布式的方法有效地检测到拓扑关键点并加以合理的消除，就能从本质上优化覆盖网的拓扑结构，增强系统对覆盖网分割的抵抗力，同时显著地提高系统容错性。

最基本的问题是：什么样的结点是拓扑关键点？如果将对等网络看成一幅图（有向图或无向图，本文出于简化通常考虑无向图，但所提出的概念、方法对有向图是同样适用的），很自然会想到图论中的“割点”。割点的概念可以表述如下：如果在一幅连通图G中删去点C后，图G被分割成两个或多个独立的连通子图，那么点C就是图G的一个割点。图4.1是割点的简单例子：

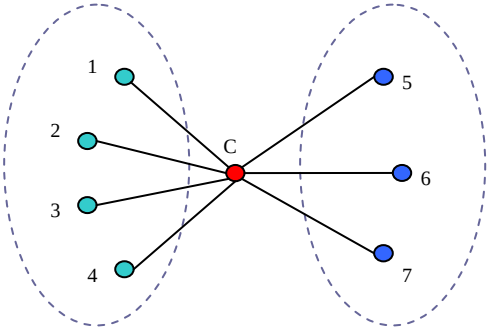


图4.1 点C为割点（仅画出C的邻居和边，图中其它结点和边均未画出）

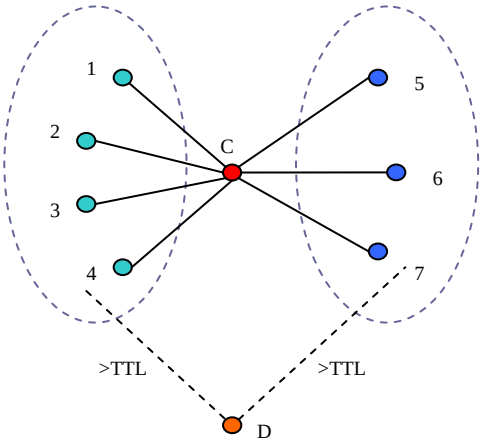


图4.2 点C不是割点，但它是分点，并且是名副其实的拓扑关键点

作为对等网络领域中重要而实用的部分，无结构对等网络有其独特的性质：覆盖网组织松散、没有固定的结构，通常采用洪泛式的路由方法或者基于洪泛的改进方法（如扩展环、随机走等等）。为了限制通信开销，无结构对等网络通常会给路由消息设定跳数限制（TTL，Time To Live）以使其局限在一定范围内，因此其路由具有明显的局部性和不确定性，不能覆盖整个网络，也就无法准确地定位结点或者数据对象。

鉴于无结构对等网络的工作特点，我们认为割点不能充分、合理地描述无结构对等网络

的拓扑关键点，因为割点是全局的、静止的，而无结构对等网络则是在动态环境下分布式工作、局部性路由的，既不可能静止地获取网络全局状态，也不可能做全局性的定位。图4.2的简单例子说明了割点不适合用来描述无结构对等网络的拓扑关键点。

图4.2中，点C并非割点，因为还有点D连接着左右两个子图。但是，D与左右两个子图的距离超过跳数限制TTL，所以如果C失效，左右子图之间虽然从图论上看仍然连通，但对无结构对等网络而言，却是相互分割的，因为两个子图之间从此没有结点可以互相定位到。在这种情况下，D的存在对左右子图并无意义，C既是连接左右子图的唯一通道，也是左右子图间通信的性能瓶颈，是名副其实的拓扑关键点，此时，我们称C为“分点”。

本文首先提出了“分点”这一概念来描述无结构对等网络的拓扑关键点，本质上它是对割点概念的扩展，使之真正符合无结构对等网络的工作特点和实际需求。基于分点的概念，我们设计了一套简单、有效、分布式的分点检测和消除方法，具体说来，该方法有如下特点：

（1）分布式：不需要任何集中式的控制，每个结点独立地运行自己的检测与消除算法；

（2）低开销：分点检测、消除的通信开销低，并且由分点消除操作导致的新分点数非常少；

（3）抗分割：分点的消除能有效地防止覆盖网被分割，这也是本文研究分点的初衷；

（4）高效率：为消除分点而对网络组织结构所做的调整，优化了覆盖网拓扑，减少了性能瓶颈，从而提升了系统查询成功率；

（5）高容错：去掉覆盖网中分点这一关键而脆弱的元素后，即使面临结点不断失效的高动态性环境，系统查询成功率仍然能保持基本稳定，不会出现急剧下降的情况；

（6）异构性：我们为消除分点而给结点加边、当结点连接度达到上限时给结点减边，都是以异构性的开发作为主要的指导原则，从而提高了系统的负载均衡，让每个成员尽量各尽所能。

4.2 国际研究现状

对等网络的“覆盖网分割”问题及与之相关的容错性问题一直被研究者广泛重视。Saroiu等人通过细致的测量来研究无结构对等网络 Gnutella 的容错性 [SG03]，其实验结果表明：在 Gnutella 中，绝大多数结点连接度很低，少数结点连接度非常高。更具体地说，Gnutella 符合 α=2.3 的幂律（Power-Law）分布，对随机结点失效有高容错性，但是对恶意攻击的抵抗力则非常弱。尤其是高连接度的结点失效后，容易导致覆盖网被分割。虽然 Saroiu 等人认识到高连接度的结点对系统容错性的重要意义，但没有能够进一步回答：到底哪部分高连接度结点属于系统拓扑关键点？哪些结点失效会导致覆盖网分割？我们认为分点是这个问题的答案。

Ripeanu 指出：如果无结构对等网络中每个结点的连接度能尽量高于某个常数下限，就能有效提高对抗恶意攻击的能力 [RG01]。这里所说的“常数下限”实际上就是一幅图 G 的最小结点度 $\delta(G)$，因为图论中有一个简单而重要的不等式：点连通度 $\kappa(G) \leq$ 边连通度 $\lambda(G) \leq$ 最小结点度 $\delta(G)$，所以最小结点度实际上确定了点连通度和边连通度的上限，也就间接确定了图的连通性上限。本文为消除分点而选择合适的结点加边，选择所依据的第一条原则正基于此，让最小结点度 $\delta(G)$尽可能大。

Liu等人 [LX06] 最先从割点的角度研究无结构覆盖网的分割问题。他们设计了一种称为CAM（Connection Adjacency Matrix）的分布式算法来检测割点。CAM算法有效地检测到结点间的可达关系，消除了重复的探测工作，将通信开销降得很低。然而CAM有一些缺点。首先，CAM算法无法准确检测割点，除非消息路由的跳数没有限制。实际上，CAM所检测到的“割点”并非割点，而是我们提出的“分点”。其次，为消除割点，CAM算法从

每个连通分量中随机选取一个结点（我们称该结点为其所在连通分量的“代表结点”）将它们相连，然而我们认为：代表结点的选取非常重要，应基于某种策略、从某个角度提高网络性能，所以本文的分点消除遵循了一种兼顾容错性提升、异构性开发的策略来选择代表结点。再次，选择了代表结点后，如何将它们相连，也是重要的。Liu等人举了“线性链连接”的例子：将代表结点线性地串成一条闭合链。线性链连接方式虽然简单，但很脆弱，对网络连通性的提高也不充分。本文对几种不同的连接方式进行了讨论，并通过实验来比较线性链连接和我们提出的“带弦环连接”。最后，任何一个网络中，结点的连接度都不可能是无穷的，既然为了消除分点而给结点加边，就应该考虑结点度超过上限时给它减边，但 [LX06] 没有提及这个问题。本文考虑了减边的原则和方法，并在模拟实验中做了实现。

