中国计算机学会通讯
COMMUNICATIONS OF CCF
http://www.ccf.org.cn

对等计算研究概论

郑纬民 胡进锋 代亚非 袁泉 马永泉 宁宁 董海涛 洪春辉 张桦楠

前言（二级标题）

互联网系统的计算模式正在发生从客户机/服务器（client/server）模式到对等计算（peer-to-peer，亦简称 P2P）模式的转变。对等计算的核心思想是所有参与系统的结点（指互联网上的某个计算机）处于完全对等的地位，没有客户机和服务器之分，也可以说每个结点既是客户机，也是服务器；既向别人提供服务，也享受来自别人的服务。实际上，对等计算的概念在很早以前就已提出，但一直没有受到广泛的重视，主要是因为没有实际运行的系统作为背景。产业界和研究界都普遍认为在大多数情况下还是客户机/服务器模式更为合理。

然而，随着 PC 技术和互联网（Internet）的发展，个人电脑的能力越来越强，接入带宽也逐渐增大，如何更好地利用所有结点（尤其是原先处于服务器地位的结点）的能力搭建更好的分布式系统自然而然地成为人们关注的问题。1999 年 Napster[32]推出后迅速普及，成为对等计算的重要实例，从此之后，越来越多的 P2P 软件的发布和流行，一步步验证了对等计算思想的成功，如 Gnutella[17]、Freenet[42]、BitTorrent[2]、KaZaA[24]、Skype[46]等等。今天，对等计算应用已经超过 WEB 应用，成为占用互联网带宽最多的网络应用，其代表系统 KaZaA 的同时在线用户已超过 300 万，其发展之势愈演愈烈，成为业界持续关注与探讨的话题。

与对等计算在产业界迅速普及同时，研究界也及时跟进，对于对等计算系统的设计方法和发展方向进行了广泛而深入的研究。今天，对等计算仍是分布式计算领域关注的焦点，受到该领域所有重要国际会议的重视。2001 年提出的结构化覆盖网（structured overlay network）及分布式哈希表（Distributed Hash Table，DHT），更是引发了对等计算研究的热潮，在此基础上提出了各种大规模分布式系统，包括存储系统、DNS 系统、在线游戏、网页缓存、新闻组等等。同时，对于如何增强对等计算系统的各种性能，如：安全性、隐私性、公平性、可扩展性、系统开销、访问性能等等，也开展了深入研究。

事实上，对等计算已逐渐成为一种将来社会不可避免的计算模式，即：人人贡献出自己的资源、人人享受他人提供的资源。或许这种方式将遇到网格（grid）方式（所有资源和服务由某大型提供商提供，用户付费以获得资源并保证服务质量）的竞争，但是由于对等计算具有良好的可扩展性，可以对资源进行充分利用等优点，必然会长期存在下去，获得到更广泛的应用空间。

从理论上说，要实现对等计算的计算模式，必须完成三项工作：资源放置、资源定位、资源获取。在对等计算系统中，并非每个人的资源（比如数据）都放置在各自的机器上，很可能是所有机器共同管理资源。比如在对等计算存储系统中经常采用分布式哈希表（DHT）放置数据，各人数据可能放置在他人的机器上，于是如何进行资源放置就成了必须回答的第一个问题。

当某用户需要获得数据时，他首先需要找到该数据，当然这与资源放置方法是直接相关的。对于以分布式哈希表方式放置的数据，可以直接定位。但在多数文件共享系统中，用户的文件都是放在各自的机器上，那么特定用户如何知道哪些机器存有他需要的数据就成为一个关键问题，常常需要较大规模的搜索才可以完成。资源定位就是研究如何更有效率地找到所需资源所处的位置，尤其是一些在网络中稀有（rare）的数据。

当找到资源的位置后就需要获取资源，对于有些资源来说，这并不是很直接的事，比如计算资源、大文件和流媒体资源。这里的问题主要在于如何才能更高效地获取资源，或者说如何使一些热点资源能够为更多的需要该资源的用户服务。通常这需要尽量发挥对等计算系统中所有参与者的能力。例如，在 BitTorrent 系统中，如果大量用户同时下载一份数据，用传统的客户机/服务器模式，一个服务器不可能同时支持非常多的用户，而使用对等计算模式，所有正在下载的用户一方面从共享文件的源点进行下载，另一方面更多地从其他用户那里下载，这样充分地利用了所有结点的带宽资源，使得并行下载能力得到了极大的拓展。当前，BitTorrent 已经成为被广泛使用的下载工具，对其性质的研究也逐渐展开。

本文试图对对等计算研究的现状进行总体综述，但是由于对等计算的研究领域非常广泛，我们只能突出其中要点。如上所述，对等计算的研究从根本上是要解决资源放置、资源定位、资源获取三个问题，我们就针对这三个问题中的主要成果在第二部分中进行着重介绍。在资源放置中，介绍近年来对等计算领域最重要的研究成果分布式哈希表，以及支撑分布式哈希表的覆盖网路由协议；在资源定位上，介绍在文件共享系统中的文件搜索技术；在资源获取上，介绍大文件下载中的应用层垂直组播技术。在对等计算领域，我国研究人员近年来开展了广泛的研究工作并取得了优异成绩，不少论文在重要国际会议上发表，同时，也做出了高水平的实用系统，并获得广泛认可。第三部分中对其中的一些具有代表性的系统进行简要介绍。它们是北京大学的文件共享系统 Maze、清华大学的存储服务系统 Granary 和华中科技大学的视频点播系统 AnySee。最后对全文做简短的总结。

对等计算系统的关键技术（二级标题）

覆盖网路由协议（三级标题）

对于对等计算系统而言，能够适应的网络规模是一项非常重要的指标。然而，早期设计的系统，比如 Gnutella 和 Napster，在这方面都有一定的缺陷。前者使用的是不适合大规模系统的洪泛策略，后者引入了集中式的目录管理。

在这样的背景下，一批基于分布式哈希表的系统应时而生，包括 Tapestry[52]、Pastry[40]、Chord[47] 和 Content-Addressable Networks (CAN)[39]。在这些系统中，文件根据系统生成的标识（ID）排列。这种标识通常是文件名经过哈希计算的结果。系统中的每一个结点都和一个特定区段内的标识关联，并保存相关联标识对应的文件的信息。当分布式哈希表系统对标识进行查询时，相应的结点便会返回对应的信息。

分布式哈希表系统的核心是路由协议。系统中的分布式哈希表结点构成一个覆盖网，每一个查询操作都是通过这个覆盖网找到目标结点。所以，分布式哈希表系统的性能就取决于其所采用的路由协议的效率。虽然各种分布式哈希表系统的路由协议都不相同，但它们都具有一个共同的特点，就是每一个结点在覆盖网中拥有的邻居数目为 $O(LogN)$[1]，完成每一次路由所需步数都会在 $O(LogN)$ 内，其中 N 为系统总结点数。

现有的覆盖网路由协议简介（四级标题）

本节将介绍一些前文提到的典型的覆盖网路由协议。这些协议都是根据接收到的标识（一般是

[1] 即随着网络规模（结点数 N）的增长，邻居数目和下面提到的路由步数会以 $Log(N)$的速度增长。

一定长度的数字串），把信息路由到相应的结点。每个结点也具有一个标识符（ID）。而且，这个标示符通常是和它对应的文件的标识相同（当一个结点对应一个区间内的文件标识时，结点标识符一定位于这个区间之内，一般是区间的中点）。同时，每一个结点都维护一张路由表，记录其他一些结点的信息。当一个结点收到一个查询操作时，如果它发现所查询的标识不在自己关联的区间内，那么该结点将会把该查询发送给其路由表中它认为**最靠近目标**的邻居。各种算法中，对**“最靠近目标”**的定义不尽相同，但是通常都是根据结点标识符和目标标识进行定义。

Plaxton 树：Plaxton 树[38]是第一个能够被分布式哈希表系统大规模使用的路由协议，虽然该算法的初衷是针对静态网络设计，而非对等计算系统。该算法采用的是“前串匹配”的路由模式。每一步路由都会把信息传送到前串更加匹配目标的结点去。

Tapestry：Tapestry[52]是对 Plaxton 树的改进算法，使之能够部署在动态的网络之中。限于篇幅，具体的改进过程不再详述。

Pastry：在 Pastry[40]中，每一个结点都被分配了一个 128 位的结点标识（nodeId）；用于确定该结点在环状标识空间中的位置。每一个结点的结点标识是在结点加入系统时随机分配的。这里假设 ID 的分配是均匀的（实际上，现在的哈希算法能够保证这一点）。因为这种随机性，在标识空间中位置相邻（nodeId 最接近）的任意两个结点在权限、所处的网络拓扑、所有权和网络配置等等方面都可能有很大差异，换句话说，在标识空间相邻的结点，在实际网络中往往差异很大。

