

基于平衡多叉树的 P2P 网络拓扑研究

倪朔东¹, 袁蔚芳², 施 仨¹

(1 南通大学 现代教育技术中心, 江苏 南通 226019; 2 南通大学 工程训练中心, 江苏 南通 226019)

摘 要: 为了进一步提高 P2P 网络的服务质量, 构建性能更为优良的 P2P 网络, 提出了一种新的基于平衡多叉树 P2P 网络(MBTN)拓扑构建方案, 对相关的调度算法进行了研究, 并详细阐述了在该拓扑网络上节点加入和退出算法. 最后, 从节点平均恢复时间、平均延迟时间两方面对 MBTN 拓扑结构组播网络上的 P2P 业务进行仿真试验. 实验表明, 基于 MBTN 仿真出的拓扑网络可以有效减少节点恢复时间和延迟时间, 较好地降低了控制成本.

关键词: P2P; 网络拓扑; 组播树

中图分类号: TP393. 03

文献标识码: A

文章编号: 1000-7180(2013)08-0044-05

Research on P2P Network Topology Based on MBTN

NI Shuo-dong¹, YUAN Wei-fang², SHI Quan¹

(1 Modern Education Technology Center, Nantong University, Nantong 226019, China;

2 Engineering Training Center, Nantong University, Nantong 226019, China)

Abstract: In order to construct more efficient P2P network topology and improve the quality of service of P2P network, a novel topological architecture and scheduling algorithm based on multi-balance tree networks in the application level are provided, which node joining algorithm, node exit algorithm, and so on. Results of simulation experiment show that the method can effectively reduce the average recovery time and delay time and better reduce the control cost.

Key words: P2P; Network topology; Multicast tree

1 引言

目前对 P2P 系统^[1]的研究主要集中在组播树的研究上, 也就是把服务器和所有参加服务的节点组织成组播树, 其中就包括组播树的创建、节点的加入和离开等方面. 按节点组成的拓扑结构划分, 主要包括基于单棵树和基于多棵树的两种数据分发结构. 基于单棵树的结构以 SpreadIt^[2]为代表, 采用 PeerCast 协议将节点组织成单一的组播树, 从根节点开始通过搜寻非饱和节点来完成树的构建, 这种模型的优点是组播树构造简单, 节点可以快速加入, 但是由于对父节点的依赖性特别强, 一旦父节点意外退出, 整个节点的恢复时间就很长. 基于多棵树的

结构以 NICE、ZIGZAG 为代表, 采用基于层次簇思想的组播树构建协议将 Peer 组织成层次式的集群, 按照一定的规则建立组播树, 这两种模型使得节点的加入和退出不会影响到根节点, 减少了节点间的恢复时间, 但是随着组播树的层次升高, 系统的控制开销就将成为瓶颈.

如何有效地管理节点恢复时间和控制传输延迟成为组播树构建的重点, 对于一定量的节点, 如果所生成的组播树是一个平衡树, 那么这棵树中节点的饱和度是比较高的, 且整棵树的高度也是比较低的. 即节点查找的时间复杂度也会降低, 整棵树的控制开销成本也相应变小. 因此, 本文提出了基于平衡多叉树(Multi-Balance Tree Network, MBTN)的 P2P

收稿日期: 2012-08-28; **修回日期:** 2012-10-19

基金项目: 国家自然科学基金(61171132); 江苏省自然科学基金(BK2010280); 江苏省现代教育技术研究 2011 课题(2011-R-18957); 南通市应用计划(BK2011003, BK2012001)

网络拓扑构建方案。

2 基于 MBTN 的 P2P 网络

2.1 MBTN 的拓扑结构和遍历

多叉树是一些节点的有限集合,每个节点的后继节点可以是 0 也可以是多个,在结构上是由一个根和多棵有序不相交的树组成。图 1 中给出的就是一棵平衡多叉树,它是由一个根 A 和三棵子树组成,这三棵子树分别为 $\{B, E, F, G, N, O\}$, $\{C, H, I, J, P, Q\}$, $\{D, K, L, M\}$ 。