对等网络中存在着普遍的结点异构性：结点间虽然在功能上对等，在实际能力（带宽、存储容量、计算能力等）上却有巨大的差异。流行的对等网络 KaZaA、eDonkey 通过显式地设置“超结点”来开发这种异构性，而 Gia 系统 [CR03] 则通过隐式的自适应调整以形成“集群效应”来开发异构性。不管显式隐式，这些方法的指导思想都在于让结点各尽所能－－能力越高、负担就越大。本文中，我们为消除分点而给结点加边、当结点连接度达到上限时给结点减边，都是以异构性的开发作为主要原则。

虽然本文研究的是无结构对等网络中的分点，但其中大多数概念、方法对结构化对等网络同样适用，所以我们也关注了结构化网络的相关研究。Liben-Nowell 等人 [LB01] 分析了 Chord [SM01] 在动态网络环境下的容错性，并给出了维护 Chord 覆盖网连通性的邻居信息获取速率下限。Saia 等人 [SF02] 构建了一个动态高容错的覆盖网，通过让每个结点维护 $O(\log^3 N)$ 项路由信息以及每次路由需要发送 $O(\log^3 N)$ 条消息，使得在任何时刻，绝大多数结点都能成功获取绝大多数数据对象。Loguinov 等人 [LC05] 研究了一些结构化 P2P 覆盖网（Chord、CAN [RF01]、de Bruijn 图）的图论属性：路由性能、集群属性、等分带宽等，并提出了一种基于最优直径 de Bruijn 图的网络架构。

对于已经提出的解决覆盖网分割问题的各种方法，我们从其执行时间与覆盖网分割的先后顺序的角度将它们分成两类：“主动消除”与“被动修复”。“主动消除”意味着采取措施主动消除可能出现的覆盖网分割、以尽可能保证覆盖网一直连通，“被动修复”意味着当发现覆盖网分割时被动地修复以合并分裂的多个子网。

（1）主动消除

Pandurangan 等人 [PR01] 提出了一种集中式的方法来维护 P2P 覆盖网的连通性，他们使用集中式的服务器来引导新加入覆盖网的结点和哪些现存结点建立连接、离开覆盖网的结点如何更新信息，前提是要求所有结点的加入和离开都必须预先通知服务器。很明显，这违反了 P2P 的思想。上文讲到的 CAM 算法和本文提出的解决覆盖网分割问题的方法，通过周期性地检测无结构覆盖网中的拓扑关键点、然后将它们调整成普通结点，来尽量消除拓扑关键点的存在，从而缓解无结构 P2P 覆盖网的分割问题。

（2）被动修复

很多结构化 P2P 网络的工作是基于环结构的，如果大环被分割成多个独立的小环（简称为“分环”），就不能正常工作。Mahajan 等人 [MC03] 设计了一种能有效检测和修复分环的方法，并对 Pastry [RD01] 做了具体的实现。Harvey 等人 [HJ03] 专门针对 SkipNet [HD03] 的分环问题做了研究，但其设计的方法只适用于 SkipNet 这样的提供数据语义、消息路由双重局部性的特殊网络。Sit 和 Morris [SM02] 提出了一种“交叉验证”路由表的方法来缓解覆盖网分割问题：每个结点每隔一段时间随机地向其路由表中的邻居发送校验查询消息，让其邻居执行此查询，然后比较自己查询的结果和邻居返回的结果，如果两者一致，可以近似地认为网络没有被分割；否则，就需要启动相应的修复机制。经典的结构化 P2P 网络 Chord 和 Pastry 中都有谈及覆盖网分割问题。在 Chord 中，一个结点 N 周期性地要求其它结点查

询自己，如果其它结点不能正确地找到 N，就认为可能存在网络分割。而在 Pastry 中，结点周期性地使用 IP 多播，以扩展环路由的方式来搜索邻近的结点，如果存在网络分割，那么被分离开的结点之间很有可能互相搜索到，那时再来做子网合并。

4.3 底层覆盖网的拓扑优化算法

4.3.1 分点的概念

首先定义无结构对等网络中的两种关系：“定位关系”和“可达关系”，然后基于上述关系定义分点。

定义1（**定位关系**）：在无结构对等网络中，如果结点A通过发送路由消息能找到结点B，则称A能定位B，记作A→B。

定位关系是不传递的，一般也不对称（除非网络可看成无向图）。图4.3是定位关系的例子。

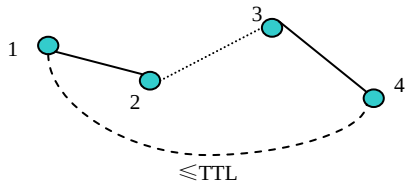


图4.3 结点1能定位结点2、3、4

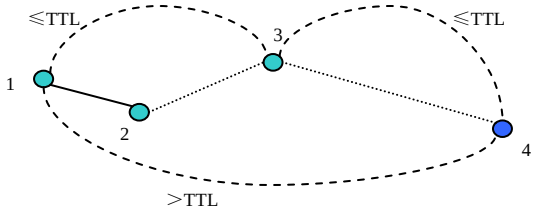


图4.4 结点1能定位2、3，但不能定位4；而结点3能定位4；故结点1可达4

定义2（**可达关系**）：在无结构对等网络中，如果结点A能定位到结点B，结点B能定位到结点C，则称A到C是可达的，记作A→→C。

可达关系是传递的，如果P→→Q，Q→→R，那么P→→R；可达关系一般是不对称的（除非网络可以看成无向图）。图4.4是可达关系的例子。可达关系是检测分点的基础：一个结点n是否为分点，取决于n被删去后其邻居间是否仍然都可达。

定义3（**分点**）：在无结构对等网络中，如果删去结点C后，C的邻居集将被分成两个或多个互不可达的子集：$S_1, S_2, \cdots, S_n$（n>1），就称C为“分点”。

任意两个子集间的结点都不可达，而每个子集内的结点是可达的。图4.5是分点的例子，每个子集内部的边和网络中其它结点均未画出。

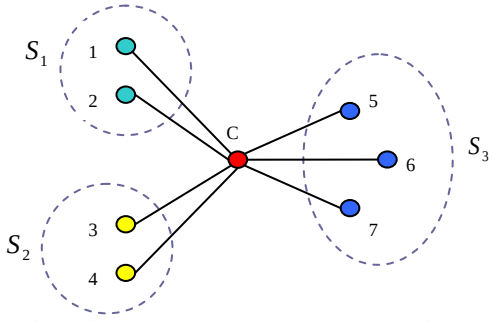


图4.5 分点C的邻居集被分成三个互不可达的子集S_1、S_2、S_3

从上述定义可以看出，分点既是其邻居子集间互相连接的唯一通道，也是邻居子集间互相通信的性能瓶颈，所以分点是无结构对等网络中真正的拓扑关键点。因此，消除了分点这一脆弱而关键的元素，就能够从本质上提高系统对覆盖网分割的抵抗力、有效地提升系统容错性。

我们对无结构对等网络中分点的定义还有如下特点：

（1）分布式：分点可以分布式地进行有效检测和消除，检测、消除所需的信息交换只限于邻居结点之间；

（2）局部性：分点符合无结构对等网络路由局部性的工作特点，因为所用到的的定位关系、可达关系都基于局部性的路由，比起割点更具实际意义；

（3）兼容性：兼容割点，因为分点包含了割点。如果对TTL没有限制，那么分点实际上就是割点。

4.3.2 分点的检测

基于前一部分对分点的定义，分点检测的关键，就是在候选分点 C 不参与消息路由的前提下，探测其邻居间的可达关系，再依据可达关系将邻居集分成互不可达的子集；若子集只有一个，那么 C 不是分点，否则，C 是分点。这就是我们设计的分点检测方法的思想。具体说来， C 为了检测自己是否为分点，分几步进行：

