P2P中应用平衡机制抑制搭便车行为的研究

刘建辉[1] 王 君[2] 冀常鹏[1] 汪 洋[2]
(辽宁工程技术大学电子与信息工程学院 葫芦岛 125105)[1]
(辽宁工程技术大学研究生学院 葫芦岛 125105)[2]

摘 要 针对近年来P2P网络中存在大量"搭便车"节点的问题,在判断一个节点是否是搭便车节点时提出一种基于平衡机制的算法,其不仅考虑节点自身的理性行为方面,还考虑节点所处的物理环境特征,并采用降低搭便车节点下载资源速度的方法来抑制其搭便车行为。仿真实验表明,该平衡机制算法可以有效降低网络中搭便车节点的数量,并提高网络的下载成功率,同时增强网络的公平性、稳定性,达到抑制搭便车行为的目的。

关键词 P2P网络,搭便车,效益值,公平指标
中图法分类号 TP393 **文献标识码** A

Balanced Algorithm to Suppress Free-riding in P2P Network

LIU Jian-hui[1] WANG Jun[2] JI Chang-peng[1] WANG Yang[2]
(School of Electronics and Information Engineering, Liaoning Technical University, Huludao 125105, China)[1]
(Institute of Graduate, Liaoning Technical University, Huludao 125105, China)[2]

Abstract With the rapid development of P2P network in recent years, there are a lot of 'free-riding' node problems. A kind of balanced algorithm was proposed to determine whether a node is free-riding or not. This algorithm not only takes the rational behavior of node itself into consideration but even also the characteristics of the physical environment where nodes exist. And it slows down the speed of downloading resources of a node to suppress the free-riding behavior. The simulation experiment shows that this algorithm can inhibit the number of free-riding nodes effectively, but also improve the download success rate of the network. It enhances the fairness and stability of network. So finally it achieves the purpose of suppressing the fee-riding.

Keywords P2P network, Free-riding, Effectiveness values, Fairness index

1 引言

从Napster和Gnutella开始,P2P网络系统就已成为Internet吸引数百万用户的不可分割部分。根据大型ISP的最近测量显示,P2P流量已超过Web流量,而Web流量曾经是Internet上的主要流量[1]。P2P系统使得资源共享网络结构从传统的C/S模式过渡到peer-to-peer模式。由于P2P网络具有节点匿名性和资源贡献的自愿性等特征,使得网络中出现一大批的搭便车节点。遗憾的是,整个网络的可扩展性、公平性、稳定性都极大地依赖于节点间的合作共享,而上传其他节点的请求资源会浪费自己节点的内存、带宽等资源,因此许多节点不愿意上传其他节点请求的资源,而只希望享受其他节点提供的资源服务,这便是"搭便车"节点的行为[2]。根据"幂律分布"分析显示,在P2P网络中20.7%的热心节点传输了近90%的P2P流量,这使得整个网络丧失了公平性,且更加重了热心节点的负担。如果这种行为不加限制,不仅会迫使热心节点离开网络,还会导致P2P网络退化为传统的C/S模式,因此迫切需要一种机制来抑制搭便车的行为。

本文提出一种基于平衡机制的抑制搭便车行为的算法。在判断一个节点是否是搭便车节点时,文献[3]通过记录节点转发、提交查询消息的数目来判断一个节点是否是搭便车节点。文献[4]提出的Eigentrust和文献[5]提出的Peertrust是经典的用来解决搭便车的信誉模型,每个节点通过在网络中与其他节点的交互累积信誉值,来判断是否是搭便车节点,同时享受何等的资源服务。上述模型各有优缺点,在判断一个节点是否为搭便车节点时,都只是单纯地从节点自身行为来设计函数,而对于节点所处的网络环境,以及自身的物理配置都未予以考虑。因此基于平衡机制的算法是从综合的角度平衡分析搭便车行为,不仅考虑了节点的自身理性行为,也考虑了节点所处的物理环境,从而能更公平地判定出某节点是否是搭便车节点,因此能更有效地提高网络公平性、稳定性、可扩展性。