Chord：Chord[47]也是一个使用环状标识空间的系统。同样，针对一个标识的路由目标就是在数值上最接近该标识的 nodeId 的结点，并称为针对该标识的承接点。在 Chord 中，每一个结点都维护着两套邻居。其中一套为在标识空间中紧接着该结点的 k 个结点，另一套为指向在整个标识环中以该结点为基准依次折半的结点的指针。其中，第一套邻居是确保路由正确的关键。Chord 可以确保路由在标识空间中单向靠近目标结点而且不会越过，并且可以保证路由在 $O(LogN)$ 步内完成。

CAN：CAN[39]采用的标识是从一个多维（假设为 d 维）环状空间中选择的。每一个结点都与其标识周围的一个立方区域相关联，同时，它的邻居就是拥有和它相邻的立方区域的结点。路由时，消息一直是从一个结点传输到它的拥有与目标标识最接近的 nodeId 的邻居。与前面几种算法不同，CAN 中每个结点维护大小为 d 的路由表，同时，每一次路由都在 $O(dN^{1/d})$ 步内完成，特别当 $d = O(LogN)$ 时，CAN 的路由表和路由长度将与其他系统相似。

改善覆盖网路由协议的研究（四级标题）

状态与效率的权衡（五级标题）

最明显的对效率的衡量标准是路由路径的长度（步数），而与之直接关联的便是每个结点维护的路由表的大小。路由表的大小不仅仅影响到路由时需要判断的状态的数量，而且关系到网络发生变化时，状态需要随之改变的结点的数量（实际上，在对等计算系统中，结点的运算成本和存储成本都是低廉的。相对而言，网络开销的成本就要大得多，所以，研究者通常更加关心后者的变化）。

Xu[49]对这种基础的权衡进行了研究，见图 1。可以看出，在规模为 N 的网络中，当每个结点维护的路由表的大小为 N 时，路由将在 $O(1)$ 内完成，但是因为对等计算系统的动态性，这将会导致极其沉重的维护开销；当每个结点仅记录一个邻居时，路由效率为 $O(N)$。这样的延时令人无法忍受。这是图中的两个端结点，也是两种极限情况。图中同样标出了各种已经存在的路由协议对状态和效率的折衷。Xu 同时也提出了三种比现有分布式哈希表系统在权衡方面更优秀的算法，但是这些

算法在特定的结点上会造成极大的拥塞。

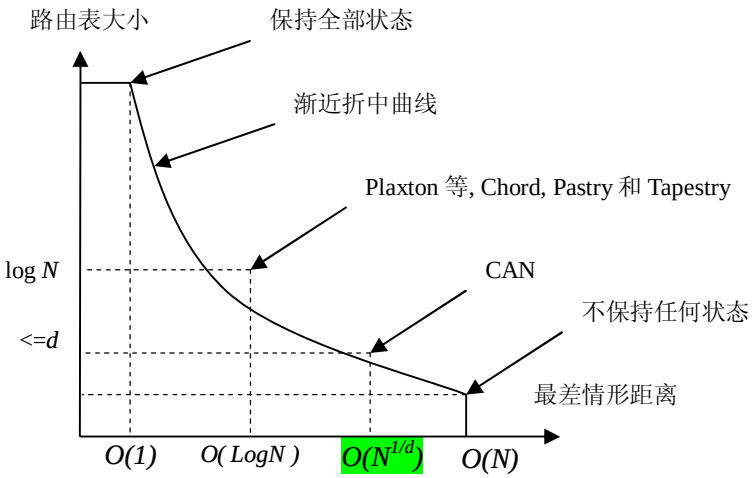

图 1. 路由表大小和网络直径关系曲线 log$N \leqslant d$

然而 Gupta **等**[18]认为，结点维护整个网络的信息是可行的。他们设计了一套良好的分层体系，以确保每一次网络成员的变化（例如结点的加入/离开）都能够在一定的时间内通知到系统中的所有结点，并且仅消耗合理的带宽。这个方案有一个非常突出的优点，就是每一个结点都仅仅需要主动维护其路由表中的一小部分结点，并在这部分结点状态发生变化时在整个系统里发起广播。

最佳路径（五级标题）

前面所讨论的效率的标准是建立在应用层的路由步数的基础上的，然而，实际的路由效率是由结点与结点之间的网络延迟决定的。结点与结点的实际距离在覆盖网中是透明的，所以，在覆盖网中的一步在实际网络中有可能是跨越大洲的跳跃，而其他的一些可能只是局域网中的传输。没有考虑上述因素的路由协议往往会导致很高的路由延时。为适应实际的网络拓扑，至少有以下三种途径：

选择最佳路径：这种方案在路由过程中选择每一步的转发结点时，不仅考虑与目标在标识空间最接近的结点，同时也将考虑与目标之间网络延迟最短的结点。不同算法采用的策略不尽相同，但是它们都采用同样的基本手段处理在标识空间的靠近和在实际网络中的延时的权衡：在路由过程中选择下一步结点的时候，将对所有可能的走法进行比较，找出延时最小的结点或者是对标识空间距离和实际距离的最佳折衷点。模拟数据表明，这种做法可以有效地降低平均路由时间，但是由于实际网络对于覆盖网而言是透明的，所以这种方法的性能很大程度上取决于覆盖网中每一步路由可以选择的路径数量。

最佳邻居的选择：与上面介绍的方法不同，这种方法在选择邻居的时候考虑了实际网络距离的长短，而非仅仅在路由过程中考虑。在构建覆盖网的过程中，每一个结点对自己邻居的选择都是从所有在标识空间符合条件的点中选择与自己在拓扑中实际延迟最短的点。这种方案的实际效果取决于每个结点在覆盖网协议允许的前提下可选择的邻居结点的数量。

规划覆盖网构架：在大多数算法中，结点的标识是随机赋予的，对邻居的选择则是建立在这些随机获得的标识的基础上。然而，应该可以根据网络的状态来对标识进行非随机选择，以达到优化的目的。在这一点，CAN[39]作了初步的尝试，并取得了较好的效果。在 CAN 成功的原因中，多维标识空间起了至关重要的作用。最近的研究[33]指出，多维的几何空间可以形象地模拟出互联网的延

迟。这就引出了采用一维标识空间的系统是否能够有效地优化网络构架的问题。然而，在前面提到的两种优化的前提下，对覆盖网构架的规划并不能使系统的延时进一步有效降低，同时，这种相关性的规划会影响到覆盖网的鲁棒性等其他性能。

互联网络的异构性（五级标题）

所有的路由协议起初都是假设网络中所有的结点具有相同的能力，然后增加适应网络异构性的技术。然而，在现有的对等计算网络中，由于带宽的差异产生的异构性还是非常显著的[43]。利用这种异构性可以提高路由协议的性能。

最简单的解决网络异构的技术是结点克隆，也就是把能力强的结点当作若干能力较弱的结点使用。例如，一个强结点具备 10 个弱结点的能力，那么，在覆盖网中便把这个强结点当作 10 个弱结点处理。这样，在进行路径优化的时候，认为这些拆分出来的结点之间的距离为 0，这些结点之间进行的路由仅仅是在标识空间发生跳跃，在实际网络中，并不发生数据迁移。

数据搜索技术（三级标题）

随着越来越多的数据存储到对等计算系统中，上层应用就需要底层架构提供数据定位与搜索能力。对等计算系统中常见的搜索功能有两种：数据定位和关键词搜索。数据定位是指以对象 ID 作为输入，得到对应的数据或存储这些数据的结点。关键词搜索则是指输入一个或多个关键词，然后得到包含这些关键词的数据。非结构化的对等计算系统主要依靠泛洪（flooding）或随机漫步（random walk）[31]来处理查询，在这样的系统中这两种搜索几乎没用区别。而对于结构化的对等计算系统，这两种搜索存在着巨大的差异。这主要是由于到目前为止，几乎所有的结构化对等计算系统都建立在分布式哈希表的基础上。分布式哈希表能够很自然地实现数据定位，而对于关键词搜索却无法提供直接的支持。下面介绍最近几年中出现的几种主要的数据定位与搜索技术。

非结构化对等计算系统中的搜索（四级标题）

基本搜索技术（五级标题）

在非结构化的对等计算系统中，泛洪和随机漫步是两种最基本的搜索方式。Gnutella[17]是最著名的非结构化对等计算系统，它就是采用泛洪的方式来处理查询。当使用泛洪来定位一个文件时，查询发起结点会发送查询到它所有的邻居结点，而这些邻居结点会继续把查询发送给它们各自的邻居结点。这个过程会一直持续到距查询发起结点一定半径内的所有结点都收到了该查询。而对于采用随机漫步技术的系统，每个结点都会在自己的邻居结点中随机选择一个结点，并把查询发送给它。