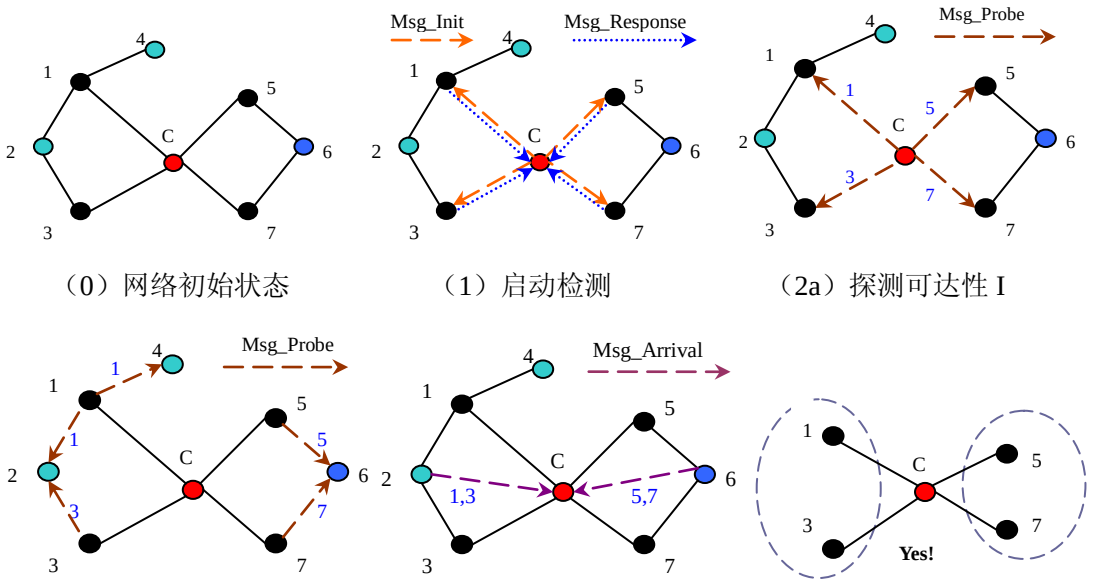
(1) **启动检测**：C 向每个邻居 $N_1, N_2, \cdots, N_n$ 发送消息 Msg_Init 以启动检测，收到消息的邻居必须回复 Msg_Response，其中包含它的 IP 地址、带宽、时延、连接度等信息（见图 4.6（1）），这些信息是以后的工作（分点消除）所必需的。如果 C 没有收到邻居 N_i 的回复，就认为 N_i 失效，将其从路由表中删除。

(2) **探测可达性**：C 给每个邻居发送探测消息 Msg_Probe（见图 4.6（2a）），此消息中包含结点 C 的地址、跳数限制 TTL 和邻居号 N_i。网络中每个结点维护一张“连接表”，每个候选分点在连接表中都有其对应的一项，形如< 候选分点地址，邻居号 1，邻居号 2，… >。结点 N 的连接表中候选分点 C 所对应的项表明 C 的哪些邻居是互相可达的。当某个结点收到 Msg_Probe 消息时，如果 Msg_Probe 中的邻居号是新的，它就把该消息发送给每个邻居（除了消息发送者）（见图 4.6（2b）），并且将消息中的邻居号 N_i 添加到候选分点 C 的连接表项中。如果连接表项的邻居号不止 1 个，就给候选分点发送 Msg_Arrival 消息，此消息包含该表项中所有的邻居号，实际上就是告诉候选分点哪些邻居可达（见图 4.6（3））。候选分点给其不同邻居发出的探测消息所到达的范围是不重叠的（只在相交的那一点回发 Msg_Arrival 消息报告邻居间的可达关系），消除了重复的可达性探测。上述探测过程借鉴了 CAM [10]探测割点的思想， [10]中有更为详细的介绍和理论分析。

(3) **划分子集**：C 不断收集 Msg_Arrival 消息，计算其邻居之间的可达关系，将可达的结点划分到同一个子集（这一步实际上是在计算等价类）。鉴于对等网络的动态性，少数 Msg_Arrival 信息可能会丢失，所以 C 可以设置一个超时值 Timeout，在 Timeout 后认为得到了最终的划分。

(4) **判定分点**：完成子集的划分后，C 根据子集的数目来判定自己是否为分点：如果所有邻居归到同一个子集，那么 C 不是分点；否则，C 是分点，进入下一步——分点的消除。

图 4.6 的 6 幅图以简单的例子形象地表现了分点检测的过程：



（0）网络初始状态　（1）启动检测　（2a）探测可达性 I

（2b）探测可达性 II　（3）收集可达关系　（4）划分子集、判定分点

图 4.6 分点检测的过程

4.3.3 分点的消除

a) 加边以消除分点

当结点 C 判定自己是分点后，为消除成为分点，C 需要给其邻居加边以合并互不可达的多个子集。假设 C 的邻居被分成多个子集：$S_1, S_2, \cdots, S_n$，那么 C 从每个子集 S_i 中找到最“适合”加边的代表结点 N_i，然后以某种方式将所有代表结点相连。为了找到最“适合”加边的代表结点，我们考虑如下原则：

（1）使每个结点的连接度尽量高于某个常数下限（也就是尽可能提高对等网络的最小结点度），从而提高系统容错性；

（2）使每个结点尽量各尽所能，也就是使负载因子（＝结点度 / 结点能力）尽量趋于一致，从而开发对等网络中普遍存在的结点异构性。

基于上述原则，分点 C 首先找到子集 S_i 中连接度最低的结点，看其结点度是否低于常数下限 *Min_Degree*，如果是则将此结点作为 S_i 的代表结点 N_i；否则，计算每个结点的负载因子，将负载因子最小的结点（也就是负担最轻的结点）作为代表结点 N_i。

选择好代表结点后，代表结点之间以何种方式相连？最简单的连接方式是“线性链连接”：给每个 N_i 加一条到 N_{i+1} 的边（给 N_n 加一条到 N_1 的边），这样所有的代表结点就形成了一条闭合链，从而合并了互不可达的多个子集，消除了分点的存在。图 4.7 是线性链连接的例子。

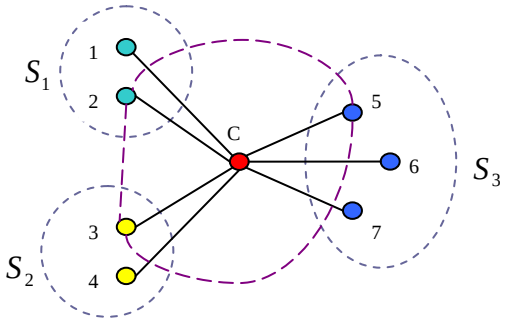


图 4.7 “线性链连接”各代表结点

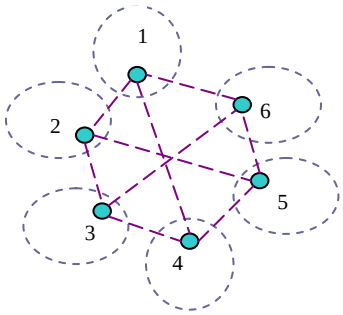


图 4.8 “带弦环连接”各代表结点

以线性链方式连接代表结点虽然简单，但很脆弱，以后如果有结点失效，这条链很可能断裂，从而又形成新的分点。因此，我们考虑稍复杂的连接方式，比如对代表结点 N_1、N_2、…、N_n，让每个 N_i 不仅连接 $N_{(i+1) \bmod n}$，还连接 $N_{(i+n/2) \bmod n}$，也就是首先连成环，然后在环上给每个代表结点加一条到对面的弦（见图 4.8）。模拟实验部分比较了带弦环连接和线性链连接的性能，测量结果表明前者通常要优于后者。

如果给每个结点再加一些边，比如像 Chord 那样，给每个结点 i 加一系列指数距离的边 $i+1$、$i+2^1$、$i+2^2$、$i+2^3$、…，那么连通性的提升自然更多；但是这带来了更多额外的开销，实现起来也更为复杂。上面所讲的带弦环连接方式，简单实用，也不会带来很多额外开销，我们认为它是介于线性链连接方式和 Chord 所采取的连接方式之间的一个合理折衷。

b) 减边以限制邻居数

每个结点所可以维护的连接数（结点度）是有限制的。连接度越高，更新路由表的自适应开销就越大，因此每个结点只能根据自己的能力（带宽、存储容量、计算能力等）维持一定数目的连接。所以，当加边操作导致结点连接度超过其上限时，该结点需要减边以限制邻居数。减边遵循如下原则：