2 基于平衡机制算法的研究

2.1 节点的贡献值函数

定义1(节点的收入值) 在某个时刻T_i,若节点P_i参与

到稿日期:2012-09-17 返修日期:2013-01-14
刘建辉(1948—),男,教授,博士生导师,主要研究方向为计算机网络与应用,E-mail:junel016@126.com;王 君(1989—),女,硕士生,主要研究方向为对等网络;冀常鹏 男,教授;汪 洋 男,硕士生。

网络中的资源共享，则节点在 T_i 时刻的收入值函数 $U(P_i,T_i)$ 为：

$$U(P_i,T_i)=\alpha\times\sum_{f_k\in UpList}p(f_k)\times SizeU(f_k)Count(f_k)\times SpeedU(P_i,T_i)+\sigma\times OL(P_i)-\beta\times\sum_{f_k\in DownList}p(f_k)\times SizeD(f_k)\times SpeedD(P_i,T_i) \quad (1)$$

式中，$p(f_k)$ 是指 f_k 文件的流行度，$SizeU(f_k)$ 指上传的 f_k 文件的大小，$Count(f_k)$ 指上传 f_k 文件的次数，亦即是该文件从 P_i 节点被请求的次数。$OL(P_i)$ 指 P_i 节点的在线时长，*UpList* 是 P_i 节点在 T_i 时刻被请求的待上传文件的列表。$SizeD(f_k)$ 是指 P_i 节点下载的 f_k 文件大小，此处下载的 f_k 文件的次数记为 1，因为对于 P_i 节点来说，某 f_k 文件下载一次就可终身受用，*DownList* 是 P_i 节点在 T_i 时刻从系统中等待要下载的文件列表，$SpeedU(P_i,T_i)$，$SpeedD(P_i,T_i)$ 分别是上传、下载文件时的网络传输速度。α,σ,β 是系数，分别是上传收入因子、在线时长因子和下载代价因子，$\alpha+\sigma+\beta=1$，且 $\alpha,\sigma,\beta\in(0,1)$，可以依据对不同因素的着重程度设置各因子的值，这里视 3 者同等重要，故各取 1/3。从以上节点的收入值函数可以看出当节点上传资源时，会增加收入值；当节点从网络中下载资源时，会扣除收入值。同时，收入值不仅与上传/下载文件的大小、次数有关，也与文件流行度、节点在线时长以及上传/下载时的网络传输速度有关，文件越流行，上传速度越快，节点的收入值越高，反之亦然。

定义 2（节点的衰减值） 节点 P_i 在时刻 T_i 的衰减值函数 $\varphi(P_i,T_i)$ 为：

$$\varphi(P_i,T_i)=\gamma\sum_{f_k\in WaitList}p(f_k)\times SizeU(f_k)Count(f_k) \quad (2)$$

当网络中有多个节点向 P_i 请求资源时，P_i 应尽量满足所有的请求。若 P_i 不能及时满足所有的请求，则应扣除 P_i 节点一定的收入值。式(2)中的 f_k 是 P_i 节点所有未能满足的请求文件列表 *WaitList* 中的文件，$p(f_k)$ 是指 f_k 文件的流行度，$SizeU(f_k)$ 指待上传的 f_k 文件的大小，$Count(f_k)$ 指 f_k 文件从 P_i 节点被请求的次数，γ 是衰减因子，且 $\gamma\in(0,1)$。对于 γ 值的设定，应综合考虑。若设置过大，则对物理性能不够优越的热心节点（例如所处的网络比较拥堵）会有大量的请求文件排队等候下载，节点的衰减值会增加，所以会迫使此类节点退出网络。但是若 γ 值设置过小，则对物理性能优越但不愿共享资源的节点惩罚过轻，这也不是应采取的措施，因此应权衡地选择 γ 值，本文取 $\gamma=0.1$。