搜索的优化（五级标题）

目前，泛洪和随机漫步的搜索效率都不尽如人意。为了使搜索的效率更高，可扩展性更好，就必须采用一些优化措施。在非结构化的对等计算系统中，搜索的优化主要有三种思路。一是信息聚合，即结点汇总来自其他结点的内容。采用这种方式，结点能够获得更多关于整个系统的信息，因此有望提高搜索效率。第二种方式是内容聚类，即系统中的信息根据语义或用户的兴趣进行聚类。可以用聚类的信息来减少搜索时涉及的结点。第三个方向则是利用结点的异构性，该技术尚在探索中。

定向泛洪与偏向性随机漫步：定向泛洪指结点仅将消息发送给其部分邻居结点，这些结点被认为有望获得更满意的查询结果。偏向性随机漫步则是指在随机漫步中，不是随机地选择结点，而是

选择那些更有可能获得更满意结果的结点。与一般的泛洪与随机漫步相比，这两种技术不但可以减小通讯开销，还可以提高搜索结果的质量。

R* 步复制**：在使用该技术的系统中，每个结点维护距该结点 ***R 步之内所有结点的数据的一个索引。当结点收到查询时，就可以代表距其 ***R*** 跳数内的所有结点来回答该查询。这样可以有效地减小开销。

超级结点：在采用这种技术的对等计算系统中，所有超级结点连接成一个对等计算网络，而普通结点则连接到一个或多个超级结点上[24] [50]。超级结点通常具有较强的处理能力，足以维护包含所有与其连接的普通结点的信息索引，查询通常由超级结点来处理。

基于分布式哈希表的关键词搜索（四级标题）

结构化对等计算网络都实现了分布式哈希表，并利用分布式哈希表将数据项映射到结点。上层应用可以插入一对<key, value>到系统中，并通过 key 得到 value。

大多数基于分布式哈希表的关键词搜索协议都使用倒排索引。在对等计算系统中，逻辑上的全局倒排索引必须被分割存放在若干结点上。目前对等计算系统中有两种基本的索引分割和放置策略：本地索引和全局索引。在采用本地索引的系统中，每个结点维护一个本地倒排索引，其中包含了该结点保存的文件的信息。进行搜索时，一个查询必须发送到所有的结点。非结构化对等计算系统通常采用这种策略。而全局索引策略把一个关键词对应的倒排表都存放在一个结点上，每个结点维护若干个这样的倒排表。结构化对等计算系统通常采用这种策略。

全局索引及其优化（五级标题）

如前所述，结构化对等计算系统中最常见的搜索方式就是全局索引，即每个关键词及其对应的倒排表通过分布式哈希表映射到系统中某个结点上。与多个关键词相关的查询必须发送到与这些关键词对应的所有结点上，相关的所有倒排表都被返回，然后通过求交运算得到最终结果。采用这种方式，对于一个与 k 个关键词相关的查询，最多需要与 k 个结点通信。然而倒排表求交集的运算需要把相关的倒排表都通过网络进行传送，其带宽消耗和时间代价都是相当可观的。为缓解这一问题，提出了许多优化方案，其中比较重要的如下：

压缩：为了减小通讯开销，数据必须经压缩后再发送。目前 bloom filter[3]是最常见的压缩方式。Bloom filter 是一种基于哈希的数据结构，它能够提取出一个较小集合中的内容信息，并以较高的成功率进行匹配。如果两个结点上的倒排表需要求交，结点 A 可以压缩其倒排表并发送 bloom filter 到另一个结点 B。结点 B 将 bloom filter 与其上存储的倒排表求交，再将结果压缩并返回 A。结点 A 就可以对结果进行整理后返回给用户。

缓存：对于采用全局索引的对等计算系统，缓存技术主要应用在两个地方：倒排表缓存和结果缓存。倒排表缓存是指结点将查询过程中收到的倒排表缓存起来，以避免多次重复地对该倒排表进行网络传输。结果缓存则意味着被缓存的是查询的结果。关键词的流行程度是符合 Zipf 分布[2]的，这就意味着大多数的查询集中在部分少数关键词上，因此缓存技术能够有效地减少通讯开销。

预计算：采用这种技术的系统会提前计算某些倒排表的交集，并存储起来。该技术与缓存技术有某些相似之处。

[2] Zipf 是一位统计学家和语言学家。他在对语料出现频度进行统计分析后发现：如果对某个语言单位（不论是字母还是词）进行统计，把这个语言单位在一个语料库里出现的频度（frequency）记作 F，并根据频度的降序对每个单元指派一个整数的阶次（rank）R，则 R 和 F 的乘积近似为一个常数，即 $F \times R \approx const$（常数）。

增量式交集：在多数查询中，用户需要的其实是部分最相关文档的交集，而不是全部结果的交集。这就使我们可以采用增量式交集的方式有效地减少通讯开销并改进效率。对于两个倒排表求交，可以将其中一个倒排表分割成若干块，然后边传送边计算。当某些条件满足后即可不再传送其他的块。

混合索引技术（五级标题）

全局索引和本地索引都有其各自的优点和不足。目前有几种技术混合采用了这两种基本方式，从而达到更好的性能。主要的混合索引技术包括：eSearch[48]、多层分割（Multi-Layer Partioning）[45]以及混合搜索底层设施（Hybrid Search Infrastructure）[30]。

应用层组播算法（二级标题）

在互联网的实际应用中，常常会遇到一点到多点或多点到多点通信的情况，即群组通信。在群组通信中，许多结点与同一群组中的其他结点同时交换信息。例如一个有许多人参加的网络电话会议。其他应用包括在线流媒体和高效的大块数据散发。在这些群组通信的应用中，组播是一个实现数据从一个结点到多个结点高效散发的关键组件。但是，尽管网络层的 IP 组播已经被提出了十多年（例如[10][11][12][16][28]），由于没有广泛的部署和难以跟踪组成员的变化，使用 IP 组播技术的应用很少。为了解决这一问题，近年来提出了一些替代 IP 组播的方案，统称为应用层组播或端到端组播。这些组播使用网络层的单播，通过建立一个覆盖网，在应用层实现数据组播。

应用层组播具有许多网络层组播不具备的优点。首先，由于不需要路由器的支持，应用层组播可以在现有的基础上渐进部署。其次，应用层组播相对于网络层组播来说更灵活，能够适应上层应用不同的散发要求。Saltzer[41]认为把组播的功能放置在系统的底层（例如网络层）可能是冗余的，或者和导致的成本相比较是没什么价值的。这部分是因为底层完成这些功能的效率低，部分是因为底层缺乏关于上层应用的知识。显然，面向应用的组播应当使用应用层的组播而不是网络层的组播。

应用层的组播已成为当前的一个研究热点。人们提出了许多可扩展的组管理和可扩展的、可靠的消息组播算法[1][13][5][54][9][14]。对这些系统来说，仍存在的挑战是建立一个支持可扩展和容错的基础结构，同时保证低延迟和网络资源的有效利用。换句话来说，为了实现高效的基于覆盖网的组播系统，必须解决以下三个关键问题：

1. 什么是适当的组播结构？另外，如何在覆盖网上以一种分布式的方法构建出这样一个结构？
2. 覆盖网如何管理数量众多的参与结点，特别是参与结点在能力和行为方面存在巨大差异的时候？
3. 如何让组播算法和覆盖网对多变的互联网环境具有自适应性，例如变化的链路、拥塞、网络分割（network partition）等？

第一个问题常常被归结为在覆盖网上构造和维护一个高效容错的生成树（spanning tree），因为生成树是一个天然的适合组播的结构。近来对等网络方面的研究力图超越单棵树的结构，进一步利用树中兄弟结点之间的“垂直链路”来克服构造树的难题。第二个问题在 IP 的组播中称为“成员管理”。由于系统的巨大规模和分布式的特点，这个问题在覆盖网和对等网络中变得越发复杂。事实上，覆盖网有时候仅仅由志愿结点组成，我们无法依赖一个可靠的、强大的结点来管理组成员。这时，使用分布式的、自组织的结构来管理组结构是适当的。第三个问题可以通过不停地测量覆盖网

的拓扑，并且采用自适应于拓扑变化的算法来解决。覆盖网常常认为其内部两个结点之间的链路是直接连接的、独立的，不管它们在实际上是否共享物理链路或路由器。因此通过测量底层的“黑盒”（主要是覆盖网中结点间的可联通性和带宽），覆盖网能够感知底层拓扑的变化，进一步提高性能。