（1）为消除分点而新加的那些边不能减；

（2）对于其它边而言，计算每条边所连接的结点的负载因子，将负载因子最大的结点（也就是负担最重的结点）对应的那条边删去。

上述两条原则的理由是明显的，而实现方法也很直观，不再累述。需要注意的是：原则（1）优先于原则（2），否则可能产生覆盖网因减边再次分割的问题。

c) 分点检测和消除的总开销

分点检测的过程中，候选分点给其不同邻居发出的探测消息所到达的范围是不重叠的（只在相交的那一点回发 Msg_Arrival 消息报告邻居间的可达关系），消除了重复的可达性探测。假设无结构对等网络中共有 n 个结点，c 为平均结点度，t 为跳数限制 TTL，那么每个候选分点检测的开销为 $\min(O(c^t), O(nc))$，所以检测的总开销（即所需消息数）为 $\min(O(nc^t), O(n^2c))$。

如果某个结点确认自己是分点，假设其邻居被分成 k 个子集，那么不管采用线性链连接还是带弦环连接，加边的开销都为 $O(k)$。由于 k 必定小于等于分点的度 d，所以加边开销也可写为 $O(d)$。考虑最坏情况，每个结点都是分点，此时加边的总开销为 $O(nc)$。同理，减边的总开销也为 $O(nc)$。因此，分点消除的总开销为 $O(nc)$，它相比分点检测的总开销 $\min(O(nc^t), O(n^2c))$而言要少得多，可见分点检测和消除算法的开销基本上都在检测这一步。综上所述，分点检测和消除的总开销为 $\min(O(nc^t), O(n^2c))$。特别地，当平均结点度 c 和跳

数限制 t 都较小时，总开销即为 *O(n)*，*O(n)*的线性开销一般认为是高效的。

4.3.4 性能评价

a) 实验方法和参数

实验是基于模拟的，随机产生一张连通的无向图作为覆盖网拓扑的基础，每个结点连接到其它哪些结点是随机的，但平均结点度有所规定（平均结点度=6）。模拟的网络结点总数 *N*=1000，跳数限制 *TTL* 取 2～5 不等。在 5.1 中我们讲过，给结点加边的第一条原则，是要使每个结点的连接度尽量高于某个常数下限，实验中我们设定结点度下限 *Min_Degree*=3（平均结点度的一半）。给每个结点随机分配其能力值（1～20 不等），代表它可以负担的连接度。为了模拟高动态性的网络环境，我们让网络中的结点不断失效，而每隔 *T* 个结点失效后做一次分点检测和消除以测量其效果。实验中分别取 *T*=5、10、20、50，*T* 越小，分点检测和消除做的越频繁，通常效果就越好，当然开销也相应越大。

b) 实验过程与测量结果

实验一：分点对覆盖网拓扑的意义

这个实验的目的在于证实分点在覆盖网中的拓扑关键性。对相同的初始网络，首先让随机结点逐个失效，记录多少个结点失效后覆盖网被分割；然后让分点逐个失效，记录多少个分点失效后覆盖网被分割。*TTL* 取 3、4，多次实验取平均值。

从图 4.9 观察到：对同样的覆盖网，如果要让它分割，失效的分点数要远少于失效的随机结点数。覆盖网被分割时失效的随机结点数在 1－150 之间，而失效的分点数，*TTL*=3 时在 1～35 之间，*TTL*=4 时在 1～2 之间（这说明 *TTL*=4 时检测到的大多数分点也是割点）。由此可以看出分点在覆盖网中的拓扑关键性：只要让极少数的分点失效，覆盖网就被分割。

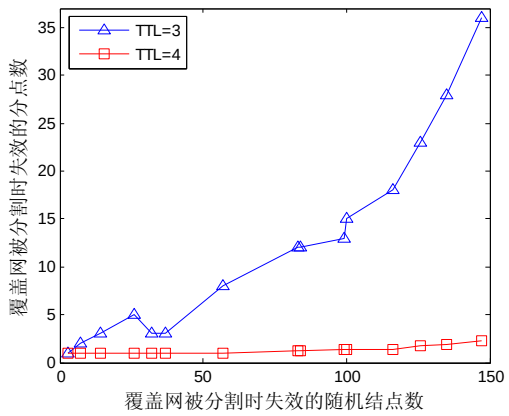


图 4.9 分点对覆盖网拓扑的意义

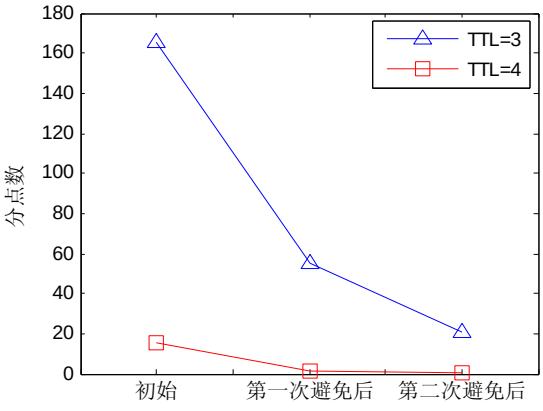


图 4.10 分点消除算法的有效性

实验二：分点消除算法的有效性

为了测量分点检测和消除算法的有效性，我们首先检测网络初始的分点数，然后做一次分点消除并检测消除后的分点数，最后再做一次分点消除并检测第二次消除后的分点数，以观察分点数的变化趋势。测量结果都是多次实验的平均值，*TTL* 取 2、3、4、5。

右下边的测量结果和图 4.10 说明，每一次分点消除后，网络分点数均有明显下降，尤其当 *TTL*=2 时最为突出：初始时 1000 个结点中几乎所有结点都是分点，第一次消除后减少

到 50 附近，第二次消除后减少到 20 附近。初始时几乎所有结点都是分点，原因在于 *TTL*=2 很小，只探测到极其有限的可达性关系，所以结点的邻居集基本上被划分到多个子集中，从而几乎每个结点都认为自己是分点。*TTL*=3 时，初始分点数在 160 左右，第一次消除后降到 55 左右，是初始的约 1/3，第二次消除后降到 20 左右，又是第一次消除后的约 1/3。*TTL*=4 时，初始分点数稳定在 15 左右，第一次消除后降到 1，第二次消除后接近 0。由于 *TTL*=2 时初始分点数太多，没有将它画在图 4.10 中，*TTL*=5 时的实验结果与 *TTL*=4 时非常相似，所以也没有画出。总体上讲，*TTL* 越大，探测到的可达性关系就越充分，分点数相应越少；但不管 *TTL* 取多少，算法本身的有效性是非常明显的。

分析第一次分点消除后为什么还存在分点，我们认为原因在于分点消除时所做的加边、减边操作改变了覆盖网拓扑结构，从而带来了新的分点。不过新的分点数通常都是很少的，并会慢慢趋于 0，网络组织结构也相应趋于稳定。

分点数	初始	第一次避免后	第二次避免后
TTL=2	981	55.6	21.6
TTL=3	165.8	55.4	20.8
TTL=4	15.4	1	0.4
TTL=5	10.6	0.8	0.2

实验三：系统对覆盖网分割的抵抗力增强

为了观察分点消除后系统对覆盖网分割的抵抗力增强，我们首先让网络结点逐个失效并且不采取任何补救措施，测得 58 个结点失效后覆盖网被分割。同样的网络，让同样的结点逐个失效，但每 *T* 个结点失效后做一次分点消除，记录多少个结点失效后覆盖网被分割。实验中，*T*=5、10、20、50 表示分点消除算法执行的频率高低。