定义 3（节点的贡献值） 节点 P_i 在时刻 T_i 的贡献值函数 $D(P_i,T_i)$ 为：

$$D(P_i,T_i)=k\times D(P_i,T_{i-1})+U(P_i,T_i)-\varphi(P_i,T_i) \quad (3)$$

即节点的贡献值与节点的收入值和节点的衰减值相关。收入值越大，贡献值越大；衰减值越大，贡献值越小。式(3)中 k 是 T_{i-1} 时刻贡献值的衰减因子，$k\in(0,1)$，本文取 $k=0.9$，这一方面督促节点经常提供良好的服务，才能把贡献值维持在一定范围内；另一方面也抵制了部分恶意节点积累一定的贡献值后转变为搭便车节点对网络进行攻击。

2.2 考虑物理性能下的平衡机制

当在确定一个节点是否是搭便车节点时，不能只观察一个节点的自身理性行为，还要考虑节点所处的物理环境特征。现今移动终端种类越来越多，笔记本、平板电脑、手机等不同类型的终端都可以随时随地联入互联网。对于不同的终端，配置也是不同的，内存大小、CPU 性能、网络带宽、硬盘容量等都会影响到节点在网络中参与资源共享时的行为，所以本文采用基于物理性能下的综合平衡机制算法来判定一个节点是否是搭便车节点。

定义 4（节点的物理性能值） 节点 P_i 在时刻 T_i 的物理性能值 $PP(P_i,T_i)$ 为：

$$PP(P_i,T_i)=\mu\times MS(P_i)+\theta\times CF(P_i)+\omega\times BS(P_i,T_i)+\eta\times HC(P_i) \quad (4)$$

式(4)中考虑的物理性能值包括的因素有节点的内存大小(Memory Size，MS)、CPU 主频(CPU Frequency，CF)、网络带宽(Band Size，BS)、节点在系统中供共享使用的硬盘容量(Hard disk Capacity，HC)。这里的 μ,θ,ω,η 分别是各个因素权重因子，$\mu+\theta+\omega+\eta=1$，且 $\mu,\theta,\omega,\eta\in(0,1)$。可以按照对不同因素的着重程度设置各自的权值。

2.3 节点的效益值函数

定义 5（节点的效益值） 节点 P_i 在时刻 T_i 的效益值函数 $ES(P_i,T_i)$ 为：

$$ES(P_i,T_i)=\frac{D(P_i,T_i)}{PP(P_i,T_i)} \quad (5)$$

从式(5)中看出，节点的效益值函数是节点的贡献值与节点的物理性能值的比值。这样，物理性能低，但尽力为系统做贡献的节点有较大的优势，但对于物理性能值高的节点，其需要更大的贡献值才能使效益值不至于太低，因此本算法可以充分利用物理性能优越的节点的物理资源，同时也符合“同情弱者”的思想。

对于搭便车节点的判定，我们按节点的效益值来界定，若一个节点的效益值小于阈值，则界定此节点为搭便车节点，否则是非搭便车节点。通过对考虑物理性能的平衡机制算法的研究，表明判定一个节点是否为搭便车节点，要依据综合性能来判断，既要考虑节点自身的理性行为特征，又要考虑节点自身的配置情况以及所处的网络环境等物理特性。

3 搭便车节点的惩罚机制

对于搭便车节点的惩罚机制，以往的策略中有两种方法。①降低查询请求的 TTL 值。这种方法可以迅速降低搭便车节点的查询请求 TTL 值，从而使得搭便车节点的查询命中率降低，进而减少因搭便车节点产生的过多的网络通信流量。②对搭便车节点发出的查询请求不予响应[6]。对于已经界定是搭便车的节点，其所发出的一切查询请求均不予响应，只有通过赚取足够的收益值转为非搭便车节点后才能享受正常的网络服务。

对于以上两种方法，都是在搭便车节点的查询请求上做文章，要么减少查询请求的 TTL 值，要么直接不响应其查询请求。这两种方式，都会迫使搭便车节点离开网络，这与延长用户的在线时长，增大网络规模是相悖而行的。因此，本文使用限制搭便车节点的下载速度方法来惩罚搭便车节点，这同时也会延长搭便车节点的在线时长，从而减少节点频繁地加入离开网络，增大网络规模，增强网络稳定性。