下面我们将解释现有的设计和算法如何解决这三个基本问题。

组播结构的构建（3 级标题）

为了实现覆盖网的组播，主要的问题是构造组播的结构，即数据流动的路径。考虑一个简单的情形（大多数群组通信可以简化为这样一个模型）：只涉及一个源结点和许多目的结点，我们的目的是尽快地把数据从源结点散发到所有的目的结点。为了解决目的结点数目巨大的问题，一些目的结点充当数据传送路径上的接力结点，把自己收到的数据转发给其他目的结点。因此，许多应用把组播树作为从一到多应用的基本结构。给定覆盖网的带权连接图 $G=(V, E)$－其中 V 代表所有的参与结点，E 表示覆盖网的连接。理论上，这个问题转变成构造一棵以源结点为根的最小生成树（MST[3]）的问题。尽管有构造最小生成树的有效算法，在实际上构造这样的一棵树却不简单，因为：在互联网上取得完整的覆盖网的连接图不容易；同时，构造出的最小生成树常常包含高出度的结点，这些结点很容易过载，成为系统的瓶颈。因此，在实践中许多组播算法不寻求构造最优的生成树，而是根据部分连接图构造出度受限的近似最优生成树。组播结构主要考虑的问题是：1）减少延迟和提高带宽；2）容错；3）构造的难度，特别是用分布的方式。

Overcast[4][23]试图解决利用互联网发送带宽密集型数据的问题—一个内容提供商面临的难题。Overcast 由一个中心的源结点、散布于网络的许多 Overcast 内部结点（具有持续存储的标准个人电脑）、以及许多具有标准 HTTP 连接的客户机组成。Overcast 利用一个简单的树状结构构建协议，把内部结点组织成一棵以源结点为根的组播树。Overcast 的树构建算法的目标是使每个结点到根结点的带宽最大化。

Bayeux[5][54]利用从覆盖网路由协议 Tapestry 继承的前缀匹配的路由协议。这样，它结合了负载平衡、可扩展到任意规模接收者的可扩展性、近邻特性、以及网络容错等特点。Bayeux 伴随了基于分布式哈希表的路由带来的“天然的组播树”。既然可以很容易地从大多数基于分布式哈希表的算法中派生出一个树结构，那么利用这样一棵树来进行覆盖网的组播，而把故障恢复和近邻问题留给覆盖网的路由层就很有好处。

然而，基于树的组播协议存在一些固有的缺点，并且有时难以充分利用协作环境中的可用资源。造成这个问题的原因是：在任何组播树中，承担复制和转发数据的结点只是组播树中的内部结点，这些内部结点只是所有参与结点的一小部分。大部分结点是叶子结点，没有贡献任何资源。这个结果和希望所有结点都承担转发负载的初衷相违背。这个问题在高带宽的应用中进一步恶化。例如在视频或大文件发送这样的应用中，许多接受者可能没有担当传统组播树中的内部结点的能力。另外在树的结构中，从上到下，带宽一般是单调递减的。较高层的任何损失都会减少低层能够取得的带宽。尽管许多技术被提出来去恢复丢失的数据，由此提高可取得的带宽（例如[1]、[4]），然而，一个结点获得的带宽却是只由它在树中的惟一的一个父亲结点决定的。

为了克服树结构的天然缺陷，近年来的研究提出使用“垂直链路”来增大结点通过树结构获得

[3] Minimal Spanning Tree
[4] 在 Cisco 公司支持下开展的一个研究项目，目的是构建针对因特网内容分布的特点的一个体系结构，实现可靠的组播业务。
[5] 一种应用层组播方式，可支持组树木很大的组播但对业务形式有严格限制。

的带宽（Bullet[6][25]），或者使用多棵树来取代单一的树（SplitStream[7][8]）。基本上，这些技术具有一个共同点：利用覆盖网中的每一个结点来增强数据的传输，而不是仅仅利用单棵树中的内部结点。

SplitStream 的关键思想是把内容分成 *k* 个带（stripe），每个带利用一棵独立的树来组播。结点根据它们愿意接收的带的个数加入同样多的组播树中。每个结点同时还给定一个它们愿意转发的带的上限。于是问题转变为如何构造一个组播树的森林：在这个森林中，一个在某棵树中是内部结点的结点在其他树中都是叶子结点，并且满足结点的出度限制（即转发的带的个数）。这确保转发负载分布到所有参与的结点。例如，如果所有的结点都希望接收 *k* 个带、转发 *k* 带，SplitStream 将会构造一个负载均衡的森林，这个森林还是低延迟和低链路负载的。图 2 描绘了 SplitStream 中的带和组播森林的结构。

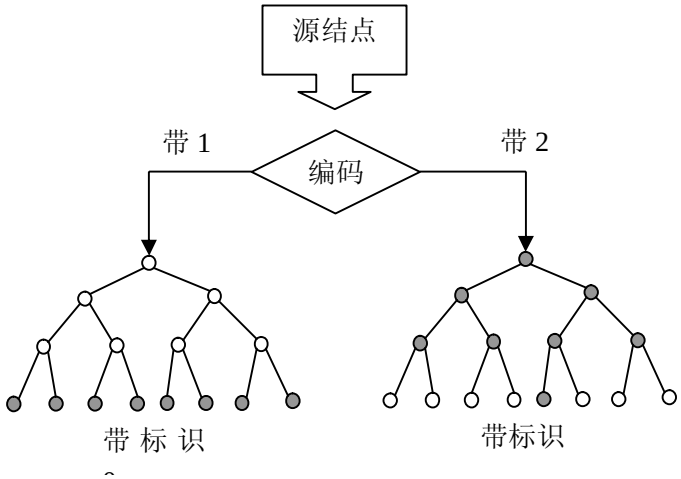

图 2. 带和 SplitStream 的组播森林

设计 SplitStream 面临的关键问题是以一种分布式的、可扩展的、高效的、自组织的方式构造一个由**内部结点正交**的组播树构成的森林。一组树是内部结点正交的，意思是说一个结点最多在一棵树中担当内部结点，而在其他树中均为叶子结点。SplitStream 利用 Pastry 的路由和 Scribe[8][7]的组成员管理来辅助内部结点正交树集合的构建。

Bullet 面向高带宽的组播数据发送，例如把巨量的内容通过互联网发送到多个接收者。Bullet 提出在组播覆盖网中通过结点协作传送正交的数据集，而不必把相同的数据流发送给所有的结点。结点仍然从父结点那里接收到一组数据，但它们还必须寻找能给其提供缺少的数据的结点。Bullet 设计了专门的分布式算法使得数据可以均匀地散布在所有参与结点之中。用这个方法使其避免了定位最后一个数据块的问题，取得持续的高带宽的数据发送。

组成员管理（Scribe）（3 级标题））

组播系统除了组播的结构，还需要管理参与的结点，特别是把结点根据特性组织成不同的组，

[6] 由美国Duke大学Dejan Kosti´等人提出的一种通过在互联网上进行结点传播来自组织宽带覆盖网的算法，该算法可以适应不同规模网络。

[7] 美国 Rice 大学开发的一个多点传输系统。

[8] 一种结构化的覆盖网

和把消息发送到一些特定组的所有成员。组成员管理是覆盖网的一个基本服务，也常常是许多组播程序的一个必要的组成部分。

Scribe 提出了一个面向大规模组的成员管理的分布式的应用层组播结构。Scribe 建立在 Pastry 之上，利用了 Pastry 的网络近邻、容错、分布式等显著特点来同时支持巨量的组。任何 Scribe 结点都可以创建一个组；其他结点可以加入这个组，或者把消息组播给这个组的成员（假定它们有恰当的证书）。每个 Scribe 组都拥有一个惟一的组标识；结点标识和这个组标识最接近的结点充当这个组的会合点(rendezvous)；这个会合点是为这个组创建的组播树的根。如果一个 Scribe 结点希望加入一个组（例如 g），它会要求 Pastry 路由一个以 g 的组标识作为消息标识的消息 JOIN。这个消息向 g 的会合点路由过去。路由路径上的每个结点查看自己的组列表，看自己是否是 g 的转发器。如果是，则把这个结点加入到自己的孩子表中，接纳为孩子结点。否则，它为 g 创建一个条目，把这个结点加入自己的孩子表中，还通过把一个 JOIN 消息发送给这条从源结点到会合点的路由路径的下一个结点而成为 g 的一个转发器。因此，组管理机制对于组规模大小不同的组来说都是高效的。Pastry 的随机性确保树是平衡的，并且转发的负载均匀散布在所有的结点之中。这种平衡使得 Scribe 可以支持许多有非常多成员的组。更进一步来说，加入请求以一种分布的方式吸纳，不需要会合点集中处理所有的加入请求。因此，Scribe 可以作为基于覆盖网的组播的基本组成部件，一个例子就是前面介绍的 SplitStream。