如果多叉树中任意两个子树的高度差不大于 1,则称这棵多叉树为平衡多叉树。图 1 中三棵子树的高度分别为 3,3,2,高度差不大于 1; $\{B, E, F, G, N, O\}$ 这棵子树又是由一个根 B 和三棵子树组成,这三棵子树分别为 $\{E\}$, $\{F, N, O\}$, $\{G\}$,它们的高度分别为 1,2,1,高度差同样不大于 1;同样,其他的两棵子树中,高度差也不大于 1,因此整棵树是棵平衡树。

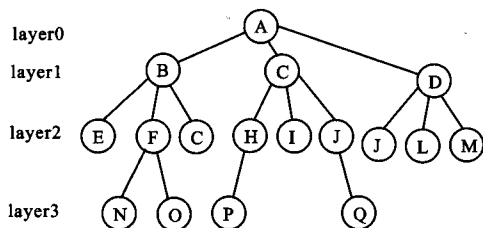


图 1 平衡多叉树的拓扑结构

相对于二叉树,多叉树同样可以采取深度优先和广度优先遍历。对于同样的节点,多叉树的广度将远大于二叉树,而深度将远小于二叉树,因此对多叉树采取深度优先遍历将更利于查找的效率和精确性,而考虑到源服务器即根节点的服务能力,查找将以先序遍历为主。

2.2 MBTN 中节点的路由查询

MBTN 中的每个节点都保存有 4 种邻居关系^[3]:父节点、子节点、兄弟节点以及先序遍历的左右邻居关系,这里把最上层的节点定义为根节点,没有子节点的节点定义为叶子节点,其余的节点称为内部节点,其中父节点、子节点及兄弟关系是所有树结构所固有的关系,先序遍历左右邻居关系是在查找中诞生的关系,使得在 MBTN 上的查找可以迅速地从根节点转移到叶子节点,同样也可以实现各子树间的转移,这样可以相对减轻对上层树节点的资源依赖。

MBTN 中的任何一个节点都可以使用一个二元组 $\langle L, N \rangle$ 来表示,其中 N 为该节点在层中的

位置编号(Node), L 是该节点在树中的层号(Layer)。每层的编号规则为:树根的层号为 0,其余节点的层号是其父节点的层号加 1。每个节点的编号规则为: L 层的节点总数不超过 ML 个(M 为树的叉数,图 1 中 $M=3$),即共有 ML 个位置编号,按从左至右的顺序编号,依次为 $0, 1, \dots, ML-1$ 。由此可见,任意一个节点通过二元组标识 $\langle L, N \rangle$ 中的层号和位置编号来确定其在树中的位置。

上下层节点间的快速跳转可以通过先序遍历很快在树的上层和下层之间进行跳转^[4],而多棵子树间的快速跳转可以通过建立一定的规则,在同层节点间建立远程连接来实现。对于任意一个节点,要求选择的连接既要使得跳转尽量快,又不造成冗余和浪费,只要能够快速跳转到要查找节点的子树上,就可以通过先序遍历顺利地遍历到该节点。因此本文制定这样的同层连接规则,取任一节点其标识为 $\langle L, N \rangle$,它的同层其他子树上的某节点为 $\langle L, N + M * i \rangle$,且 $0 \leq N + M * i < ML$ (i 为非零整数)。实际上,通过加 $M * i$,就可以使得网络中的任意两个节点都处于不同的子树,也达到了快速跳转遍历的目的。如图 1 所示, F 的标识是 $\langle 2, 1 \rangle$, $M=3$,按规则对节点 F 进行同层连接,因为 $0 \leq 1 + 3 * i < 32$,所以 $1 + 3 * i$ 依次等于 4,7,对应的节点分别为 $I \langle 2, 4 \rangle$, $L \langle 2, 7 \rangle$,分别处于第 2,3 棵子树中。