4 仿真实验

因为网络中存在搭便车节点和非搭便车节点，所以非搭

便车节点要承载搭便车节点提出的大量查询请求，这对非搭便车节点是不公平的，因此为了衡量整个系统的公平与否，引出系统的公平指标 FI(Fairness Index)来衡量整个系统的公平程度[7]。首先定义节点的上传下载率 X_i(X_i=上传的资源量/下载的资源量)，由于搭便车节点上传给其他节点的资源量极少，因此设定其上传资源量为 0，即 $X_i=0$。对于非搭便车节点，设定其上传下载率 $X_i=c$，即为一个常数 c。则系统的公平指标 FI 定义为：

$$FI=\frac{(\sum_{i=1}^{n}X_i)^2}{n\times\sum_{i=1}^{n}X_i^2} \tag{6}$$

式中，n 是网络中节点的数量，因为系统中存在搭便车节点，所以系统的 FI 不可能为 1。现设整个网络中节点数目为 N，其中搭便车节点的比例为 N_f，则 FI 为：

$$FI=\frac{[N\times N_f\times 0+N(1-N_f)c]^2}{N[(N\times N_f\times 0)+N(1-N_f)c^2]}=1-N_f \tag{7}$$

故现在用 FI 来考察系统的公平指标，由式(7)可以看到整个系统的公平指标受网络中搭便车节点的比例影响。

4.1 仿真环境

仿真使用基于 myeclipse 和 peersim 的模拟器，它可以支持结构化和非结构化 P2P 网络模拟，用来构造 Gnutella 或其他结构的 P2P 网络模型。该仿真器的目的是验证该平衡机制在具体应用中对搭便车者的抑制效果，并评估其性能和价值。

4.2 网络拓扑

本仿真基于 Gnutella 网络，假定网络中有 5000 个节点，即将 Network. size 设为 5000(在仿真过程中暂不支持另有节点加入网络)，并且共享文件的数量为 50000 个。设网络是理想的，任一个节点可以随意地找到所需资源的持有者。系统中的每个节点都被定义为一个线程，每个线程都携带着对应节点所维持的共享文件资源，线程被创建后执行文件上传或下载操作，最后当节点退出网络时撤销掉对应的线程。为了模拟 P2P 网络的运行，随意地赋给每个节点相应的内存大小、CPU 频率、带宽值、共享硬盘容量，在本次仿真中 4 个物理因素的权重因子均设为 0.25。

4.3 仿真模拟

将该文提出的基于平衡机制的算法与文献[8]中提出的只基于收入值(文献[8]中被称作是效用值，utility value)机制的算法和没有任何机制的 P2P 网络联合起来进行模拟试验，并分别从①系统公平指标 FI 值随时间变化，②系统的成功下载率随时间变化，③系统的稳定性随公平指标 FI 值变化 3 个方面对该平衡机制的性能做出评估。

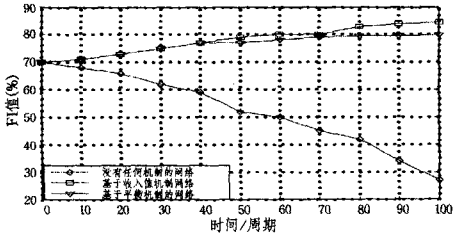

图 1 系统公平指标 FI 值随时间变化曲线图

系统的公平指标 FI 值随时间变化曲线见图 1，由图 1 可以看出，随着时间的增长，没有任何机制的 P2P 网络的 FI 值迅速降低，而平衡机制的和基于收入值机制的 FI 值都是处于上升趋势，因为这两种机制都对搭便车者予以一定的惩罚，因此有效提升了系统的公平指标。通过图 1 可知本文提出的平衡机制相对于基于收入值机制在提升 FI 值方面是稍逊色的。因为平衡机制在衡量一个节点是否是搭便车节点时，不仅看节点自身的理性行为，同时也看该节点所处的物理环境。因此物理性能值较小的节点可以仅凭较小的贡献值就可获取一定的效益值，并且享受到系统中的资源，即允许一定的搭便车节点存在，所以系统的公平指标 FI 值会略小。