CoopNet[9][17]面向现场直播的流媒体应用。这些应用把一个服务器的内容发送给许多可能具有高度动态性的接收结点。CoopNet 使用客户端的组播来减轻流媒体服务器的负载，并且帮助服务器克服瞬间冲击的问题。CoopNet 没有使用纯粹的对等计算方式。因此，CoopNet 受益于一个比分布方式更高效的中心的管理。CoopNet 有一个指定的工作站负责管理结点的加入和离开。工作站把组播树的整个结构存储在内存中。当一个结点开始接收现场直播的流媒体时，这个结点与工作站接洽加入的操作。工作站从保存在内存的组播树中找到一个合适的位置，把这个结点的父结点返回给这个结点。这个方式是非常高效的：它只需要一轮消息。尽管集中管理可扩展性不好，但是研究表明 CoopNet 的管理任务是非常“轻”的：一个有 2GHz Mobile Pentium 4 处理器的笔记本电脑可以支持每秒 400 个加入和离开消息。因此，集中管理也是可行的，在实际应用中可以扩展到很大规模。

基于覆盖网的测量和组播调整（3 级标题）

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

当前很多基于检测的技术[35][6][44][15][36][19]可以测量覆盖网的连接。这些技术使用轻量的消息检测来估计结点间的延迟与带宽。在这些技术中，RTT（round-trip time）和 10KB 的 TCP[10]检测是常用的方法。另外，一些方法[26]试图去估计瓶颈带宽。尽管检测消息和轻量的估计不能完全勾画连接的特性，但它们在故障检测和路径选择方面是很有用的。通常，覆盖网把结点间的连接看作“黑盒”，而黑盒测量对基于覆盖网的组播系统来说已经足够了。

检测出变化之后，覆盖网需要调整自身和组播结构来提高性能。许多覆盖网的设计采用了组播结构的局部调整：允许结点动态地选择能够提供更好服务的结点。在 Overcast 中，为了获得移动数据时观察到的近似带宽，协议测量下载 10KB 花费的时间。这个方法给出的结果好于使用底层带宽

[9] 微软的一个研究项目，试图用对等计算通信来改进客户机/服务器体系结构的性能和功能。
[10] Transmission Control Protocol 传输控制协议

测量（例如 ping）得出的结果。除此之外，结点还通过周期性地测量到当前兄弟结点、父亲结点、和祖父结点的带宽来估计自身的位置。结点直接测量到祖父结点的带宽，作为对做当前父亲结点的孩子结点是否正确的一个测试。如果必要的话，结点向上移动一层，成为它原来父亲结点的兄弟结点。由此，Overcast 内在地可以容忍除了根结点之外的结点失效和网络拥塞。

国内研究系统介绍（2 级标题）

Maze 文件共享系统——北京大学（3 级标题）

Maze 是网络应用向对等计算结构发展的一个实例（4 级标题）

Maze是北京大学网络实验室开发的一个中心控制与对等连接相融合的对等计算文件共享系统，在结构上类似于Napster，对等计算搜索策略类似于Gnutella。网络上的一台计算机，不论是在内网还是在外网，可以通过安装运行Maze的客户端软件自由加入和退出Maze系统。每个结点可以将自己的一个或多个目录下的文件共享给系统的其他成员，也可以分享其他成员提供的资源。Maze支持基于关键字的资源检索，也可以通过好友关系直接获取资源。和一些商业的对等计算系统不同，Maze是一个以科研为目的开发的并且得到广泛应用的系统。由于它的控制功能集中在可控制的结点上，使得Maze能够收集到整个系统运行中各个环节的大量信息，为进一步开展对等计算系统研究积累了宝贵数据。

Maze 的结构及核心技术

图 3 是 Maze 的系统结构图。

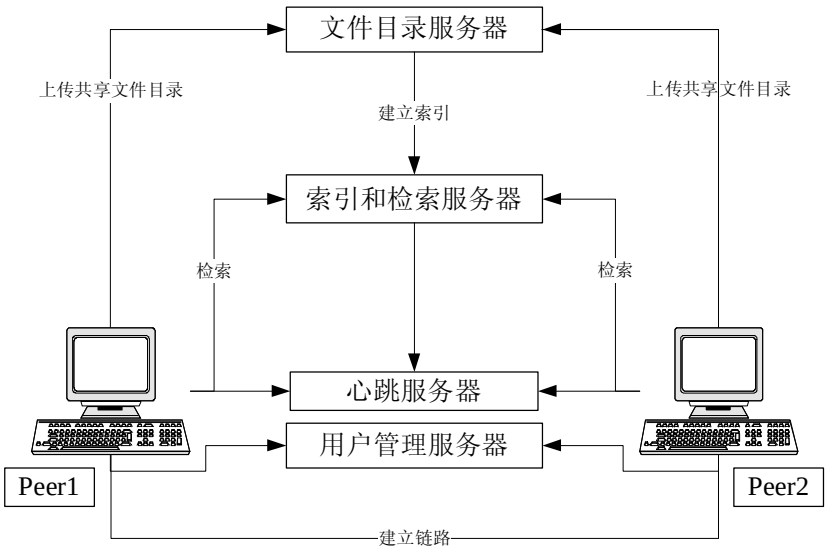

图 3. Maze 系统的结构图

在这个系统中的每个结点就相当于一个传统 FTP[11]服务器和 FTP 客户端的结合体。整个系统除了多个结点外，还包括集中式的用户、目录、检索、心跳服务器。用户管理服务器实现用户注册与身份认证。目录服务器负责把每个结点的共享的目录列表收集到集中的数据库。检索服务器读取数据库，为所有结点的文件目录建立索引并提供 XML[12]接口的检索服务。心跳服务器负责维护在线用户的列表。每个在线的结点每隔几秒向心跳服务器发送心跳包。同时每个结点每隔一段时间就把自己的目录信息在 Maze 的目录服务器上更新。检索服务器定期重新建立索引，并根据心跳服务器提供的在线状态只显示在线用户的文件检索结果。

在这个体系架构上，通过一系列的核心技术保证文件共享系统的提供高效可靠的服务。这些核心技术主要体现在如下方面：

- 结点的发现：采用分布式认证算法保证用户认证的安全性和有效性，支持结点的身份认证和任意两个结点间的通讯。
- 文件共享与传输：支持资源目录的浏览，确保资源可下载。通过 MD5[13]算法确认多个相同资源，以此提供多点并行下载功能，提高下载速度。其中还包括种子机制，帮助热点资源的快速扩散。
- 跨越防火墙技术：系统通过代理转发技术，解决了 NAT[14]问题，使得不论是内网还是外网的用户都建立联系，互相通信，传输文件。
- 基于关键字的搜索技术：通过中心索引服务器建立索引文件，支持基于关键字的匹配查找。
- 共享的激励策略：通过积分计算方法给那些愿意提供共享文件的用户增加积分，使得他们享有较高的下载带宽和较少的等待时间，以此鼓励对系统有贡献用户。相反，对有自私行为的用户则减少积分降低服务质量作为惩罚。
- 实现社会性软件的特性：社会性软件应具有帮助人们建立社会网络和自动组织群体的特点。Maze 通过提供加入好友，查询好友，好友间通信等功能，在结点间建立了基于好友关系的社会网络结构。
- 对等计算搜索技术：采用类泛洪策略，沿着好友关系链直接进行对等计算形式搜索。

在 Maze 上正在开展的研究内容

随着 Maze 用户数的迅速增长，系统的规模越来越大，使得我们可以在 Maze 提供的真实的环境下，依据基于真实的数据开展研究工作，目前已经开展的研究有以下几方面：

1. 分析了 Maze 日志，获得很多有意义的结果，验证了 Maze 的好友关系构成的社会网络符合小世界（small world）现象[15]。
2. 统计出了网络中资源的分布情况，包括类型的分布、大小的分布、数量的分布、访问频率、下载频率等。这些结论对于资源的管理、冗余控制、可靠性可用行性研究、建立生命周期的规律模型等都是很重要的依据。

[11] File Transfer Protocol 文件传输协议
[12] Extensible Markup Language 可扩展标记语言
[13] 一种单向加密的加密算法
[14] Network Address Translation 网络地址转换
[15] 所谓小世界现象，或称“六度分离（six degrees of separation）”，是社会网络（social networks）中的基本命题，即每个人只需要很少的中间人（平均 6 个）就可以和世界上另外任意一个人建立起联系。

