文章编号:1001 - 9081(2005)12 - 2885 - 03

内容寻址网络 CAN 研究

刘蜀豫,李之棠,黎 耀
(华中科技大学 计算机科学与技术学院,湖北 武汉 430074)
(shuyu@hust.edu.cn)

摘 要:新一代结构化可扩展 P2P 系统采用支持分布式哈希表(Distributed Hash Table,DHT)的路由算法。CAN(Content-Addressable Network)在 d 维虚拟坐标空间上利用 DHT 来实现内容定位,具有较好的可扩展性、容错性和完全自组等特点。介绍了 CAN 的原理,重点分析了 CAN 的构建和路由算法,并讨论了 CAN 算法的几种改进策略。

关键词:对等网;分布式哈希表;内容寻址网络;路由;坐标空间
中图分类号: TP393 **文献标识码**:A

Research of content-addressable network

LIU Shu-yu, LI Zhi-tang, LI Yao
(*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan Hubei* 430074)

Abstract: The new generation of scalable P2P systems adopts routing algorithms which support a distributed hash table (DHT) functionality. CAN(Content-Addressable Network) uses DHT to relocate resources on a virtual d-dimensional cartesian coordinate space. CAN is scalable, fault-tolerant and completely self-organizing. The concept of CAN was introduced, the construction and the routing of CAN was described, several improvements for CAN was also discussed.

Key words: P2P network; distributed hash table(DHT); CAN; routing; coordinate space

0 引言

对等网络(Peer-to-Peer Network,P2P)的核心是 P2P 路由算法,算法的优劣直接关系到 P2P 系统的性能和可扩展性。但 P2P 最初设计时在扩展性方面存在重大问题,如早期的 Napster 使用集中目录服务而存在单点故障问题,Gnutella 采用类似 OSPF 路由协议的洪泛搜索机制,不仅造成过多的网络流量,同时可扩展性也较差。为了解决 P2P 系统可扩展性差问题,一些研究工作组提出了新一代支持分布式哈希表(DHT)技术的结构化可扩展 P2P 系统,这是一种采用纯分布式的消息传递机制和根据关键字进行查找的资源定位服务,是目前扩展性最好的 P2P 路由方式之一。此类路由算法主要包括加州大学伯克利分校的 CAN(Content-Addressable Network,内容寻址网络)和 Tapestry,麻省理工学院的 Chord、IRIS,以及微软研究院的 Pastry。它们采用各自的 DHT 算法,而不同的 DHT 算法决定了 P2P 网络不同的逻辑拓扑,比如 CAN 是一个 d 维坐标空间,Chord 是一个环形拓扑,Tapestry 则是一个网状的拓扑。

CAN 是一种用于结构化对等网络 P2P 的分布式哈希查找系统,可以在 Internet 规模的大型对等网络上提供类似哈希表的功能,具有可扩展、容错和完全自组织等特点。

1 CAN 原理分析

CAN 具有较好的可扩展性、容错性和完全自组等特点,它在 d-虚拟坐标空间上利用 DHT 来实现资源定位。整个虚拟坐标空间在系统中的所有结点间动态地划分,以保证系统中的每个结点都拥有坐标空间的一个区(zone)。系统中每个数据的关键字(key)根据 DHT 映射到虚拟坐标空间的点 P 上,key 所对应的(key,value)值即存储在 P 所在 zone 的结点上。当结点要发起数据查询时,它先利用 DHT 将查询数据的 key 映射到虚拟坐标空间的点 P 上,查询消息目的地址就是坐标空间中点 P 的坐标,然后,结点再根据自己的路由表将查询消息发送到离 P 最近的邻居上。对于具有 N 个结点的 CAN 系统,每个结点维护 $O(d)$ 的状态,每次搜索的搜索代价是 $O(dN^{1/d})$。

1.1 CAN 的组成

可以把整个 CAN 系统看成一张保存(key,value)对的大哈希表。CAN 的基本操作包括插入、查找和删除(key,value)对。其中 key 是对被搜索资源的关键字(如文件名)哈希后的值,而 value 则是资源的存储位置(如 IP 地址和目录)。整个 CAN 系统由许多独立的结点组成,每个结点保存哈希表的一部分,称之为一个区。此外,每个结点在邻接表中还保存了少量邻接区的信息(邻接区概念在后面会有详细描述)。对指定关键字的插入(或者查找、删除)请求被中间的 CAN 结点路由到区里含有该关键字的 CAN 结点。CAN 有如下的特点:CAN 的设计是完全分布式的,不需要任何形式的中央控制点;CAN 具有很好的可扩展性,结点只需要维护少量的控制状态而且状态数量与系统中的结点数量无关;CAN 支持容错特性,结点可以绕过错误结点进行路由。