对于任意一个节点 $\langle L, N \rangle$,通过计算 $N + M * i$ 可以得到树中同层其他子树的节点,这里把这些对应的节点称为其左右邻居表,以下简称邻居表。如果所计算出的节点 $\langle L, N + M * i \rangle$ 都存在于该树中,则称其邻居表是满的。图 1 中有一种特殊的情况就是 Layer1 的节点都不存在邻居表,因为它们本身就在一棵子树中,因此把 Layer1 的节点视为互为邻居。

2.3 MBTN 中节点的加入

节点的加入过程分为 2 个步骤:第 1 个步骤是请求加入的节点在树中找到合适位置,第 2 个步骤是加入拓扑树,并更新其相关网络信息。加入的过程设计如下:请求加入的节点 P 根据路由找到离自己最近的节点 A ,把这个处理请求信息的节点 A 称为当前节点,首先看 A 的邻居表,若未滿,则把 P 加进邻居表,作为 A 的邻居加入组播树;若 A 的邻居表已经滿,但子节点尚未滿,则把 P 作为其子节点加入,这时不会影响树的平衡;若子节点已滿则将处理请求转发给后继合适节点。

节点 P 加入的具体算法步骤描述如下:

- (1) 请求加入的节点 P 向 A 发出加入请求;
- (2) 节点 A 收到请求后先计算自己所处组播树的叉数 M ;
- (3) 节点 A 计算自己在组播树中所处的位置 $\langle L, N \rangle$;
- (4) 计算 A 的邻居表 $\langle L, N + M * i \rangle$, 其中 i 为非零整数;
- (5) 若 $\langle L, N + M * i \rangle$ 为空, 则 A 将请求转发至该位置节点的父亲节点;
- (6) 父节点收到请求后将 P 作为自己的子节点插入, P 加入组播树, 跳转至(10);
- (7) 否则计算 A 的子节点节点 $\langle L + 1, N * M + t \rangle$, 其中 $0 \leq t < M$, 为整数;
- (8) 若 $\langle L + 1, N * M + t \rangle$ 为空, 则 A 将 P 作为自己的子节点插入, P 加入组播树, 跳转至(10);
- (9) 否则 A 将请求转发至其后继节点, 后继节点收到请求后跳转至(3);
- (10) P 节点加入组播树后, 更新网络信息。

图 2 所示的是一棵平衡二叉树, 节点 t1 向节点 B 发出请求将其加入, 通过计算 B 的邻居表, 发现已满, 且三个子节点也已经满, 因此 B 把请求发送给其先序遍历的后继节点 E, E 此时邻居表未满, 只有一个邻居 H, 于是它就把 t1 作为其邻居加入网络中。当 t2 请求节点 B 把它加入时, 同样的步骤转发请求到 E, 但是这时 E 的邻居表已满, 但是其子节点还没满, 于是它就把 t2 作为其子节点加入到了网络中, 同样的道理 t3 也作为 E 的子节点加入到了网络中。当 t4 再请求 B 把它加入到网络中时, B 同样把请求发到了节点 E 上, 此时 E 的邻居表和子节点均已经满, 因此它把要求发送给其先序遍历的后继节点 F, 这时 F 的邻居表已满, 但它的子节点还有一个空, 故 t4 就作为其第三个子节点加入到网络中。

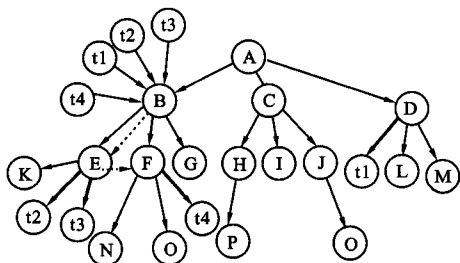


图 2 节点的加入

查找到合适位置加入网络后, 此时需要更新网络信息。以 t1 节点加入为例, 当 D 把新节点 t1 作为其子节点加入后, 它会把自己的一部分资源划分给 t1, 接着向其邻居表中的邻居发出消息, 通知它们 t1

