P2P中一种使用缓存和测量的节点选择模式

傅建明[1,2], 孙芳琪[1], 刘 力[1], 张焕国[1]
(1. 武汉大学计算机学院, 湖北武汉 430072; 2. 武汉大学软件工程国家重点实验室, 湖北武汉 430072)

摘 要: 针对非结构化 P2P中服务节点的选择问题, 利用缓存和测量技术, 提出了测量次数可变的选择模式(VM)和测量次数不变的选择模式(FM). 同时, 给出了 4种缓存更新策略: 随机丢弃, 最大延迟丢弃, 最早时间丢弃和最少使用丢弃. 仿真实验的结果表明, FM 优于 VM, 最大延迟丢弃和最少使用丢弃策略获得了较好的性能, 同时, 该选择模式对节点加入和离开有较好的鲁棒性, 且并发测量可以降低节点选择的时延.

关键词: 对等网; 节点选择; 缓存策略; 并发测量
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2006) 08-1545-04

A Peer Selection Scheme Using Cache and Measurement in P2P

FU Jian-ming[1,2], SUN Fang-qi[1], LIU Li[1], ZHANG Huan-guo[1]
(1. School of Computer Science, Wuhan University, Wuhan, Hubei 430072, China;
2. State Key Lab of Software Engineering, Wuhan University, Wuhan, Hubei 430072, China)

Abstract In order to solve peer selection problem in unstructured P2P, size-Varied Measurement (VM) and size-Fixed Measurement (FM) using cache are provided. Moreover, four caching strategies are investigated: Random Drop (RD), Largest Latency Drop (LLD), Earliest Timestamp Drop (ED) and Least Use Drop (LUD). Finally, our experimental results show the following observations: FM greatly outperforms VM, LLD and LUD are prior to others, our schemes are stable when the rate of topology change is below 5%, and concurrent measurement can sharply reduce the peer selection latency.

Key words peer to peer; peer selection; caching strategy; concurrent measurement

1 引言

近来, Peer-to-Peer(P2P)系统在文件共享和信息搜索等方面得到了越来越多的应用, 如 Gnutella 和 KaZaA 等[1,2]. 一个客户节点在信息搜索时, P2P系统一般会返回满足要求的一组服务节点, 客户节点希望选择其中一个离自己较近的服务节点进行通信, 从而降低通信成本和减少下载时延. 节点选择问题研究如何从一组节点中发现离自己较近(最近)的节点. 该问题同样存在于其他的应用中, 如 ISP(Internet Service Provider)服务簇的服务选择和 CDN (Content Distribution Network)[3]的资源选择, 与一般 P2P不同的是其服务较少变化, 因此, 问题容易解决.

组播, 任播[4]和 IDMaps[5]等技术有助于节点选择问题的解决, 但其实施需要修改网络基础设施. 本文以单播为基础, 研究使用缓存和测量技术的节点选择模式.

2 相关工作

我们首先给出问题的基本描述, 然后介绍相关的研究工作.

在一个 P2P系统中, 其所有节点集设为 H, $A \subset H$ 是服务节点集. 节点 $h \notin A$, 希望从 A 中寻找离自己较近的节点. 节点选择问题可以表示为寻找一个 $p \in A$, 使得 $\forall q \in A, D(p,h) \leqslant D(q,h)$. $D(x,y)$ 表示节点 x 和 y 之间的距离, 一般可以使用跳数, 或时延, 或带宽等衡量. 这些距离值可以通过测量或其他技术获得[6]. 本文使用时延作为距离的度量. 为了评价节点选择问题的各种算法, 我们引入文献[7]采用的 stretch

$$\text{stretch} = \frac{\text{算法计算的最短距离} + \delta}{\text{实际的最短距离} + \delta} \quad (1)$$

δ为时间分辨因子, 为常量, δ越小则分辨率越高. Stretch越大, 则算法越差. 当 Stretch等于 1, 则算法最优, 表明选择的节点是最佳节点.

一种直观的方法是测量 h 与 A 中所有节点的距离, 然后选出其距离最小的节点作为问题的解, 该算法为贪婪算法. 测量次数为 $|A|$, 其 stretch 等于 1. 为了减少测量次数, 简单的近似方法是, 随机从 A 中选取 k 个节点, 然后测量 h

收稿日期: 2004-07-09; 修回日期: 2006-04-30
基金项目: 国家自然科学基金 (No. 90104005; No. 60373089); 湖北省科技攻关计划 (No. 2005AA101C44)

与这 k 个节点之间的距离, 选距离最小的节点作为问题的解. 其测量次数为 k, 其 stretch 可能大于 1.