CAN 基于虚拟的 d 维笛卡儿坐标空间实现其数据的组织和查找功能,它将整个坐标空间动态地分配给系统中的所有结点,每个结点都拥有独立的互不相交的一块区域。图 1

收稿日期:2005 - 06 - 16
基金项目:湖北省自然科学基金资助项目(2004ABA051);国家网络与信息安全保障可持续发展计划资助项目(2004-研 1-917-C-021)
作者简介:刘蜀豫(1973 -),女,四川简阳人,讲师,博士研究生,主要研究方向:计算机网络安全、下一代互联网; 李之棠(1951 -),男,湖北人,教授,博士生导师,主要研究方向:计算机系统结构、计算机网络与信息安全; 黎耀(1978 -),男,湖北人,讲师,博士研究生,主要研究方向:计算机网络安全、下一代互联网.

给出了一个2维的[0,1]×[0,1]的笛卡儿坐标空间划分成五个结点区域的情况。从图1中可以看到,虚拟坐标空间中的每个区被动态地分配给CAN中的某个结点,如坐标空间(0.5-1,0.0-0.5)被分配给结点B。

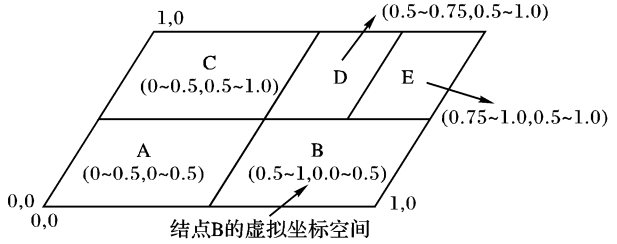

图1 5个结点的笛卡儿坐标空间

虚拟坐标空间采用下面的方法来保存(key, value)对:当要保存(K1, V1)时,首先通过统一的哈希函数把关键字K1映射成坐标空间中的某个点P的坐标,相应的(key, value)对即存放在点P所在区的结点内。当需要查询关键字K1对应的值时,任何结点都可以使用同样的哈希函数找到K1对应的点P,并从点P所在区的结点取出相应的值。如果点P所在区的结点不是发起查询请求的结点或其邻居,则CAN负责将此查询请求路由到点P所在区的结点。因此,在CAN中,有效的路由机制是一个关键问题。

1.2 CAN的路由机制

从图1中还可以观察到,CAN中的路由实际上就是穿过笛卡儿空间从源坐标到目的坐标的一条直线段路径。

在CAN中,每个结点都维护一张坐标路由表,此表用来存放该结点所有邻居的IP地址和虚拟坐标区。在d维坐标空间中,如果两个结点的坐标在d-1维上重叠而在另一维上相邻接,则称这两个结点是邻居关系。例如,图1中结点D和结点E的坐标在y维重叠而在x维相邻接,所以D和E是邻居结点;而D和A在x和y维都是相邻的而没有重叠,所以不是邻居结点。不难看出,对于d维的坐标空间,每个结点有2d个邻居结点,因此每个结点的坐标路由表会维护2d个邻居的IP地址和坐标信息。如2维的坐标空间,每个结点有4个邻居结点,因此每个结点将维护4个邻居结点的IP地址和坐标空间。因为每条CAN消息都包含目的坐标,结点在路由时会将该消息转发给坐标路由表中距目的坐标最近的结点,直到目的坐标。图2给出了一个路由例子,虚线是点1到点(x,y)的路由,图2中还标明了在结点7加入系统前结点1的邻居结点集为{2,3,4,5}。