3. 研究了 Maze 用户的行为特征，初步总结了“搭蹭车（free riding）”现象发生发展的规律。这是对等计算领域普遍关心的问题之一，尚没有很好的解决方案。我们的结论有利于确定更有针对性的激励策略和建立更有效的管理模式。
4. 研究了一种建立在好友关系之上的非结构化的泛洪搜索算法。很多相关研究着眼于对搜索算法的改进，提出的方法包括分层次搜索的方法，引入超级结点等。目前这些研究由于缺少实际环境的检验，只能在模拟的结果上进行讨论。Maze 上实现对等计算搜索将会给出一个可信的结论。

后续研究

Maze 从第一个版本发布到现在仅一年多时间，速度发展之快超出预期。目前 Maze 的现状为：注册用户达到 41 万人，最高在线用户达到 3 万人，最大同时在线用户数 8000，索引文件数 1.5 亿，目录文件容量 200TB，平均每天有 10 万个查询请求，平均每天大约有 100 万个文件被下载。

基于 Maze 的后续研究包括：

1. 在 Maze 的每个结点上运行一个计算代理软件（agent），使 Maze 成为一个大规模的分布式计算环境，借此展开与计算网格相关的研究。
2. 加入音频视频的组播，使 Maze 成为一个大规模协同工作环境。
3. 利用 Maze 结点对网络状态进行观测，使 Maze 成为一个大规模网络环境测试平台。
4. 以 Maze 为基础，研究用户的一般社会行为。
5. 利用 Maze 的用户日志，开展深层次的数据挖掘研究。

Granary 广域存储服务系统——清华大学（3 级标题）

Granary[53]是清华大学自主开发的对等计算存储服务系统。所谓对等计算存储服务系统是指：存储服务的提供者在互联网中部署一定数量的存储服务器，为用户提供数据存储服务，确保数据的可靠性、可用性、安全性和访问效率；存储服务的使用者按照所存储数据的容量和质量付费。与已有系统相比，Granary 具有的显著特性为面向对象的数据存储和访问方式。

Granary 以“对象”（Object）格式存储数据。对象的概念与面向对象程序设计（OOP）中的对象概念类似，即由一系列用户自定义的属性（attribute）组成。基于此，Granary 进一步支持属性级的数据查询。这种访问特性对于提供公共存储服务的存储系统是非常有用的。因为存储系统的提供者必须同时提供很多方便且实用的上层应用，才能推广其系统的使用。面向对象的数据存储和查询可以极大地简化上层应用的编码实现。上层应用的开发者可以直接将其面向对象应用程序中的对象数据存储至 Granary 中，随后通过查询的方式直接获取数据，而不需要额外地进行应用程序与存储系统间的数据转换。而如果使用文件系统或数据库作为存储后台则无法避免这种转化。

另外，Granary 设计了专门的结点信息收集算法 PeerWindow 和结构化覆盖网路由协议 Tourist。与已有算法和协议相比，它们具有更好的自适应性，即可以在当前的网络环境下自动优化，取得最佳效率。PeerWindow 和 Tourist 使得 Granary 可以在更多更复杂的环境下运行。当前 Granary 在清华大学校园网中运行，并将很快在 PlanetLab 平台上部署提供实际存储服务。

Granary 体系结构（4 级标题）

Granary 为上层应用提供独立的客户端应用程序开发接口（Client API），使得用户可以按照对象的格式存取数据。对象由一系列属性组成。一个属性由<名称、类型、可见性、值>(<name, type, visible, value>)构成。名称是由对象的拥有者定义的该属性的标识；类型必须为数值、字符串、二进制数据块之一；可见性表示该属性在存储中是否应被加密；值是该属性的实际数据。

在将对象存入到 Granary 中之前，用户必须预定义它的类（class）。类描述了属于该类的对象应该拥有那些属性。这同样与面向对象程序设计是一样的。

用户对其存储的数据可以通过提交查询的方式进行读取。例如在个人存储的电影中查询 “由冯小刚执导的所有电影”等，系统就会返回符合查询条件的所有电影的列表。Granary 只支持在不加密存储的属性上进行的查询。

Granary 的存储空间被划分给不同用户。用户之间完全隔离，不能通过存储层共享或交换数据。每个对象数据在系统都拥有一个惟一标识 objectId，形式为“用户名+类名+对象名”。所有用户的用户名都不一样，一个用户不能创建两个同名的类，因此 objectId 总是全局惟一的。

由于分布式哈希表在结点变化时需要经常移动数据，Granary 没有直接采用分布式哈希表存储实际数据。Granary 的数据放置方式如图 4 中所示。一个对象有多个复本存储在系统中，这些存储复本的结点（称为 Object Node）的 IP 地址列表（Object List）由分布式哈希表来存储。这样的放置方式给复制算法的设计留下了充足空间。类数据（用于描述对象格式）的放置与对象数据的放置非常相似。对每一个类，Granary 放置多个复本到不同结点上（称 Class Node），然后将这些类结点的 IP 地址列表（Class List）存储到分布式哈希表中。

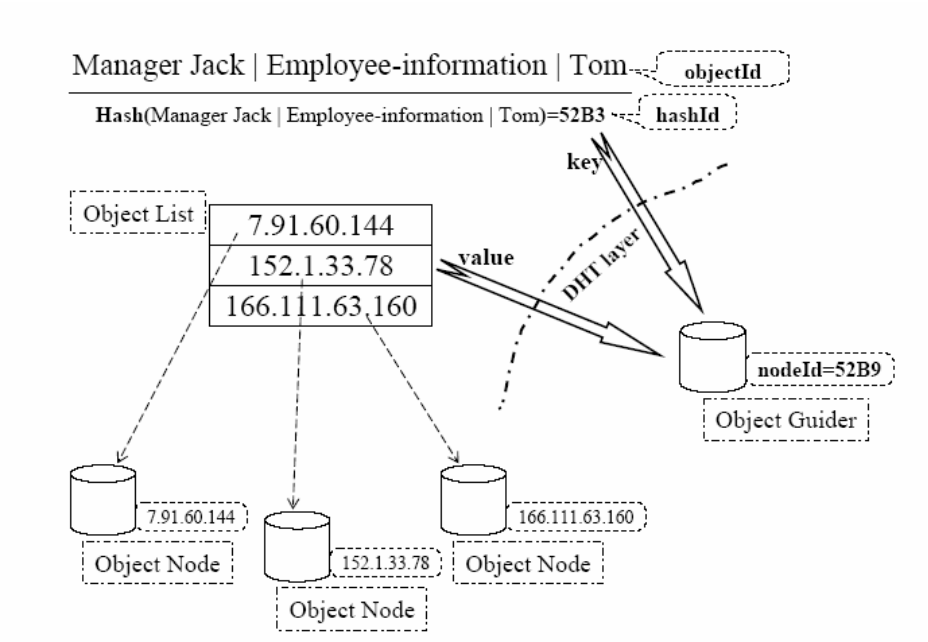

图 4. Granary 对象数据放置方法示意图

结点收集协议 PeerWindow（4 级标题）

PeerWindow[21]位于 Granary 的最底层，每个结点借此收集其他结点的信息。PeerWindow 的框架结构如下所述：

每个 Granary 结点都拥有一个 128 位的 nodeId（结点标识），为其 IP 地址和端口的哈希（Hash）值。nodeId 在地址空间中均匀分布。每个结点保存一个 Peer List，Peer List 中包含大量指向其他结点的指针。每个指针包含相应结点的 nodeId、IP 地址、端口、级别（level）以及一些上层应用附加的信息。级别由各个结点独立决定，可设为 0、1、2、……等等。规定：**一个 *l* 级的结点的 Peer List 中包含所有 nodeId 前 *l* 位与自己的 nodeId 前 *l* 位相同的结点的指针**。这样，一个 0 级结点就知道系统中所有结点的信息，1 级结点知道约半数结点的信息，*l* 级结点知道约 $1/2^l$ 个结点的信息。

Peer List 的维护通过“探测-组播”的机制来进行。当一个结点加入、离开、或者改变其状态时，会有某结点探测到该事件，并且启动一个组播过程将该事件告知所有需要知道该时间的结点（也就是那些 Peer List 中包含了变化结点的指针的结点）。

通过 PeerWindow，每个 Granary 中的结点都拥有一大批指向其他结点的指针：一个结点可以仅以 1kbps 带宽的开销就收集到超过 1000 个结点的指针。而且，结点可以自行决定它的级别，并在运行时动态调整。关于 PeerWindow 的详细算法可参见[21]。

Granary 的结点将几类不同的信息附加到结点指针中，用以进行各种决策。包括 GNP[16]坐标[34]（用于在选择结点时预测结点距离）、操作系统信息（用于选择操作系统不同的结点进行复制）、负载能力和当前负载状态（用于负载平衡）等等。