为了改进节点选择的精度, 提出了基于地标的近似算法. 在 P2P系统中设置一些地标, 即测量点, 用 B 表示所有的地标. A 中节点定期与 B 交换距离值, h 也测量与 B 之间的距离, 然后根据这些值计算基本解空间. 对解空间直接排序, 可直接得到问题的解, 也可以对解空间的部分节点进行测量以获得更精确的解. 典型的算法有 AVG 排序[8], Beaconing[9]和 Binning[10]. Beaconing仅让 B 传送与 h 相近的距离值, 从而减少从 B 传送给 h 的数据量. Binning 把所有测量值放入不同 bin 中, 从而实现相似性计算. 这些方法对地标的放置较敏感.

3 利用缓存策略与测量技术的节点选择模式

我们希望应用缓存和测量技术, 提出一种节点选择模式, 因为测量技术能改良节点选择的质量, 而且缓存能减少测量次数且降低网络负荷.

缓存 R 储存 4元组的距离信息 < peer, timestamp, latency, hit>. peer指要到达的目标节点, Latency指从客户节点到目标节点的网络时延, timestamp为测量时的时间戳, 而 hit为目标节点被选择的次数.

一般缓存大小是受限的, 须采用更新策略. 下面对更新策略的效率给出简单的形式化描述. 假定节点规模为 n, $\eta(i,t)$ 表示在 t时刻的节点 i被选择的比例, c(i)表示到达节点 i 的代价, $\beta(i)$表示节点 i是否在缓存中, 如 $\beta(i)=1$表示在, 否则 i不在. r 表示缓存的长度. 缓存策略的效率如下:

$$\mathrm{Max}\{C=\sum\beta(i)\cdot\eta(i,t)\cdot c(i)\},\ \text{且}\sum_{i=1}^{n}\beta(i)\approx r \quad (2)$$

我们用 $\{\eta(i,t)\cdot c(i)\}$的值来对节点进行排列, 接着按顺序将节点信息写入缓存, 直到缓存写满, 从而获得一种近似的最优解. 该求解的关键是如何选择 $\eta(i,t)$和 c(i), 使得 $\eta(i,t)$表示客户的访问模式, 同时使得到目标节点的网络时延较小.

我们引入四种缓存策略:

(1)随机丢弃 (Random Drop, RD): 随机丢失缓存中的节点记录. 这种策略简单, 不考虑 $\eta(i,t)$和 c(i).

(2)最大延迟丢弃 (LLD): 将缓存中的记录按时延排列, 丢失时延最大的记录. 该策略使用 1/RTT(i)来近似 c(i). RTT(i)是 h 到节点 i 的 RTT(Round Trip Time).

(3)最早时间丢弃(Earliest timestamp Drop, ED): 将缓存中的记录按时间戳排序, 丢失时间戳最早的记录. 这里用 timestamp(i)来近似 c(i).

(4)最少使用丢弃 (Least Use Drop, LUD): 将缓存中的记录按节点被选择次数排序, 丢失访问次数最少的记录. 这里只考虑把 hit近似为 $\eta(i,t)$.

利用上述缓存更新策略, 我们设计了两种节点选择模式. 一种是尽量利用缓存减少测量次数, 这种模式命名为 VM (size-Varied Measurement), 每次被测量的节点数应该小于一个常量; 另一种是每次测量次数不变, 利用缓存提高解的精度, 这种模式命名为 FM (size-Fixed Measurement).

VM 算法如下:

(1)取得服务节点组 S

(2)从 S 中随机选取 k 个节点, $K=\{s_1, s_2, \cdots, s_k\}$

(3)生成集合 $U=K\cap R$

(4)测量 h 与集合 (K-U) 中每一个节点的距离 d_i, $i\in(K-U)$

(5)从 U 和 d_i 中选择距离最短的节点

由于采用了缓存技术, VM 加快了节点选择的速度, 而且其代价变为 $\Theta(|K-U|)$.

FM 算法如下:

(1)取得服务节点组 S

(2)生成集合 $U=S\cap R$

(3)在集合 (S-U) 中随机选取 k 个节点, $K=\{s_1, s_2, \cdots, s_k\}$

(4)测量 h 与集合 K 中每一个节点的距离 d_i, $i\in(S-U)$

(5)从 U 和 d_i 中选择距离最短的节点

其测量代价变为 $\Theta(k)$.