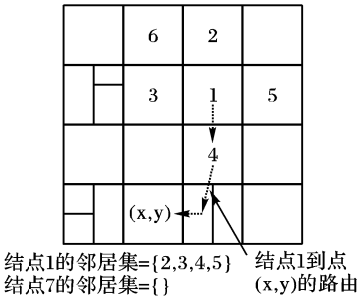

图2 路由例子

对于划分成n个同等大小的d维坐标空间,平均路由长度为$(d/4)(n^{1/d})$跳,每个结点维护$2d$个邻居的信息,这就意味着可以在不增加每个结点状态的同时增加网络结点数,而平均路由长度只以$O(n^{1/d})$的数量级增长。而且,坐标空间中两个结点之间有多条不同路径,如果结点的一个或更多个邻居失效,结点可以自动沿着下一条可用的路径路由。但是,如果某个结点在某个方向失去了所有的邻居,而应用的修复机制还没有来得及在坐标空间中重建空间,那么转发请求将会暂时失败。在这种情况下,结点可以用扩展环搜索来找到一个比自己更接近目的的结点,并将消息转发给该结点进行路由。

1.3 新结点的加入

正如上面描述,整个CAN空间是由当前系统里所有的结点来动态划分的。每当新的结点加入时,系统必须为它分配相应的坐标空间。一般的做法是:系统中某个现有的结点将自己的区域一分为二,自己保留一半,将另一半分配给新的结点。整个过程包括:1)新的结点必须找到一个在CAN中已经存在的结点;2)通过CAN的路由机制,新结点必须找到一个区域将要被分割的结点;3)进行区域划分操作,并通知被分割区域的邻居有新的结点加入以便在邻居的坐标路由表中包含新的结点。

在第一步中,新结点要获得当前CAN系统中某个结点的IP地址,可以用类似YOID一样的自举机制。可以假设该CAN系统被分配了一个域名,该域名可以解析成一个或多个CAN自举结点的IP地址,且每个自举结点维护一张它认为在当前系统中的结点列表。当新结点加入时,首先通过DNS解析CAN域名获得某个自举结点的IP地址,并向该自举结点发出请求,自举结点从自己的结点列表中随机选择几个结点的IP地址响应给新结点。

新结点从收到的响应中随机地选择点P,并向点P发送JOIN请求。该消息通过任何一个现有的CAN结点发送进入CAN,每个CAN结点用CAN路由机制将JOIN消息路由到点P所在区的结点。该结点于是将点P所在的区分割成相等的两部分,并将另一半分配给新结点。分割时按照维的顺序来决定沿着哪个方向划分以便该结点离开时区域能再次合并。比如,对于2维的空间,可以先沿着X轴划分然后再沿着Y轴划分。同时,新划出来区域的(key,value)也移交给新的结点。

获得自己的区后,新的结点可以从被分割区的结点那里学习到邻居集,新结点的邻居集是以前的邻居子集再加上被分割区域的结点。同样,被分割结点从自己的邻居表中删掉不再是邻居的结点。由于重新划分了区域,坐标发生了变化,新结点和被分割区域的结点都要通知它的邻居以修改邻居表。紧接着周期性更新,系统中的每个结点发布立即更新消息,向所有的邻居告知当前分配的区。这种更新保证所有的邻居能快速地学习到变化并更新自己的邻居表。

下面举例描述新结点加入的过程。图2和图3显示了结点7加入一个2维CAN的例子。在结点7加入系统前结点1的邻居结点集为{2,3,4,5}。首先结点7找到要划分区域的结点,即结点1;然后,结点1将自己的区域一分为二,自己保留一半,并将另一半区域分配给结点7;区域重新分割后,结点1的邻居集变为{2,3,4,7},结点7的邻居集为{1,2,4,5}。

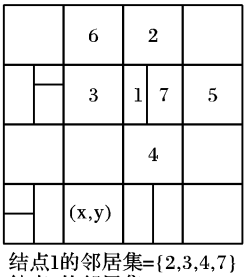

图3 新结点7加入后邻居集的变化

从图2和图3中还可以发现,新结点的加入仅对坐标空

间中少数结点有影响。结点维护的邻居数仅依赖坐标空间的维数而与系统中的所有的结点数无关。因此,新结点的加入仅影响 O(维数) 个现有结点,这个特性对大规模 CAN 网络是很重要的。

1.4 结点离开,恢复和 CAN 维护

当结点离开 CAN 系统时,要保证空出的区能移交给剩下的结点。正常的过程是结点明确地将它的区和相关的(key, value)数据库移交给其中的一个邻居。如果某个邻居的区可以合并该区并产生单个有效的区,那么任务就完成了;如果不行,那么就将该区移交给区最小的邻居结点,该结点将暂时负责两个区。