已作为它的子节点加入, 请他们的子节点更新相应的邻居表, 最后还需更新整棵树的先序遍历, D 原来的后继节点是 L, 因此 D 必须通知 L 把 t1 作为其前驱加入。

整个节点的加入过程, 首先满足上层的节点, 且节点加入请求的转发总是在那些邻居表或者子节点未了的节点, 这就极大减少了访问上层节点时的资源负载。每一个节点的加入都是先满足邻居表, 在邻居表已满的条件下再作为子节点加入, 因此整个的节点加入过程, MBTN 一直是保持平衡。

2.4 MBTN 中节点的退出

节点的退出过程同样分为 2 个步骤: 第 1 步是节点直接退出或寻找合适的节点来替换退出节点, 目的是为了使节点退出后保持整棵树的平衡。第 2 步是节点退出拓扑树后更新其相关网络信息, 最底层的叶子节点可以直接请求退出, 完全不影响树的平衡性。若该节点作为叶子节点, 同时有兄弟存在, 理论上它的退出也不影响树的平衡, 但是考虑到 P2P 网络的服务能力和扩容能力, 在本方法中该节点退出后, 会让其兄弟或邻居的子节点, 也就是最底层叶子节点来替换它。

节点退出的过程设计如下: 请求退出的节点 P 作为最底层叶子节点时, 它可以直接退出, 否则就从邻居表中某个邻居的子节点发出替换它的请求。若 P 的邻居都没有子节点存在, 则将替换请求转发给后继合适节点。请求退出的节点 P 作为内部节点时, 先发出替换请求给其后继节点, 当然转发请求尽可能往树的下层节点, 这样使得替换请求尽可能脱离了上层, 减少了对上层的访问负载。

节点 P 退出的具体算法描述如下:

- (1) 节点 P 计算自己所处组播树的叉数 M ;
- (2) 节点 P 计算自己在组播树中所处的位置 $\langle L, N \rangle$;
- (3) 计算 P 的子节点节点 $\langle L + 1, N * M + x \rangle$, 其中 $0 \leq x < M$, 为整数;
- (4) 若 $\langle L + 1, N * M + x \rangle$ 存在, 则将请求转发至 P 的后继节点, 后继节点收到请求后, 跳转至(2); // 判断 P 是否为叶子节点;
- (5) 否则计算 P 的下层节点 $\langle L + 1, y \rangle$, 其中 $0 \leq y < M^{L+1}$, 为整数;
- (6) 若 $\langle L + 1, y \rangle$ 为空, 则删除节点 P, 跳转至 11; // 判断 P 是否为最底层叶子节点;
- (7) 否则 P 将退出请求转发至其邻居节点;
- (8) 邻居节点收到请求后, 计算其子节点 $\langle L +$

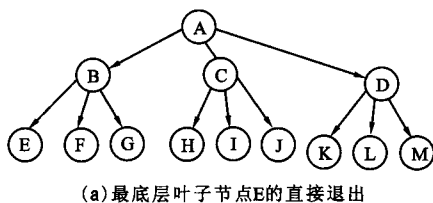
$1, N * M + M^2 * i + t >$, 其中 i 为非零整数, $0 \leq t < M$, 为整数;

(9) 如果 $\langle L+1, N * M + M^2 * i + t \rangle$ 存在, 则孩子节点替代并删除节点 P , 跳转至(11);

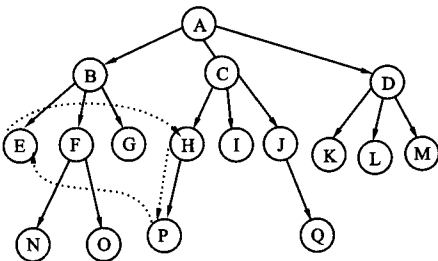
(10) 否则把请求转发至 P 的后继节点, 后继节点收到请求后, 跳转至(2);

(11) P 节点退出组播树后, 更新网络信息.