右边的测量结果和图 4.11 表明：*TTL*=3 的情况下，当 *T*＝5 时，超过 900 个结点失效，覆盖网才被分割，这说明分点消除做得及时、有效，使覆盖网一直保持很好的连通性；当 *T*=10、20、50 时，分别有约 1/3、1/4、1/10 的结点失效时，覆盖网被分割。*TTL*=4 的情况下，当 *T*=5、10、20、50 时，分别有约 1/3、1/5、1/6、1/10 的结点失效时，覆盖网被分割。

失效结点数	*T*=5	*T*=10	*T*=20	*T*=50
TTL=3	58 → 926	58 → 341.6	58 → 237.4	58 → 117
TTL=4	58 → 357	58 → 218.2	58 → 170	58 → 110

随着 *T* 变大，分点消除执行的频率变低，网络拓扑结构得不到及时、有效的修复，覆盖网分割时的失效结点数也随之减少，说明系统对覆盖网分割的抵抗力变弱了。而当 *TTL* 从 3 变大到 4 后，由于每次检测到的分点数变少（从实验二可以看出），对网络结构的调整也相应变少，所以对覆盖网分割的抵抗力也变弱了。总体上讲，*T* 越小、*TTL* 越小，系统对覆盖网分割的抵抗力就越强，当然通信开销也相应增大。

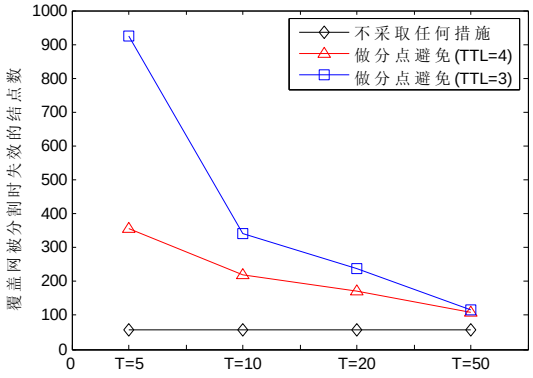


图 4.11 系统对覆盖网分割的抵抗力增强

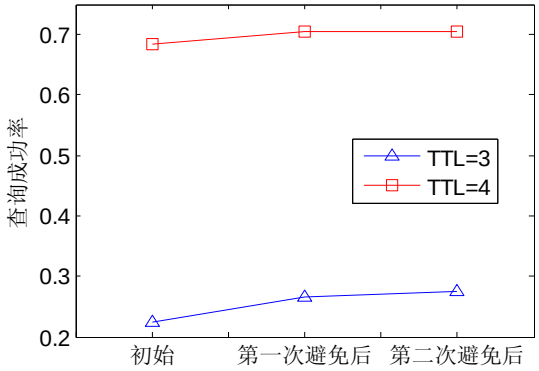


图 4.12 系统查询成功率的提升

实验四：系统查询成功率提升

我们首先测量初始网络的查询成功率（查询对象随机指定），然后进行一次分点消除，并且测量查询成功率（同一结点每次都做同样的查询），最后进行第二次分点消除，再次测量查询成功率。注意网络环境是静态的，不存在结点失效的情况，查询的跳数限制 *TTL* 与分点检测的 *TTL* 是一样的。

查询成功率	初始网络	第一次消除后	第二次消除后
TTL=3	0.225	0.267	0.274
TTL=4	0.684	0.706	0.706

从右边的测量结果和图 4.12 可以看到：无论 *TTL* 取 3、4，第一次分点消除后的查询成功率相对初始网络都有所提高，TTL 越小提高越多。联系实验二来分析其原因，*TTL* 越小，检测到的分点就越多，对网络所做的调整也越多。这同时也表明：分点消除对网络结构的调整是有利于系统查询成功率的，是对覆盖网拓扑的一种优化。第二次分点消除对查询成功率的提升很小甚至没有，因为第一次分点消除带来的新分点数非常少。

实验五：系统容错性提升

我们将系统容错性理解为结点不断失效的高动态性环境下系统的查询成功率变化趋势，希望采取了周期性的分点消除后，查询成功率总体上能比较稳定。首先让网络结点逐个失效但不采取任何补救措施，每 *T* 个结点失效后测量一次查询成功率（同一结点每次都做同样的查询）。然后，对同样的网络，让同样的结点失效，但每 T 个结点失效后做一次分点消除，同时测量系统查询成功率。

取 *TTL*=3、4，T=10、50，因此就有 2*2=4 种组合，需要做 4 次实验。查询的跳数限制 *TTL* 与分点检测的 *TTL* 是一样的。图 4.13 显示了 *TTL*=3 时的实验结果：随着结点不断失效，做分点消除相对于不采取任何补救措施而言，查询成功率要更高、更稳定，并且不会出现急剧的下降。*TTL* 越大、*T* 越大，容错性的提升就越不明显，前者是因为 *TTL* 越大，检测到的分点越少，所以对覆盖网的修复也越少；后者是因为 *T* 越大，对覆盖网的修复越不及时，所以容错性相应降低了。

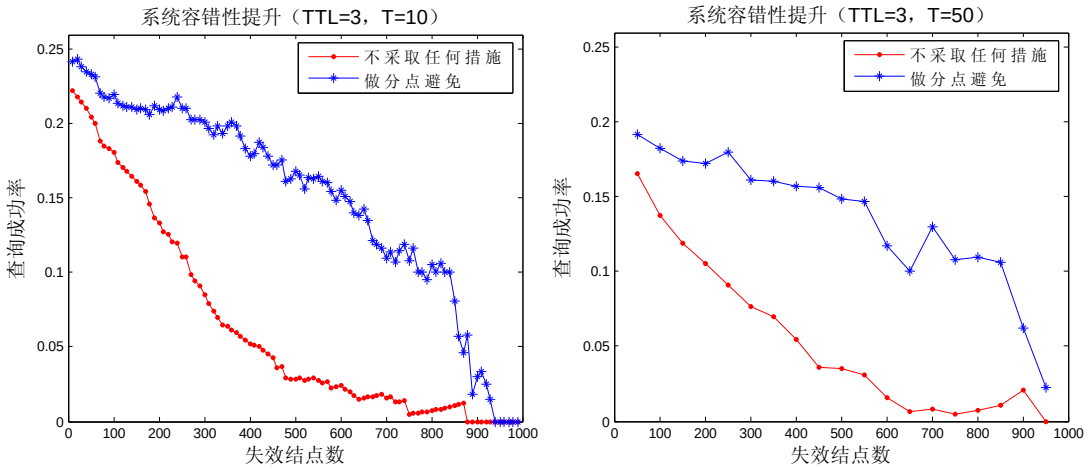
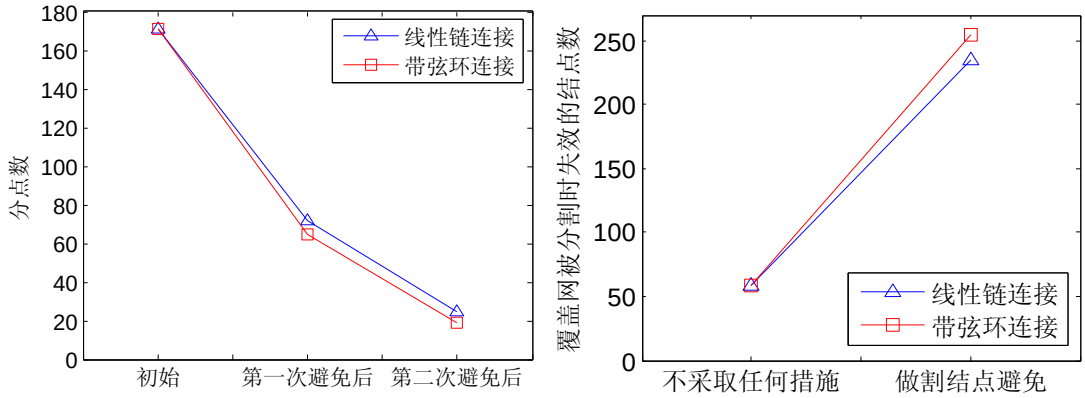
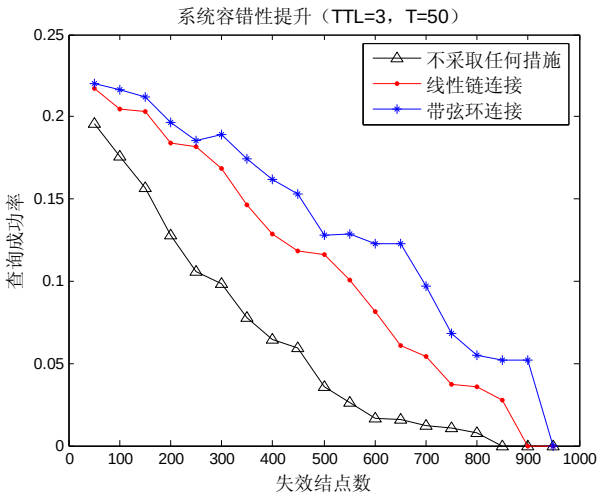