当出现结点不可达或网络故障时,要求 CAN 必须是健壮的。这可以通过即时接管算法来处理,该算法可以保证由失效结点的邻居来接管失效结点的区。但是离开结点的(key, value)将会丢失,除非数据的拥有者重新更新。

正常情况下,结点周期性地发布更新消息给它的每个邻居,更新消息中包含自己和邻居的区坐标列表。如果某个结点在给定时间内未收到邻居结点的更新消息,则说明该邻居结点失效。一旦某个结点失效,则它的每个邻居结点会分别初始化接管算法并启动接管定时器,定时器的大小和该结点拥有的区大小成比例。当定时器超时,每个邻居结点向其他所有邻居结点发送 TAKEOVER 消息,消息中含有自己的区信息。每当收到一条 TAKEOVER 消息时,每个邻居结点都会做同样的操作:比较自己的区和消息中的区,当消息中的区小于自己的区时,结点会取消自己的定时器,反之则回应自己的 TAKEOVER 消息。用这种方法,可以有效地选择区小的存活邻居结点,由这个邻居结点接管失效结点的区。

在多个邻居结点同时失效的情况下,结点检测到失败是可能的,但只有不到一半的失效结点的邻居仍然可达。如果结点在这些环境下接管另一个区,CAN 状态将可能变成不一致。在这种情况下,在触发修复机制前,结点执行扩展环搜索来搜索失效区域旁的任意结点,因此最终重建充分的邻居状态来安全地初始化接管算法。

可见,不管是结点的正常离开还是失效结点的即时接管算法都会导致一个结点负责多个区。为了防止空间重复分片,后台将运行区重分配算法来保证 CAN 系统中每个结点负责一个区。

2 CAN 改进

前面描述的 CAN 的基本算法提供了低结点状态(d 维空间为 $O(d)$) 和最短路径长度(d 维空间 n 个结点为 $O(dn^{1/d})$ 跳)之间的平衡。这里所说的跳数是应用层的跳数,不是 IP 层的跳数,并且每跳的延迟是真实的。在 CAN 中相邻的结点实际上可能相隔几里或中间经过很多 IP 跳,查找的平均总延迟是 CAN 平均跳数乘以每个 CAN 跳的平均延迟。我们希望能得到一个可以与请求者和拥有关键字的结点之间的真实 IP 延迟相比较的查找延迟。

要减少路由延迟可以通过减少路径长度或减少每跳延迟来实现。可以在 CAN 的基本设计上作些改进,如加上简单的负载均衡机制。

下面分别描述每个改进,并讨论它们对整体性能的影响:在带来了重大的改进的同时也增加了结点状态(结点维护的邻居数)的开销(尽管每个结点的状态和系统中的结点数无关),并增加了算法的复杂度。在使用这些技术时要考虑一方面提高路由性能和系统健壮性,另一方面增加结点状态和系统复杂度之间的平衡。这需要有大量的实际部署经验,并非常了解应用的需求,才能平衡好性能和复杂度之间的关系。

2.1 多维坐标空间

因为 CAN 对坐标空间的维数不作限制,可以通过增加 CAN 坐标空间的维数来减少路由路径长度和路径延迟,但这样做的同时也会少量增加坐标路由表的大小。增加维数意味着每个结点有更多的邻居,在一个或多个结点崩溃的事件中,每个结点会有更多可以用来路由消息的下一跳结点,因此路由容错也得到了改进。

2.2 实体:多个坐标空间

设计多个独立的坐标空间,系统中每个结点在不同的坐标空间中分配不同的区。假设将坐标空间称为“实体”,因此对于有 r 个实体的 CAN,每个结点将被分配 r 个坐标区,每个实体一个,并且维护 r 个独立的邻接表。

这样,哈希表的内容在每个实体上复制,提高了数据的可用性。例如,指向某个文件的指针存在坐标空间(x,y,z)中。对于四个实体的情况,指针会存在每个实体坐标空间(x,y,z)的四个不同的结点中,这样,只有当四个结点都失效时,数据才不可用。多个实体也提高了路由容错,因为在一个实体路由崩溃的情况下,消息可以沿着其他实体进行路由。