自适应的分布式哈希表路由协议 Tourist（4 级标题）

Tourist 为 Granary 的分布式哈希表路由协议，具有以下三个特性：1）异构性，充分利用所有用户的可用带宽，与已有协议[40][39][47][52]相比，具有更高的路由效率；2）路由效率随结点的稳定性变化，结点越稳定，路由效率越高；3）自然的激励机制，结点的路由表越大，其发出的消息路由越快。

Tourist 的基本构建是两个对称的 PeerWindow：一个常规的 PeerWindow 保存在地址空间中邻近的结点、一个反转的 PeerWindow（使用后串匹配代替标准 PeerWindow 中的前串匹配）保存地址空间中远处的结点。即使在 1,000,000 结点、平均生存周期 1 小时的系统中，Tourist 仍可以保证所有路由在 2 步之内完成。由于继承了 PeerWindow 的异构性，不同的结点路由表大小不同。强结点保存了较大的路由表，所以具有更高的路由效率；而弱结点只保存了较小的路由表，路由效率也较低。当系统规模不大，或者结点较稳定时，Tourist 自然演化成全连接的结构。SmartBoa[20]是 Tourist 的初期设计版本，为反转的 PeerWindow 与叶集合（leaf set）相结合的结构，Tourist 较 SmartBoa 更高效，也更美观。关于 Tourist 的详细算法可参见[21]。

索引管理算法 PB-link Tree（4 级标题）

Granary 为对象数据建立索引，以提高查询速度。由于存储索引和实现查询需要很大的存储空间和 CPU 资源，所以需要一些临近结点相互协作来完成这一工作。当一个 Class Node 缺乏足够的

[16] Global Network Positioning

资源时，它将会寻找一些临近结点，共同完成索引的存储与管理。索引的组织算法为 PB-link 树。PB-link 树是针对广域网分布式环境改进的 B^+树算法。PB-link 树将 B^+树的顶节点和中间节点存储在 Class Node 上，将叶子节点分布到其他合作结点，称“助理结点”（assistant）。分布时根据索引项指向的对象的 objectId 的哈希结果（hashId）进行数据分布（而不是根据属性值进行简单分布）。

PB-link 树的优点在于在进行多属性联合查询时，不用进行大规模的中间结果传输，因为对于某个对象，其所有属性的索引项必然存储在同一个助理结点上（由其 hashId 决定），这样可以大幅度减少查询引起的网络开销。关于 PB-link Tree 的详细算法可参见[27]。

AnySee 视频组播系统——华中科技大学（3 级标题）

AnySee 是华中科技大学自主设计研发的视频直播系统。它采用一对多的服务模式，支持部分 NAT 和防火墙的穿越，提高了视频直播系统的可扩展性；同时，它利用近播原则、分域调度的思想，使用 Landmark[29][51]路标算法直接建树的方式构建应用层上的组播树，克服了 ESM[17]等一对多模式系统由连接图的构造和维护带来的负载影响。与其他类似系统相比，AnySee 具有如下优点：

- 采用一对多的模式，系统中的每个结点既是服务的提供者，也是服务的接收者，提高了系统的可扩展性；
- 处理结点异构性。服务于不同操作系统、不同硬件性能、不同带宽的结点；
- 利用 Landmark 路标算法实现近播思想，降低了系统负载和复杂度，提高了 QoS[18]和系统的可用性；
- 解决了某些结点因为 NAT 或者防火墙而无法享受服务的问题，扩大了系统的可扩展性。

AnySee 的体系结构（4 级标题）

AnySee 是一个基于对等计算的视频直播服务系统。系统由五个部分组成：节目源结点 Broadcaster、为数众多的对等结点 Peer、编码服务器 Encoder Server、黄页服务器 YPS(Yellow Page Server)和历史信息记录服务器 LS(Log Server)。如图 5 所示。

[17] End System Multicast
[18] quality-of-service

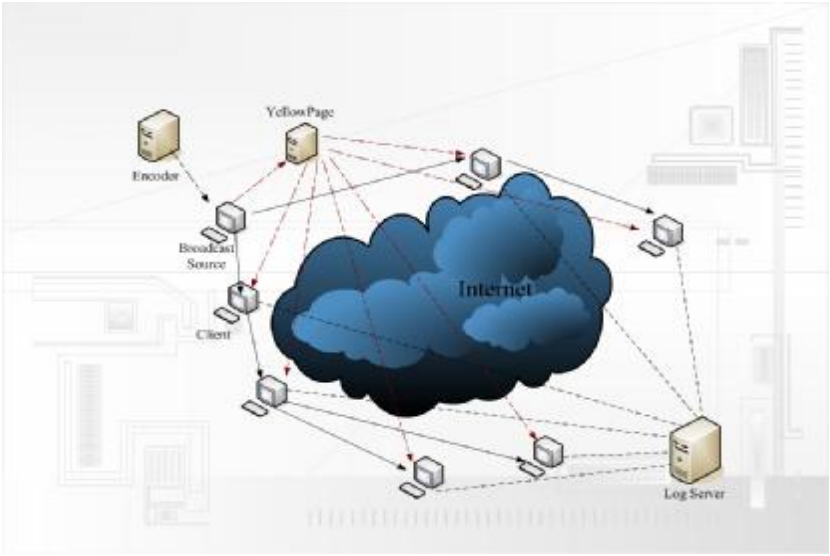

图 5. AnySee 系统结构图

在 AnySee 系统中，所有对等结点连接成一个以 Broadcaster 为根的树。每一个 Peer 接收组播树中父结点提供的媒体数据，同时将这些数据转发给其子结点。Broadcaster 与所有的 Peer 构成一个完整的对等计算系统。Broadcaster 需要使用编码服务器，将捕获的视频信号编码为合适的流媒体格式，以供直播使用；当视频来源为存储在磁盘上的媒体文件时，则不需要编码服务器。黄页服务器 YPS 收集已经由 Broadcaster 发布的直播频道信息，如频道名称、频道介绍、在线人数等等。同时将这些频道信息告知给每一个加入 AnySee 系统的 Peer。历史信息记录服务器 LS 收集在 AnySee 系统中每一个 Peer 的活动状态信息，这些信息对整个系统的性能分析有着非常重要的作用。

可以看到，Broadcaster 只是 Peer 中一个比较特殊的结点。AnySee 的树状结构保证了媒体数据的一次路由，减少了网络负载，其对等计算体系结构保证了系统的可扩展性、高可用性特点。

AnySee 的关键技术（4 级标题）

下面讨论 AnySee 系统实现的三个关键技术：应用层组播（Application-Level Multicast，ALM）、Landmark 路标算法，以及数据缓冲区管理机制。

ALM（5 级标题）

AnySee 采用应用层组播协议。同时，为了保证系统的 QoS，要求提供服务结点与被服务者之间要有良好的网络带宽。基于这个目的，AnySee 使用近播原则、分域调度的思想，用 Landmark 路标算法将物理地址相近的 Peer 尽量调度到一起。比如，在组播树中已经存在的结点集合为{A, B, C}，其中 A 在武汉华中科技大学，B 在北京清华大学，C 在武汉大学，这时在武汉华中科技大学的 Peer D 申请加入对等计算网络，D 将会被调度到 A 的分支上，这样就保证了 A 与 D 之间的高带宽。

AnySee 的网络结构是一棵应用层组播树，树中父结点直接服务子结点。在对等计算网络中，每一个 Peer 的行为都难以预知，因此必须对组播树进行动态维护。

Landmark 路标路由算法（5 级标题）

Landmark 路标路由算法用于实现近播策略。Landmark 是一个长度为 56 位数据类型的值，使用其中的固定几位分别表示国家、网络类型、地区、城市等等。每个 Peer 根据比较树中各个 Peer 的 Landmark 值完成路由调度，最终选择最好，也就是物理位置相隔最近的上层 Peer 做为服务的提供者。

数据缓冲区管理机制（5 级标题）

在任何一个流媒体服务系统中，流媒体的服务质量会受网络状态变化的影响。特别是在 AnySee 的对等计算网络中，由于各个 Peer 的动作难以预见，组播树的维护过程会暂时性地影响媒体数据的持续传输。为了平滑媒体数据的传输，保证系统的 QoS，AnySee 的缓冲区管理机制如下。

- 每个 Peer 的数据缓冲区以时间间隔为单位进行管理。
- 缓冲区缓存最近 40 秒长度内的媒体数据。
- 缓冲区的占用空间的大小随时间动态变化。
- 头部元数据包和媒体数据包分别进行管理。
- 缓冲区内的数据长度达到一定的阈值的时候才能开始向下层 Peer 传送。