4 性能分析

下面将对我们的节点选择模式作模拟实验, 先给出实验方法, 然后给出分析结果.

4.1 实验方法

选用 GT-ITM[11]生成网络拓扑. 模拟网络有 10000的节点, 随机选取 10% 的节点为 P2P节点, 节点 h 与服务器节点组 S 也是随机选取的. 我们控制 S 的规模在 19到 50之间, 缓存长度 r依次为 50, 100, 150, 测量长度 k 值为 3, 6, 9. 为了获得统计结果, 我们将选取 10个不同的节点 h, 每个 h 测试 1000次.

4.2 实验结果

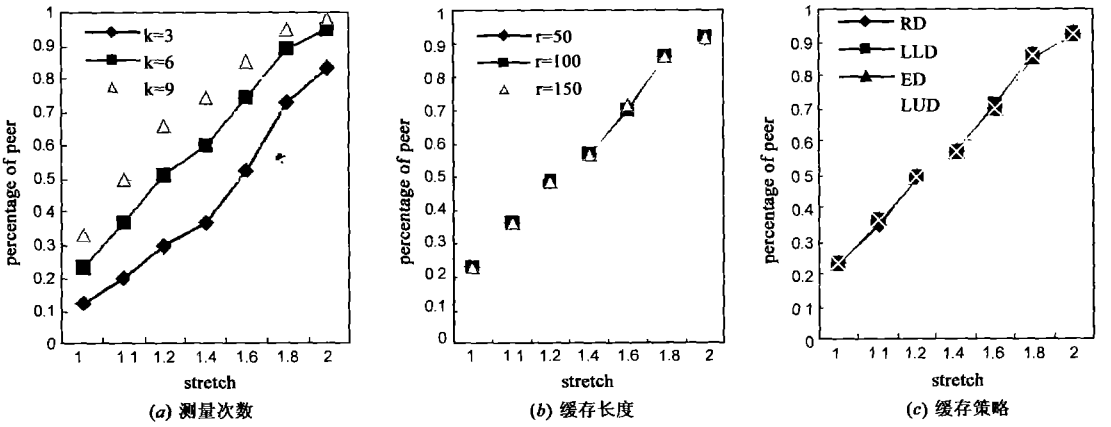
实验的目的是验证缓存长度、缓存更新策略和测量长度三个因素对节点选择模式的影响. 为了验证单因素的作用, 保持一个因素不变, 改变其他两个因素, 最后的统计结果见表 1、表 2以及图 1、图 2.

表 1 VM 性能

因素 / 性能	K			r			缓存策略			
	3	6	9	50	100	150	RD	LLD	ED	LUD
stretch	1.63	1.37	1.25	1.42	1.42	1.42	1.41	1.41	1.42	1.42
hit	0.29	0.58	0.87	0.29	0.58	0.87	0.58	0.58	0.58	0.58

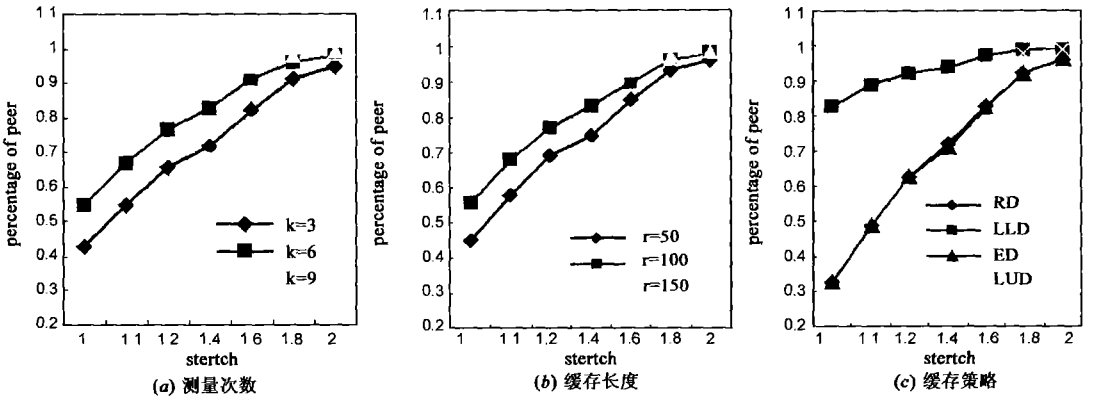
表 2 FM 性能

因素 / 性能	K			r			缓存策略			
	3	6	9	50	100	150	RD	LLD	ED	LUD
stretch	1.27	1.17	1.11	1.24	1.17	1.14	1.28	1.06	1.29	1.11
hit	3.30	3.34	3.36	1.68	3.34	5.00	3.32	3.34	3.33	3.33



(a) 测量次数　(b) 缓存长度　(c) 缓存策略

图 1　VM 的 *stretch* 累积分布



(a) 测量次数　(b) 缓存长度　(c) 缓存策略

图 2　FM 的 stretch 累积分布

hit是指每次请求中缓存命中目标节点的平均值.