没有采用任何机制的系统中，系统的成功下载率是随机变化的，如图 2 所示，但是大致处于降低的趋势。而在有抑制搭便车节点的情况下，系统的成功下载率是有规律可循的，总体来说是提高了系统的下载成功率。由于平衡机制对搭便车节点的抑制是降低其下载资源的速度，而基于收入值机制是每隔一段时间分给搭便车节点一定的收入值(被称为免费赠品，大小为 5.6M)，只有当节点的收入值大于要下载文件的大小时才能执行下载操作，否则只能等待下一次免费赠品的发放。因此与基于收入值机制相比，本文提出的平衡机制在网络交互的初期是略处于劣势的，但是随着网络交互活动的继续，基于平衡机制算法的优越性逐渐明显，系统的成功下载率在后来的时间段内都高于基于收入值机制的算法，并且维持在一个较高的水平。

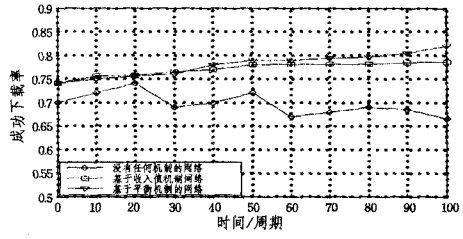

图 2 系统成功下载率随时间变化曲线图

网络系统的稳定性随公平指标 FI 值的变化而变化，从图 3 可以看出随着系统公平指标 FI 的降低，系统的稳定性也在降低。当公平指标 FI 值近乎为 0 时，通过比较 3 条曲线，基于平衡机制系统的稳定性最强，因为该机制对搭便车者的处罚是限制其下载速度，所以搭便车者会通过延长在线时长下载资源。由此系统的离线率会降低，所以因节点频繁地加入离开网络而引起的抖动性影响会减少，故系统的稳定性较好。而基于收入值机制的系统中，搭便车者在等待少量免费赠品的发放过程中会有部分搭便车者离开网络，所以当公平指标 FI 值较小时，系统的稳定性明显小于平衡机制系统。

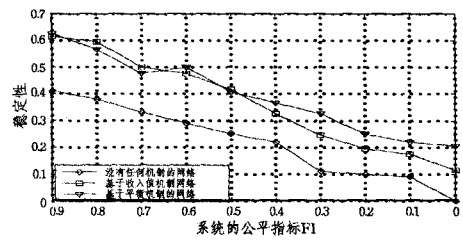

图 3 系统稳定性随公平指标 FI 值变化曲线图

结束语 本文提出的基于平衡机制的抑制搭便车者的算法，在判断一个节点是否是搭便车节点时，从节点自身的理性

行为和所具有的物理特性两方面平衡考虑。这种平衡机制对所有的节点在判断是否是搭便车节点时，更公平合理。同时对于搭便车者的处罚机制，是通过限制其下载资源的速度来实现的，从而可以延长其等待下载资源的在线时长。从仿真结果来看，基于平衡机制的算法可以在一定程度上有效地抑制搭便车行为，提高成功下载率以及系统的稳定性。

参 考 文 献

[1] Steinmetz R, Wehrle K. P2P 系统及其应用[M]. 王玲芳，陈焱，译. 北京：机械工业出版社，2008：1-22

[2] 许晓东，邹宝军，朱士瑞. 信任模型中搭便车节点的抑制[J]. 计算机科学，2012，39(3)：88-92

[3] 康江，房鼎益，陈晓江. 一种无结构 P2P 网络中对抗 Free-rider 的新方法[J]. 小型微型计算机系统，2010，31(8)：1538-1541

[4] Michal F, Christos P, John C, et al. Free Riding and Whitewashing in Peer to Peer Systems[J]. IEEE Journal on Selected Areas in Communications, 2006, 24(5): 1010-1019

[5] Xiong L, Liu L. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(7): 843-857

[6] 余一娇，金海. 对等网络中的搭便车行为分析与抑制机制综述[J]. 计算机学报，2008，31(1)：1-15

[7] Hua J-S, Huang S-M, Yen D C, et al. A dynamic game theory approach to solve the free riding problem in the peer-to-peer networks[J]. Journal of Simulation, 2012(6): 43-55