图 4.13 系统容错性的提升（*TTL*＝3）



（a）算法本身的有效性比较　　（b）系统对覆盖网分割抵抗力的增强



（c）线性链连接和带弦环连接的容错性比较

图 4.14 “线性链连接”和“带弦环连接”的各方面性能比较

实验六：“线性链连接”和“带弦环连接”的各方面性能比较

分点消除的重要步骤是给互不可达的各个子集的代表结点加边将它们连接，但是连接方

式孰好孰坏是个问题。实验中我们在其它条件（*TTL*、*T*、初始的网络拓扑结构、失效结点）相同的情况下，分别采用“线性链连接”和“带弦环连接”两种不同的连接方式，比较它们的多方面性能：（1）分点消除算法的有效性（见图 4.14（a））、（2）系统对覆盖网分割的抵抗力增强（见图 4.14（b））、（3）系统容错性提升（见图 4.14（c））。

（1）分点消除算法的有效性。取 *TTL*=3。

TTL=3	初始分点数	第一次消除后	第二次消除后
线性链连接	171	72	25
带弦环连接	171	65	19

（2）系统对覆盖网分割抵抗力的增强。取 *TTL*=3、*T*=50。

TTL=3，*T*=50	覆盖网被分割时失效的结点数
线性链加边	58 → 235
带弦环加边	58 → 254

从上面的测量结果和图 4.14(a)(b)(c)可以看出：不论就哪个方面性能而言，“带弦环连接”通常都比“线性链连接”有所提升。原因是很直观的：“带弦环连接”能将同样的代表结点连接得更好、连通性更高，而“线性链连接”形成的单环相对脆弱，很可能因为以后的结点失效而分裂。

实验七：系统异构性开发

我们为消除分点而给结点加边、当结点连接度达到上限时给结点减边，都是以异构性的开发作为主要的指导原则，从而提高系统的负载均衡，让每个成员尽量各尽所能。为了更具体地量化系统异构性的开发，我们测量一个实验网络从开始到稳定期的结点负载因子分布，包括：最小负载因子、最大负载因子、平均负载因子、负载因子方差。取 *TTL*＝4，加边采用“带弦环连接”，结点能力值取 1～20 之间的随机数，平均结点度为 6，最小结点度为 3。

	最小负载因子	最大负载因子	平均负载因子	负载因子方差
初始网络	0.05	1.0	0.619	0.341
稳定期	0.05	1.0	0.635	0.227

从右边的实验结果我们可以看出，处于稳定期的网络与初始网络具有相同的最小、最大负载因子，平均负载因子略大，而负载因子方差则有明显减少。这与我们预期的结果基本相符。由于实验规模为 1000 个结点，出现能力值为 20 而连接度仅为 1 的结点（即最小负载因子的结点）的概率是较高的，即使在稳定期仍然如此。最大负载因子 1.0 的出现几乎是必然的，因为很多结点的连接度可以达到其能力上限。由于消除分点时加边情形的发生概率要高于减边，所以不难理解稳定期的平均负载因子要略大于初始网络。最后，由于无论加边、减边时我们都考虑到了负载均衡，所以稳定期的负载因子方差要明显小于初始网络。

4.4 小结与进一步的工作

本文的工作主要是对P2P流媒体系统的底层覆盖网（即无结构对等网络）所做的拓扑优化。首先指出，对松散化组织、局部性路由的无结构对等网络而言，静态的、全局意义的割点既难以准确检测，也远非系统拓扑关键点的全部，所以我们提出了“分点”这一概念来描述无结构对等网络的拓扑关键点。然后从分点的定义出发，设计了一套简单、有效、分布式的分点检测和消除方法来优化覆盖网拓扑结构。模拟实验的结果表明：我们的方法能有效地

检测并消除分点，使系统对覆盖网分割的抵抗力得到本质的增强；分点消除后，系统查询成功率有所上升，在高动态性网络环境下系统的容错性得到明显提高。在做分点消除时，我们对代表结点的选取做了仔细的考虑，兼顾了容错性和异构性；对代表结点的连接，则采取了“线性链连接”和“带弦环连接”两种方式，实验证明后者在性能上通常要更优。

下一步的工作将把分点的概念、检测、消除应用到结构化对等网络（如Chord、Pastry 等）中，以增强结构化对等网络对覆盖网分割的抵抗力和提高其容错性。鉴于结构化对等网络和无结构对等网络在组织结构、路由方式等多方面的不同，本文的许多地方需要做相应的修改，尤其是结构化对等网络的覆盖网不能看作无向图。虽然如此，分点作为对等网络的拓扑关键点，对优化其覆盖网拓扑结构有重要意义。

第5章 结束语

P2P流媒体系统在过去几年里快速发展并应用广泛。根据最新发布的中国互联网发展状况统计报告，中国内地网民观看网络视频的几种主要方式中，通过P2P流媒体软件下载的比率达到29.91%，接近三分之一，可见P2P流媒体是一个拥有巨大用户群、充满研究潜力与意义的领域。

P2P流媒体系统值得研究的属性有很多，本文对P2P流媒体系统的3个关键属性进行了研究：（1）播放连续度、（2）源切换时延、（3）系统容错性。研究工作概括如下：

在第2章，设计了一个具有高播放连续度的P2P流媒体系统ContinuStreaming，采用基于分布式散列表的数据预取方法来弥补Gossip协议传播数据的缺陷，从而保证流媒体系统能保持高播放连续度。对ContinuStreaming系统的播放连续度进行了理论分析并将理论分析的结果与模拟实验的结果进行了比较。在多幅真实P2P网络拓扑上所做的大量模拟实验结果表明：相比于当前具有代表性的基于Gossip协议的P2P流媒体系统CoolStreaming而言，ContinuStreaming系统能以低于4%的额外开销带来接近1.0的高播放连续度。

在第3章，通过对本质特性与关键参数的分析给P2P流媒体系统中数据发布源切换过程建立了数学模型，从而将源切换问题形式化为一个数学优化问题，然后推导出此数学优化问题的最优解。鉴于实际系统情况的复杂性与网络环境的动态性，提出了一个称为快速源切换算法的实用贪心算法，它通过交错旧源与新源的数据传递来趋近理论上的最优解。在多个真实测量的P2P覆盖网拓扑上做的大量模拟实验的结果显示：快速源切换算法相比传统源切换算法能节省20%-30%的切换时间，同时不会带来额外的通信开销，并且切换时间的减少比例随着系统规模的增大而趋于增加。