此外,因为 hash 表的内容在每个实体间互为备分,到(x,y,z)的路由转换成到任何实体的(x,y,z)的路由。每个实体中结点拥有一个区,每个区的坐标明显不同。因此,结点有能力一跳达到坐标空间中的近的部分,因此可以大大减少平均路径长度。转发消息时,结点检查它在每个实体所有的邻居并将消息转发给坐标上离目的最近的邻居。可见,使用多个实体可以大大减少路径长度,也就是整个 CAN 的路径延迟。

2.3 更好的 CAN 路由 metrics

CAN 中的路由 metric 就是到目的方向的笛卡儿距离。可以通过测量结点通过网络层到达每个邻居的 RTT 时间来改善 metric 以更好地反映底层的 IP 拓扑。假设给定一个目的,消息转发给与 RTT 最大比例的邻居。这不仅带来更低的延迟路径,而且有助于避免应用级 CAN 路由经过不必要的长距离跳数。

与增加维数和实体数不同的是,以 RTT 为 metric 的路由旨在减少沿着路径的每跳的延迟而不是减少路径长度。因此,用每跳延迟作为评估基于 RTT 路由的有效性,而每跳延迟可以用整个路径延迟除以路径长度(跳数)来得到。

2.4 过载的坐标区

CAN 的基本设计是基于这样的假设:在某一刻时间,区只分配给系统中的一个结点。现在假设允许多个结点共享同一个区,称这种现象为区过载并将共享区的结点称为 peers。定义一个系统参数 MAXPEERS,表示每个区可允许的最大 peers 数(我们想象这个值典型比较低,如 3 或 4)。

当区超载时,结点除了维护邻居列表外还要维护 peers 的列表。虽然结点必须知道在自己区里所有的 peers,但它不需要跟踪邻居区里的所有 peers,只需要从 peers 的邻居表中选择一个邻居就可以了。因此,区过载不会增加结点中邻居信息的数量,但要求结点保存 MAXPEERS 个 peer 结点状态。

区过载可以这样完成:当一个新结点 A 加入系统,它找到系统中已经存在的结点 B,并准备占领 B 的区。和前面描述的区域分割不同的是,结点 B 首先检查自己现有的 peer 结点数是否小于 MAXPEERS。如果是,则新结点 A 加入 B 区并不进行任何空间划分,同时 A 从 B 那里获得坐标邻居列表和 peer 列表。A 发布周期性的更新以通知 A 的 peer 和邻居。

(下转第 2891 页)

宽的能力,使得算法的公平性和带宽利用率都有很大提高。

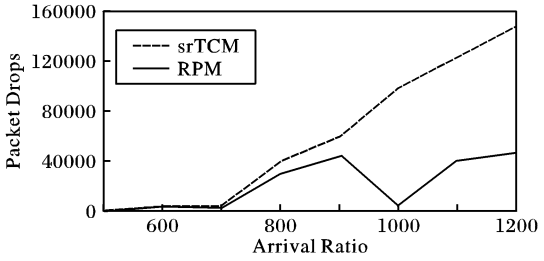

图6　数据包丢失率

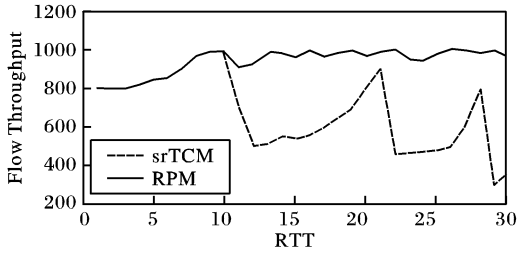

图7　吞吐量

3　结语

提出了一种适用于比例区分服务的数据包标记算法RPM算法,该算法通过预测汇聚流的到达率和带宽估计,能够为DiffServ域提供更好的服务质量。RPM算法将MMSEP和NMSEP两种在线流量预测算法引入计量器,用于估计汇聚流的到达率,将汇聚流的到达率和历史均值进行加权,得出加权后的汇聚流到达率,将此到达率作为带宽分配的依据。通过观测汇聚流的到达率的变化情况,当发现汇聚流进入拥塞恢复阶段时,通过给相关汇聚流分配额外带宽,来缩短汇聚流的拥塞恢复时间,从而提到了吞吐量。通过仿真试验,验证了RPM算法在目标速度的影响,吞吐量等性能方面,与srTCM,trTCM,TSWTCM,ItswTCM这四种标记算法相比具有更好的性能。

RPM算法可被放置于DiffServ核心节点或者DiffServ边界。如果RPM算法被放置于核心节点,那么其标记依据为经过分类器划分,而被划入同一汇聚流的网络流。相关文献指出[9],由多个自相似流复用形成的汇聚流,其统计特征仍然满足自相似特性。因此,我们的预测模型仍然能够对流量到达率做出精确的预测。由于该算法是按照比例区分服务的原则进行带宽分配的,因此并不会引起高级别汇聚流服务质量的下降。

参考文献:

[1] POSTEL J. Transmission Control Protocol. RFC 793[S]. 1981.