表 1反映了 VM 的整体性能. k 值的变化对 stretch和 hit的影响较大. 缓存长度从 50变化到 150时, stretch 保持不变, hit变化较大. 不同缓存策略对 stretch与 hit的影响很小.

表 2反映了 FM 的整体性能. 与表 1相比, FM 比 VM 更好. k 值的变化对 stretch和 hit的影响较轻, 但是 stretch 值较 VM 的要小, 并且 hit接近 3. 3, 比 VM 的要高. 缓存长度变化时, stretch值变化轻微, 但是 hit次数从 1. 68增长为 5. 00. 因此, 缓存长度越大, 算法越好. 总的来说, LLD 和 LUD 是比较好的缓存策略.

图 1和图 2更详细地刻画了 VM 和 FM 关于 stretch的累积分布情况 (Cumulative Distribution Function, CDF).

在图 1中, k = 9时, 33% 的节点 h 可以找到其最近节点, 98% 的节点的 stretch值小于 2 stretch值随 k 值线性增长. 在图 1(b)和 1(c)中, 23% 的节点 h 可以找到其最近节点, 92% 的节点的 stretch值小于 2 但是 stretch值对缓存策略与缓存长度的变化不敏感.

图 2中, k = 9时, 64% 的节点 h 可以找到其最近节点, 99% 的节点的 stretch 值小于 2 在图 2(b)中, r = 150时, 61% 的节点 h 可以找到其最近节点, 98% 的节点的 stretch 值小于 2 在图 2(c)中, 采用 LLD 策略时, 83% 的节点 h可以找到其最近节点, 99% 的节点的 stretch值小于 2 RD 和 ED 的性能几乎相同, 但其性能明显低于其他策略.

4.3　P2P拓扑的动态变化

由于已有节点可能离开 P2P系统, 新的节点加入系统, 这种现象会影响选择模式. 不失一般性, 我们将节点离开和节点加入的比例设为相同, 用 v 表示. 为了简单起见, 我们假定 k 为 6, 缓存长度为 100, 节点选择模式选 FM 算法. 表 3为测试的结果. 当 v 不大于 5% 时, v 的变化对我们的选择模式影响轻微. 这就是说, 选择模式对网络的动态变化具有一定的鲁棒性. 表中缓存策略对应的性能为对不同 v 变化后的平均值, LLD 和 LUD 仍保持良好的性能.

表 3　网络变动对性能的影响

因素 / 性能	v (%)				缓存策略			
	0	1	2	5	RD	LLD	ED	LUD
stretch	1. 17	1. 17	1. 17	1. 18	1. 28	1. 04	1. 28	1. 08
hit	3. 34	3. 30	3. 24	3. 12	3. 30	3. 16	3. 30	3. 25

4.4　并发测量

在并发测量中, 一个低 RTT 会被高 RTT 抑制, 从而可以减少测量的时间. 因此, 我们将在同一时间并发测量所有节点的距离以缩小总的选择时间, 并用 sr(Suppressed

Rate)衡量其性能.

$$sr=\frac{\text{串行测量的总时间}}{\text{并发测量的总时间}} \quad (3)$$

sr越大表明其性能越好. 从 Internet上选了 36个站点, 经过 5次实验, 并每次任选 k个站点. 其实验结果为, 当 k = 3, 6, 9时, sr分别等于 1.63, 3.68和 5.36. sr的值随 k值得增长而迅速增长. k值一定时, k个节点的 RTT差别越小, sr越大.

4.5 选择模式的分析

节点选择问题的求解追求距离较短. 在上述实验中, 服务节点组的选择是随机的, 好像随机丢弃策略适合这种情况, 但实际上随机丢弃的节点可能是距离最短的节点, 背离了节点选择问题的求解宗旨. 同时, 最早时间丢弃与距离无关, 仅仅与进入缓存的先后有关, 也无助于问题的求解. 因此, 这两种策略在实验表现中性能较差.