在第4章，提出分点这一概念来描述P2P流媒体系统的底层覆盖网、即无结构对等网络的拓扑关键点，然后基于分点的概念，设计了一套简单、有效、分布式的分点检测和消除方法来优化覆盖网拓扑结构。模拟实验的结果表明：设计的方法能有效地检测并消除分点，使系统对覆盖网分割的抵抗力得到本质的增强；分点消除后，系统查询成功率有所上升，在高动态性网络环境下系统的容错性得到明显提高。

除了上述三方面工作之外，本文认为，P2P流媒体领域还有很多值得深入研究的问题，例如：

1. P2P流媒体视频点播系统的先驱性研究。到目前为止，P2P流媒体领域主要的工作都集中在对实时流媒体系统的研究，实际上，实时流媒体系统的研究已经趋于成熟，剩余的研究空间不大、研究价值也不高，相反，对P2P流媒体视频点播系统的研究很少，其原因主要在于视频点播的需求要比实时播放高很多，而且从本质特性上来看，P2P技术与实时播放的需求更为契合，与视频点播的需求存在差距。正是因为P2P流媒体视频点播系统的研究有难度、有挑战，所以才有广阔的研究空间与深刻的研究价值，这一点从SIGCOMM’07、SIGCOMM’08两次会议都录用了关于P2P视频点播的论文就可以看出。

2. P2P流媒体系统属性间关系的研究。在本文第1.4节列举了P2P流媒体系统值得研究的多项属性，其中绝大多数属性都能找到对应的研究工作，但很少发现有研究这些属性间关系的工作存在。实际上，P2P流媒体系统的各项属性中，有的互相依赖，有的互相制约，有的互相促进，有的互相矛盾，如能从较深的层次分析和挖掘这些依赖、制约、促进、矛盾的关系，必将取得有价值的研究成果。

参考文献

[BB02] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (SIGCOMM). Pages 205-217, 2002.

[CD02] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-layer multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8): pages 1489-1499, 2002.

[CD03] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. *Proceedings of the 19th ACM Symposium on Operating Systems Principles* (SOSP), pages 298-313, 2003.

[CL90] T. Cormen, C. Leiserson, and R. Rivest. Introduction to algorithms. *MIT Press Cambridge, MA, USA*, 1990.

[CR00] Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan and Hui Zhang. A Case for End System Multicast. *ACM SIGMETRICS Performance Evaluation Review*, Volume 28, pages 1-12, 2001.

[CR03] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. *Proceedings of the ACM SIGCOMM*, pages 407-418, 2003.

[GK03] A. Ganesh, A. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, 2003.

[HD03] N. Harvey, J. Dunagan, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. SkipNet: A Scalable Overlay Network with Practical Locality Properties. *Proceedings of the USENIX Symposium on Internet Technologies and Systems* (USITS), pages 9-9, 2003.

[HF08] Y. Huang, Tom T.J. Fu, D.M. Chiu, John C.S. Lui, and C. Huang. Challenges, Design and Analysis of a Large-scale P2P VoD Systems. *Proceedings of the 2008 conference on Applications, technologies, architectures, and protocols for computer communications* (SIGCOMM), 2008.

[HJ03] N. Harvey N, M. B. Jones, M. Theimer, and A. Wolman. Efficient Recovery From Organizational Disconnects in SkipNet. *Proceedings of the Second International Workshop on Peer-to-Peer Systems* (IPTPS), 2003.

[HL07] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications* (SIGCOMM), pages 133-144, 2007.

[JG00] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole. Overcast: reliable multicasting with on overlay network. *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation* (OSDI), Volume 4, pages 14--14, 2000.

[KM03] A. Kermarrec, L. Massoulie, and A. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, 2003.

[LB01] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. *Proceedings of the twenty-first annual symposium on Principles of distributed computing* (PODC), pages 233-242, 2002.

[LC05] D. Loguinov, J. Casas, and X. Wang. Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience. *IEEE/ACM Transactions on Networking*, Vol. 13, No. 5, October 2005.

[LJ06] X. Liao, H. Jin, Y. Liu, L. Ni, and D. Deng. AnySee: Peer-to-Peer Live Streaming. *Proceedings of the IEEE INFOCOM*, volume 6, 2006.

[LP05] J. Li. PeerStreaming: An On-Demand Peer-to-Peer Media Streaming Solution Based On A Receiver-Driven Streaming Protocol. *Proceedings of the IEEE 7th Workshop on Multimedia Signal Processing*, pages 1–4, 2005.

[LX06] X. Liu, L. Xiao, A. Kreling, and Y. Liu. Optimizing Overlay Topology by Reducing Cut Vertices. *Proceedings of the ACM International Workshop on Network and Operating System Support for Digital Audio and Video* (NOSSDAV), 2006.

[MC03] R. Mahajan, M. Castro, and A. Rowstron. Controlling the Cost of Reliability in Peer-to-Peer Overlays. *Proceedings of the Second International Workshop on Peer-to-Peer Systems* (IPTPS), 2003.

[MM02] P. Maymounkov, and D. Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (IPTPS'02), pages 258-263, 2002.

[MR07] N. Magharei, and R. Rejaie. PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming. *Proceedings of the IEEE INFOCOM*, 2007.

[PR97] C. Plaxton, R. Rajaraman, and A. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed System. *Proceedings of the Symposium of Parallel Algorithms and Architectures* (SPAA), pages 311-320, 1997.

[PR01] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter P2P networks. *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science* (FOCS), pages 492-499, 2001.

[PS01] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: an application level multicast infrastructure. *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems* (USITS), Volume 3, pages 5-5, 2001.

[RD01] A. Rowstron, and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Proceedings of the IFIP/ACM Conference on Distributed Systems Platforms* (Middleware), 11:329–350, 2001.

[RF01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. *Proceedings of the ACM SIGCOMM*, pages 161-172, 2001.

[RG01] M. Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. *Proceedings of the IEEE International Conference on Peer-to-peer Computing* (IEEE P2P), 2001.

[RH01] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. *Proceedings of NGC*, 2001.

[RO03] R. Rejaie, and A. Ortega. PALS: peer-to-peer adaptive layered streaming. *Proceedings of the 13th ACM International Workshop on Network and Operating System Support for Digital Audio and Video* (NOSSDAV), pages 153–161, 2003.

[SF02] J. Saia, A. Fiat, S. Gribble, A. Karlin, and S. Saroiu. Dynamically fault-tolerant content addressable networks. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (IPTPS), 2002.

[SG03] S. Saroiu, P. Gummadi, and S. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Systems* 9: 170-184 (2003).

[SM01] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proceedings of the ACM SIGCOMM*, 31(4):149–160, 2001.

[SM02] E. Sit, and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. *Proceedings of the 1st International Workshop on Peer-to-Peer Systems* (IPTPS), 2002.

[TH03] D. Tran, K. Hua, and T. Do. ZIGZAG: an efficient peer-to-peer scheme for media streaming. *Proceedings of the IEEE INFOCOM*, 2003.

[XH02] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. *Proceedings of the 22nd International Conference on Distributed Computing Systems* (ICDCS), pages 363–371, 2002.

[ZK01] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. *Computer*, volume 74, 2001.

[ZL05a] M. Zhang, J. Luo, L. Zhao, and S. Yang. A peer-to-peer network for live media streaming using a push-pull approach. *Proceedings of the 13th ACM international conference on Multimedia*, pages 287–290, 2005.

[ZL05b] X. Zhang, J. Liu, B. Li, and T. Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. *Proceedings of the IEEE INFOCOM*, 2005.

[ZZ01] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. *Proceedings of the 11th International workshop on Network and Operating Systems Support for Digital Audio and Video* (NOSSDAV), pages 11-20, 2001.