[2] JACOBSON V. Congestion Avoidance and Control[J]. ACM Computer Communication Review, 1988, 18(4): 314 - 329.

[3] HEINANEN J, GUERIN R. A single rate three color marker. RFC 2697[S]. 1999.

[4] HEINANEN J, GUERIN R. A tow rate three color marker. RFC 2698[S]. 1999.

[5] CLARK DD, FANG WJ. Explicit allocation of best effort packet delivery service[J]. IEEE/ACM Transactions on Networking, 1998, 6(4): 362 - 373.

[6] SUN H, ATIQUZZAMAN M. ItswTCM: a new aggregate marker to improve fairness in diffserv[A]. Global Telecommunications Conference[C]. 2001.

[7] 王峥. IP网络下的自适应区分服务体系研究[D]. 西北大学, 2005.

[8] JAIN R. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling[M]. New York, NY: John Wiley and Sons Inc., 1991.

[9] PARK K, WILLINGER W. Self-Similar Network Traffic and Performance Evaluation[M]. Wiley InterScience, New York, 1999.

[10] STEVEN B, DAVID B. Architecture for Differentiated Services. RFC2475[S]. 1998.

(上接第2887页)

如果区满了(已经有MAXPEERS个结点),则执行以前的操作,将区划分成两半。结点B通知peer列表中的每个结点,通知它们空间被分割。

区过载带来很多好处:减少路径长度;减少每跳延迟;提高容错,但也有不利的一面,即区过载因为要跟踪peers集而增加了系统复杂度。

2.5　多个Hash函数

为了提高数据的可用性,可以用k个不同的Hash函数将一个关键字映射到坐标空间的k点,并将(key,value)复制到系统中不同的k个结点。这样,只有当k个副本同时失效时数据(key,value)才不可用。同时,查找请求可以同时发送给所有的k个结点,因而减少了查找延迟。当然,这也增加了(key,value)数据库的规模和请求流量。

2.6　均衡分区

新的结点加入时,JOIN消息随机地发送给系统中的某点。因为该点不仅知道自己的区坐标,还知道邻居的区坐标。因此,该点并不直接划分自己的区,而是划分区最大的邻居。这种方法可以使CAN的分区更均衡。

3　结语

目前,基于DHT的P2P系统一般都假定结点具有相同的能力,这对于规模较小的系统较为有效,但并不适合大规模的Internet部署。同时基于DHT的拓扑维护和修复算法也比Gnutella模型和Kazaa模型等无结构的系统要复杂得多。事实上,目前大量实际应用还大都是基于无结构的拓扑和泛洪广播机制,而采用DHT方式的P2P系统大多缺乏在Internet中大规模真实部署的实例,成功应用还比较少见。

在P2P研究领域方面,Sylvia Ratnasamy等人在总结现有的结构化P2P路由算法的基础上提出了结构化对等网络面临的十五个问题,这些问题体现在五个方面:状态效率折中、容错性问题、路由热点问题、地理异构性问题和主机能力异构性问题等。这些都需要做进一步的研究。

参考文献:

[1] Napster[EB/OL]. http://www.napster.com/, 2005.

[2] Gnutella[EB/OL]. http://www.gnutella.com/, 2005.

[3] RATNASAMY S, FRANCIS P, HANDLEY M, *et al.* A Scalable Content-Addressable Network[A]. Proceedings of ACM SIGCOMM 2001[C]. San Diego, California, USA, 2001.

[4] RATNASAMY S, HANDLEY M, KARP R, *et al.* Application-level Multicast using Content-Addressable Networks[A]. Proceedings of NGC[C]. 2001.

[5] RATNASAMY S, SHENKER S, STOICA I. Routing Algorithms for DHTs: Some Open Questions[A]. Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)[C]. Cambridge, MA, USA, 2002.