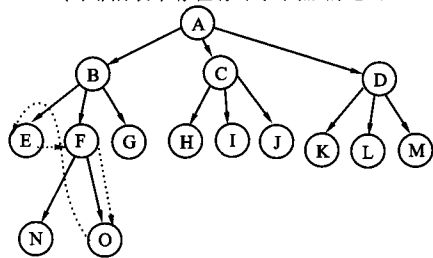
图3所示为MBTN中节点的退出过程, 其中图3(a)中节点 E 是最底层叶子节点, 可以直接退出. 但退出前需通知其父节点 B , 把其负责的所有内容及索引转移给 B , 且还需通知原来的邻居它的退出, 让它们更新关于 E 的信息.



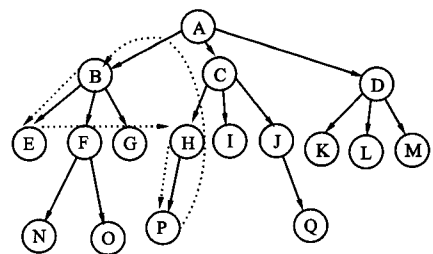
(a) 最底层叶子节点E的直接退出



(b) 邻居表中存在有叶子节点E的退出



(c) 邻居表中不存在有叶子节点E的退出



(d) 内部节点B的退出

图3 MBTN中节点的退出过程

如图3(b)所示, 当叶子节点 E 要退出时, 而此时其邻居表中还有存在子节点的邻居 H , 此时 E 把替换请求发给邻居 H , H 再发给子节点 P , 因为 P 是个叶子节点且是最底层叶子节点, 它可以作为替

换 E 的节点. E 在替换前将其负责的内容以及索引转移到 P 身上, P 在替换前要通知其原来的邻居它的退出, 让它们更新关于 P 的信息.

如图3(c)所示, 节点 E 不是最底层叶子节点, 而通过转发替换请求也没发现其邻居表中有存在子节点的邻居, 此时 E 把替换请求转发给其后继节点 F , F 具有两个子节点, 且都是最底层叶子节点, 可以作为替换 E 的节点, 于是随机选取叶子节点 O 作为替换. E 在替换前将其负责的内容以及索引转移到 O 身上, E 在替换前要通知其原来的邻居它的退出, 让它们更新关于 O 的信息.

如图3(d)所示, B 作为内部节点退出时, 首先找到其后继节点 E , 将查找替换请求转发给它, 此时 E 作为叶子节点退出回到了第二种情况. 而 E 的邻居表中还有存在子节点的邻居 H , 于是 E 把替换请求发给邻居 H , H 再发给子节点 P , 因为 P 是个叶子节点且是最底层叶子节点, 它可以作为替换 B 的节点于是它与 B 联系. B 被替换前将其负责的内容及索引转移到 P 身上, P 在替换前要通知其原来的邻居它的退出, 让它们更新关于 P 的信息. 如果 E 不存在有子节点的邻居, 则回到第三种情况.

最底层叶子节点可以直接退出, 当其他节点退出时, 也都能选择到最底层叶子节点来替换, 较好的保持了整棵树的上层结构, 使得所有的活动脱离上层节点, 降低了网络的负载, 同时能始终保持 MBTN 的平衡性.

3 仿真实验及结果分析

仿真实验环境采用 Windows XP + Cygwin + NS 2.27, 利用 NS2 提供的 NAM 组件来模拟仿真拓扑网络. 在实验中可选择一种前向式树方法^[5] (PAR 算法) 以及后向式树方法^[6] (Reactive 算法) 与本文提出的方法进行仿真模拟分析. PAR 算法通过在失效节点的子节点和叶子节点之间构造备用路由, 从而有效缩短组播树恢复时间, 保证较低的延迟, 但是它的计算量和控制开销比较大; Reactive 算法通过事后受影响节点在父节点和根节点之间构造链路, 有较小的开销但是延迟比较大. 在这两个方面, 本文方法都有优势.