如果缓存中的节点长时间没有被选中, 表明该节点的距离值较大, 应该被丢弃. 最少使用丢弃近似于最大延迟丢弃, 而最大延迟丢弃直接面向问题求解. 因此最大延迟丢弃和最少使用丢弃都符合问题求解的宗旨, 而且随着时间的推移, 缓存中留下的节点都可能是较好的候选解. 这就是这两种策略表现良好的原因.

VM 算法本身是简单近似方法的扩展. 每次只有 k个节点从服务节点组中选出, 若最优解不在 K 中, 则解的精度明显降低, 此时缓存和测量也无法获得最优解, 缓存的目的仅仅是减少了测量次数. 但 FM 则不同, 先首先获得缓存中的解, 然后从未命中的剩余节点中随机选中 k 个节点测量. 按照最大延迟丢弃和最少使用丢弃的策略, 最优解极有可能在缓存中, 再加上每次测量的次数为 k. 因此, FM 明显优于 VM.

5 结束语

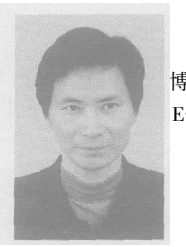
利用缓存策略与测量技术, 提出了 VM 与 FM 的节点选择模式. 我们讨论了四种缓存策略: 随机丢弃, 最大延迟丢弃, 最早时间丢弃和最少使用丢弃. 根据实验结果和分析, FM 优于 VM 且随着测量次数的增大, 性能也随之增强. 对 FM 来说, 最大延迟丢弃与最少使用丢弃策略优于其他策略, 尤其是 LLD 的性能最佳. 当测量次数为 6和缓存长度为 100时, FM 使用 LLD 策略, 80%的节点可以获得其最近节点而且 97%的节点的 stretch值小于 2. 另外, 该选择模式对网络的变动具有鲁棒性, 同时并发测量可以极大地降低节点选择的时延.

事实上, P2P应用存在 free riding现象[12]. 将近 70%的 Gnutella用户不提供任何共享资源, 而将近 50%的网络请求由 1%的共享主机响应. 因此我们的选择模式更适合于现实 P2P系统.

参考文献:

[1] Gnutella[EB/OL]. http://gnutella.wego.com, 2005-01-01.
[2] KaZaA[EB/OL]. http://www.kazaa.com, 2005-01-01.
[3] Cisco CDN[EB/OL]. http://www.cisco.com/go/cdn, 2006-03-25.
[4] D Katabi, J Wroclawski. A Framework for Scalable Global IP-Anycast (GIA) [C]. Stockholm: ACM SIGCOMM, 2000. 3-15.
[5] S Jamin, C Jin, Y Jin, D Raz, Y Shavitt, L Zhang. On the Placement of Internet Instrumentation[C]. Israel: TelAviv, 2000. 295-304.
[6] James D Guyton, Michael F Schwartz. Locating Nearby Copies of Replicated Internet Servers[C]. Cambridge: ACM SIGCOMM, 1995. 288-298.
[7] S Ratnaswamy, M Handley, K Karp, S Shenker. Topologically-aware overlay construction and server selection[A]. INFOCOM[C]. New York: INFOCOM, 2002. 1190-1199.
[8] S Hotz. Routing information organization to support scalable routing with heterogeneous path requirements[D]. California: University of Southern California, 2001.
[9] C Kommareddy, N Shankar, B Bhattacharjee. Finding Close Friends on the Internet[EB/OL]. http://citeseer.ist.psu.edu/cachedpage/508903/1, 2006-03-25.
[10] S Ratnaswamy, M Handley, K Karp, S Shenker. Topologically-aware overlay construction and server selection[A]. INFOCOM[C]. New York: INFOCOM, 2002. 1190-1199.
[11] E Zegura, K Calvert, S Bhattacharjee. How to model an Internetwork[A]. IEEE Infocom[C]. San Francisco, CA, USA: IEEE Infocom, 1996. 594-602.
[12] E Adar, B A Huberman. Free Riding in Gnutella[EB/OL]. Xerox PARC report, www.parc.xerox.com/istl/groups/iea/papers/gnutella, 2000-10.

作者简介:



傅建明 男, 1969年生于湖南宁乡, 副教授, 博士, 主要研究方向为网络安全和 P2P.
E-mail: fujm@public.wh.hb.cn

孙芳琪 女, 1984年生于湖北武汉, 硕士研究生, 主要研究方向为网络安全和 P2P.