[8] Lakshmish R, Liu L. Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems[C]// Proceedings of The 36th Hawaii International Conference on System Sciences. IEEE Computer Society, 2003: 1-10

(上接第 23 页)

[17] Byna S, Chen Yong, Sun Xian-he. A Taxonomy of Data Prefetching Mechanisms [J]. Journal of Computer Science and Technology, 2009, 24(3): 405-417

[18] Rui Hou, Zhang Long-bing, Hu Wei-wu. Accelerating sequential programs on Chip Multiprocessors via Dynamic Prefetching Thread[J]. Microprocesors and Microsystems, 2007(31): 200-211

[19] Lu Ji-wei, Das A, Hsu W-C, et al. Dynamic Helper Threaded Prefetching on the Sun UltraSparc CMP Processor[A]// Proc. 38th Ann. IEEE/ACM Int'l Symp. Microarchitecture(MICRO '05), 2005[C]. New York: ACM, 2005: 93-104

[20] Zhang Wei-feng, Calder B, Tullsen D M. A Self-Repairing Prefetcher in an Event-Driven Dynamic Optimization Framework [A]// IEEE Proceedings of the International Symposium on Code Generation and Optimization(CGO'06)[C]. 2006

[21] Luo Yang-chun, Packirisamy V, Hsu Wei-Chung, et al. Dynamic Performance Tuning for Speculative Threads[A]// Proc. of the 36th Int. Symp. on Comp. Arch. (ISCA-09)[C]. 2009

[22] Kim D, Liao S S-W, Wang P H, et al. Physical Experimentation with Prefetching Helper Threads on Intel's Hyper-Threaded Processors[A]// Proceedings of the International Symposium on Code Generation and Optimization[C]. Mar. 2004

[23] Jung C, Lim D, Lee Jaejin, et al. Helper Thread Prefetching for Loosely-Coupled Multiprocessor Systems[A]// IPDPS[C]. 2006

[24] Lee J, Jung C, Lim D, et al. Prefetching with Helper Threads for Loosely Coupled Multiprocessor Systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(9): 1309-1324

[25] Lyle J, Wallace D, Graham J, et al. Unravel: A CASE Tool to Assist Evaluation of High Integrity Software[EB/OL]. http://hissa.ncsl.nist.gov/publications/nistir5691/vol1/, 2012-08-15

[26] Song Yong-hong, Kalogeropulos S, Tirumalai P. Design and Implementation of A Compiler Framework for Helper Threading on Multi-Core Processors[A]// Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT'05)[C]. 2005

[27] Zhang Jian-xun, Gu Zhi-min, et al. Performance Evaluation of data-push Thread on Commercial CMP Platform[A]// Proceedings of the International Network Computing Conference[C]. Korea, 2010

[28] Gu Zhi-min, Zheng Ning-han, Zhang Yi, et al. The Stable Conditions of a Task-Pair with Helper-Thread in CMP[A]// Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, 2009. Las Vegas, Nevada, USA: IEEE, 2009: 125-130

[29] Gu Z, Fu Y, Zheng N, et al. Improving Performance of the Irregular Data Intensive Application with small workload for CMPs [A]// International Conference on Parallel Processing Workshops[C]. Taiwan, China, 2011

[30] Zhang J, Gu Z. Exposing the Shared Cache Behavior of Helper Thread on Commercial CMP Platforms[A]// 11th International Symposium on Pervasive Systems, Algorithms, and Networks. Dalian, China (I-SPAN'2011)[C]. 2011

[31] Huang Y, Tang J, et al. The Performance Optimization of Threaded Prefetching for Linked Data Structures[J]. Intern. Journal of Parallel Programming, 2011, 4(20): 141-163

[32] Huang Y, Gu Z, et al. Reducing cache pollution of threaded prefetching by controlling prefetch distance[A]// Proc. IPDPS[C]. 2012

[33] Zhang J, Gu Z, et al. Solving Prameter Selection Problem of Helper Thread Prefetching via Realtime Hardware Performance Monitoring[A]// 13th International Conference on Parallel and Distrbuted Computing, Applications and Technologies (PDCAT 2012)[C]. 2012