系统应用情况介绍（4 级标题）

在 2004 年奥运会期间，系统使用了 CPU 为 1.7GHz Pentium III、内存为 KingMax 256MB DDR、操作系统为 Windows 2000 Professional 的 PC 机做为 Broadcaster；CPU 为 1.4GHz Pentium III、内存 KingMax 256MB DDR、操作系统为 Windows 2000 Professional 的 PC 机做为黄页服务器；CPU 为 1.4GHz Pentium III、内存为 KingMax 256MB DDR、操作系统为 Windows 2000 Professional 的 PC 机做为信息记录服务器，对 CCTV-5 的奥运转播向教育网进行了 20×24 小时的直播服务。共有 10 万人次使用，根据日志显示，教育网中使用过 AnySee 的高校共计 20 多个，北至清华大学，南到中山大学。

2004 年 11 月 11 日，武汉市电子政务网络试用 AnySee，分布于武汉三镇的 896 个社区使用了 AnySee 收看政务节目，试用效果良好。

总结（2 级标题）

本文概括性的介绍了当前国际对等计算研究的发展状况和当前的研究热点问题。概括地说，对等计算系统要解决的问题为资源放置、资源定位、资源获取三个方面。针对这三个方面，我们分别介绍了覆盖网路由协议、对等计算数据搜索技术和应用层组播算法。同时，本文也着力介绍了我国研究人员在这三个方面进行的研究工作和开发的实际系统。事实上，P2P 研究领域的问题还很多，比如安全问题、隐私保护、负载均衡等等，本文不再做更详细的讨论。总的说来，在对等计算领域的研究我国研究人员由于起步早、着力深，在国际研究界具有一定的地位，近年来也在如 IPTPS、ICPP 等一流国际会议上发表了重要学术论文。对等计算系统的构建与研究如今尚处于起步阶段，相信今后几年会有更多更好的对等计算系统出现，研究界也会产生更多的优秀成果。

（配郑玮民照片）**郑纬民** 清华大学计算机科学与技术系教授，博士生导师，清华大学计算机科学与技术系高性能计算研究所所长。研究方向：分布式系统、集群系统、P2P 系统、并行计算等。现任中国计算机学会副理事长。

参考文献

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

[2] http://bitconjurer.org/BitTorrent/. December 2004.

[3] Burton H. Bloom. "Space/time trade-offs in hash coding with allowable errors". *Communications of the ACM*, 13(7):422–426, 1970.

[4] John W. Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. "Informed Content Delivery Across Adaptive Overlay Networks". In Proceedings of ACM SIGCOMM, August 2002.

[5] Luis F. Cabrera, Michael B. Jones, and Marvin Theimer, "Herald: Achieving a global event notification service," in *HotOS VIII*, Schloss Elmau, Germany, May 2001.

[6] R. L. Carter and M. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proceedings of IEEE INFOCOM*, 1997.

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

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

[9] Yang hua Chu, Sanjay G. Rao, and Hui Zhang, "A case for end system multicast". ACM SIGMETRICS international conference on Measurement and modeling of computer systems (Sigmetrics 2000). June 2000.

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

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

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

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

[14] P.T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," Tech. Rep. DSC ID:2000104, EPFL, January 2001.

[15] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar, "A novel server selection technique for improving the response time of a replicated service". in Proceedings of IEEE INFOCOM, 1998.

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

[17] http://gnutella.wego.com. December 2004.

[18] A. Gupta, B. Liskov, and R. Rodrigues. "One Hop Lookups for Peer-to-Peer Overlays". In Proceedings of the Ninth Workshop on Hot Topics in Operating Systems, Lihue, Hawaii, May 2003.

[19] K. Hanna, N. Natarajan, , and B. N. Levine, "Evaluation of a novel two-step server selection metric," in Proceedings of IEEE ICNP, Nov. 2001.

[20] Jinfeng Hu, Ming Li, Weimin Zheng, Dongsheng Wang, Ning Ning, Haitao Dong. "SmartBoa: Constructing p2p

Overlay Network in the Heterogeneous Internet Using Irregular Routing Tables". 3rd International Workshop on Peer-to-Peer Systems (IPTPS '04). February 2004.

[21] Jinfeng Hu, Ming Li, Hongliang Yu, Haitao Dong, and Weimin Zheng. "PeerWindow: An Efficient, Heterogeneous, and Autonomic Node Collection Protocol". The 2005 International Conference on Parellel Processing (ICPP-05). June 2005. Available at http://hpc.cs.tsinghua.edu.cn/granary/.

[22] Jinfeng Hu, Chunhui Hong, Huanan Zhang, Ming Li, Weimin Zheng, Dongsheng Wang. "Tourist: Utilizing Heterogeneity to Build a Scalable, Efficient, and Apative DHT Routing Protocol". In Submission. Available at http://hpc.cs.tsinghua.edu.cn/granary/.

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

[24] http://www.kazaa.com. December 2004.

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

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

[27] Ming Li, Dongsheng Wang, Weimin Zheng, Jinfeng Hu, and Yongquan Ma. "PB-link tree: a numeric range query algorithm in peer-to-peer architecture". In Chinese. China National Computer Conference (CNCC 2003). November 2003.

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

[29] Y. Liu, X. Liu, L. Xiao, L. M Ni, and X. Zhang, "Location-Aware Topology Matching in P2P Systems". Proceedings of IEEE INFOCOM 2004, Hong Kong, China, March 2004.

[30] Boon Thau Loo, Ryan Huebsch, Ion Stoica, Joseph M. Hellerstein. "The Case for a Hybrid P2P Search Infrastructure". In Proceedings of the 3nd International Workshop on Peer-to-Peer Systems (IPTPS'04), 2004. USA.

[31] Qin Lv, Pei Cao, Edith Cohen, Kai Li and Scott Shenker. "Search and Replication in Unstructured Peer-to-Peer Networks". Proc. ACM ICS 2002, 2002.

[32] http://www.napster.com. December 2004.

[33] NG, E., AND ZHANG, H. "Towards global network positioning". In Proceedings of ACM SIGCOMM Internet Measurement Workshop 2001 (Nov. 2001).

[34] T. S. E. Ng and H. Zhang. "Predicting Internet Network Distance with Coordinates-based Approaches". 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002). June 2002.

[35] T. S. Eugene Ng, Yang-hua Chu, Sanjay G. Rao, Kunwadee Sripanidkulchai, Hui Zhang. "Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems". In IEEE INFOCOM'03, 2003

[36] K. Obraczka and F. Silva, "Network latency metrics for server proximity," in Proceedings of IEEE Globecom, Dec. 2000.

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

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

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

[40] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale

peer-to-peer systems". International Conference on Distributed Systems Platforms (Middleware 2001). November 2001.

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

[42] Clarke, I., Sandberg, O., Wiley, B., and Hong, T.W. "Freenet: A distributed anonymous information storage and retrieval system". Workshop on Design Issues in Anonymity and Unobservability. June 2000.

[43] Saroiu, S., Gummadi, P. K., and Gribble, S. D. "A Measurement Study of Peer-to-Peer File Sharing Systems". In Proceedings of Multimedia Computing and Networking 2002 (MMCN '02). January 2002.

[44] M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, "Selection algorithms for replicated web servers". in Proceedings of the Workshopon Internet Server Performance, 1998.

[45] Shuming Shi, Guangwen Yang, Dingxing Wang, Jin Yu, Shaogang Qu, Ming Chen. "Making Peer-to-Peer Keyword Searching Feasible Using Multi-level Partitioning". In Proceedings of the 3nd International Workshop on Peer-to-Peer Systems (IPTPS'04), 2004. USA.

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

[47] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for Internet applications". Annual conference of the Special Interest Group on Data Communication (SIGCOMM 2001). August 2001.

[48] Chunqiang Tang and Sandhya Dwarkadas. "Peer-to-Peer Information Retrieval in Distributed Hashtable Systems" In USENIX/ACM Symposium on. Networked Systems Design and Implementation (NSDI'04), March 2004.

[49] J. Xu, A. Kumar, and X. Yu. "On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks". IEEE Journal on Selected Areas in Communications, vol 22, no 1, pp. 151-163, Jan 2004

[50] Beverly Yang and Hector Garcia-Molina. "Designing a Super-Peer Network". ICDE'03, 2003.

[51] X. Zhang, Z. Zhang, G. Song, and W. Zhu, "A Construction of Locality-Aware Overlay Network: mOverlay and Its Performance". IEEE Journal on selected areas in communications, Vol. 22, NO. 1, pp. 18-28, Jan., 2004.

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

[53] Weimin Zheng, Jinfeng Hu, Ming Li. "Granary: Architecture of Object Oriented Internet Storage Service". IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-EAST 2004). September 2004.

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