致谢

本文对我攻读硕士学位期间所做的主要研究工作进行了总结。衷心感谢我的导师陈贵海教授，是他引领我进入P2P领域，指导我做研究的态度、方法和方向。在我做本科毕业论文、攻读硕士学位的3年多时间里，陈教授为我提供了国内领先的学习条件、科研环境、研究氛围和进修机会，此外，他在生活上也给予了我很多的帮助、指导和支持。所有的这一切我将终身难忘。同时，导师勤勉负责的做事态度和谦逊宽厚的为人永远值得我敬仰和学习。

感谢香港理工大学计算机系的曹建农教授。在我的导师陈贵海教授的推荐和帮助下，我获得了硕士阶段赴港进修的宝贵机会，在香港理工大学任助理研究员的5个月中，陈教授和曹教授对我寄予了殷切的希望，给予了有价值的指导，虽然研究压力很大、生活异常辛苦，但收获极多。本文第2章的工作全部在香港完成，最终发表在分布式系统领域著名会议IPDPS'08上，第3章的工作主要在香港完成，最终发表在并行计算领域著名会议ICPP'08上。同时，在香港的助研经历教会了我应该如何向别人表达自己、如何做PPT、如何做Presentation、如何选择值得研究的课题、以及如何判断值得深入的想法。此外，香港的生活丰富了我的人生阅历，成为我人生宝贵的财富。

感谢香港科技大学计算机系的倪明选与刘云浩教授，国防科技大学的肖侬教授，以及华中科技大学的金海教授，他们在百忙之中审阅我的书稿，并且热心担当书稿的审稿人和推荐人，使得《对等网络：结构、应用与设计》的出版最终得以成功。关于这本书的出版，最应该感谢的，仍然是我的导师陈贵海教授，没有他的帮助，这本书到现在仍然会是草稿。

感谢南京大学计算机系的路通副教授，也是我的表哥，给予我方方面面的关心与帮助。感谢同寝室的袁瑞峰、薛永岭、张博、颜异同学，他们给了我真诚的友谊与真实的欢乐。感谢南京大学计算机系的高海燕老师和贾师傅、南京大学研究生院的卞静老师，她们在我完成大量事务性、手续性工作的过程中提供了便利。感谢GPS研究组的杨盘隆、张希伟、吴小兵、曹晓梅、陈跃泉、李成法、邱彤庆、叶懋、刘之育、羊锴等师兄师姐，同届的袁瑞峰、李宏兴、成宁宁、陈欢，他们都曾给予我经验、指导或帮助；特别感谢谢军峰师弟，我们合作了两篇论文，是研究上的伙伴、生活中的朋友；此外还要感谢于南南、陈锦铭、严允陪、展安东、李韬、徐立杰、沈世卿、罗迪军、周伟、王世光、朱小军、苑靖等师弟师妹，和他们在一起的时光是快乐的。

感谢南瑞继保公司和东方海外集团，他们先后给我提供了4000元和5000元的奖学金，给予我硕士阶段的拮据生活一定程度的改善。

深深感谢我的父母在我漫长的求学生涯中给予我的关心与帮助，在下岗以后多年的艰苦人生中，我是他们唯一的希望，虽然他们自己省吃俭用，但从来不会为我吝啬一分钱；而我有生之年从未给过他们实质性的回报，是我一直以来最深的歉疚。深深感谢我的未婚妻夏洁媛对我的理解、支持和爱，她对我痴迷事业的容忍、追求理想的肯定、落魄时光的鼓励、贫困状况的坦然，将成为我一生的美丽回忆。

一个多月以后，我将离开我生活7年的南京大学，离开我生活3年多的南京大学计算机系

GPS研究组，我对这里的人、物、景、事怀有深厚的感情和深切的眷恋。由于个人、家庭、地域等多方面的原因，我最终选择了进入计算机工业界继续自己的下一步人生，暂时告别了我曾经深深痴迷和热爱的学术研究领域。正如我身边很多人所体会到的，我自身对于这一选择也是充满权衡、突然、惋惜、遗憾的复杂心态，是我经历过长期的矛盾、斗争、思考与痛苦之后做出的决定。“客观事实存在着，我们无法改变。但我们对客观事实的态度，却永远有着无限自由。而一旦表明态度、做出决定，就必须承担为之应负的责任。”我已经做出了选择，就只能勇敢地迈出下面这一步。然而我相信有朝一日，我仍然会回到学术研究领域，完成我未尽的事业，发挥我未尽的潜力，因为在这里留下了我的不甘与不舍、不忍与不愿，这里有一些东西，会成为对我永远的呼唤。

最后，向参加本文审稿和答辩的各位老师一并致以最诚挚的谢意！

附录1 个人简历与硕士期间参加科研项目情况

个人简历

李振华，男，1983年8月生，江苏盐城人，分别于2005年6月、2008年6月（预期）获南京大学计算机科学与技术系学士、硕士学位，2007年5月-10月在香港理工大学计算机系任助理研究员。研究兴趣主要集中在P2P网络、P2P流媒体系统、分布式系统方面。

硕士期间发表学术专著1本、国际会议论文5篇（其中包括第一作者的IPDPS’08和ICPP’08会议论文）、国内期刊论文2篇，另有一篇《计算机学报》论文已通过初审。

硕士期间参加的科研项目

1、国家自然科学基金项目：实用化对等网络技术的研究（2005年-2008年）

2、江苏省自然科学基金前期预研项目：新型P2P计算技术的基础研究（2005年-2008年）

3、香港政府资助研究项目：无线传感器网络（2007年5月-10月）

附录2 攻读硕士学位期间撰写论文情况

专著：

陈贵海，李振华。**对等网络：结构、应用与设计**，中国计算机学会学术著作丛书。清华大学出版社，2007 年 9 月。中文，约 40 万字，370 页。这是国内第一本关于 P2P 网络的学术专著，目前已被国防科技大学计算机系、华中科技大学计算机系、华中师范大学计算机系及 PPLive 公司采用为研究生课程教材或公司培训教材。详细信息可参见网上链接 http://www.china-pub.com/computers/common/info.asp?id=36547。

会议：

（1）Zhenhua Li, Jiannong Cao, Guihai Chen and Yan Liu. **Fast Source Switching for Gossip-based Peer-to-Peer Streaming**. The 37th International Conference on Parallel Processing (ICPP 2008), 8-12 September 2008 in Portland Oregon, USA. (录取率：81/263=30.8%)

（2）Zhenhua Li, Jiannong Cao and Guihai Chen. **ContinuStreaming: Achieving High Playback Continuity of Gossip-based Peer-to-Peer Streaming**. The 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), 14-18 April 2008 in Miami Florida, USA. (录取率：105/410=25.6%)

（3）Zhenhua Li and Guihai Chen. **Detecting and Reducing Partition Nodes in Limited-routing-hop Overlay Networks**. The 6th International Conference on Grid and Cooperative Computing (GCC 2007), 16-18 August 2007 in Urumchi, Xinjiang, China.

（4）Junfeng Xie, Zhenhua Li and Guihai Chen. **A Semantic Overlay Network for Unstructured Peer-to-Peer Protocols**. The 13th International Conference on Parallel and Distributed Systems (ICPADS 2007), 5-7 December 2007 in Hsinchu, Taiwan.

（5）Zhiyu Liu, Ruifeng Yuan, Zhenhua Li, Hongxing Li and Guihai Chen. **Survive under High Churn in Structured P2P Systems: Evaluation and Strategy**. *Lecture Notes in Computer Science* (ICCS 2006), Reading, UK, May 28-31, 2006. (SCI Index: BEO22, EI Index: 063210048717)

期刊：

（1）李振华，陈贵海，曹建农。具有高播放连续度的 P2P 流媒体系统的设计。*计算机学报*，2008 年投稿，已通过初审。

（2）袁瑞峰，李振华，陈贵海。**基于虚拟节点交换方法优化 P2P 覆盖网性能**。*计算机科学*，2008 年录用。

（3）李振华，陈贵海，邱彤庆。**分点：无结构对等网路的拓扑关键点**。*软件学报*，2007 年录用。