4 结束语

本文提出了一种新的基于平衡多叉树 (MBTN) 的 P2P 网络拓扑结构. MBTN 采用的平衡树结构, 使得树的高度尽可能的降到最低, 节点间跳

转引入了左右邻居表,使得节点间查找时间大大减少,有效的提升了整个网络拓扑的性能.

参考文献:

- [1] Stoica I, Morris R, Karger D, et al. Chord: a scalable peer-to-peer lookup service for internet applications [C] //Proc. of ACM SIGCOMM 2001. San Diego, CA. 2001:149-160.
- [2] Deshpande H, Bawa M, Garcia-molina H. Streaming live media over a peer to peer network[J]. Technical Report Stanford database group technical report, 2001, 3(2):15-23.
- [3] 刘金岭. 基于 P2P 网络的 AVL 索引树范围查询研究[J]. 微电子学与计算机, 2011, 28(2):11-12.
- [4] 毛瑞芳, 李清霞. 一种基于二叉平衡树的 P2P 覆盖网络的研究[J]. 应用科技, 2009, 36(4):40-44.
- [5] Suyong E, Arakawa Shin'ichi A, Masayuki M. Self organizing topology transformation for peer-to-peer networks[J]. IEICE Transactions on Communications, 2010(3):516-524.
- [6] Jonathan S, Omer L, Raphael G. An optimal topology for a static P2P live streaming network with limited resources [C] //Proc. of International Conference on Parallel and Distributed Systems. Tainan, Taiwan, 2011(27):729-734.

作者简介:



倪朔东 男, (1981-), 硕士, 实验师. 研究方向为 P2P 网络、流媒体技术.

袁蔚芳 女, (1982-), 硕士, 讲师. 研究方向为网络与数据库.

施 佳 男, (1973-), 博士, 副教授, 硕士生导师. 研究方向为智能信息处理.

(上接第 43 页)

5 结束语

文中提出了两种 128 点 FFT 结构, 从算法、结构以及功能方面, 对它们各自的优点和缺点进行分析. 我们发现, 在功耗方面 Arch2 要比 Arch1 更有优势, 而在面积方面, Arch1 更好. 两种算法都有规则的数据流, 控制器结构都很简单, 都有利于充分发挥流水线运算单元的处理能力. 两种结构的 FFT IP 核都能满足 UWB-OFDM 系统的要求.

参考文献:

- [1] Aiello G A, Rogerson G D. Ultra-wideband wireless systems[J]. IEEE Microwave Mag June, 2003, 4(2): 36-47.
- [2] Huggett C, Maharatna K. On the implementation of 128-pt fft/fft for high performance WPAN[C]//IEEE International Symposium on Circuits and systems, ISCAS2005. UK: Bristol University, 2005.
- [3] Mathew J, Maharatna K, Jose B R, et al. Pseudo-parallel datapath structure for power optimal implementation of 128-pt FFT/IFFT for WPA-N[J]. Circuits Syst Signal Process, 2011, 30(4):871-882.
- [4] Shen-fu Hsiao, Wei ren Shiue. Design of low cost and high throughput linear arrays for DFT computations: algorithms, architectures, and implementations [J]. Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, 2000, 47(11):1188-1203.
- [5] 程佩青. 数字信号处理教程[M]. 2 版. 北京: 清华大学出版社, 2001.
- [6] Mathew J, Maharatna K, Pradhan D K. A low power 128-Pt implementation of FFT/IFFT for high performance wireless personal area networks[C]//IEEE PRIME Conference. Italy: Otranto, 2006:377-380.

作者简介:

施隆照 男, (1968-), 副教授. 研究方向是数字集成电路设计与系统集成.

高峰 男, (1988-), 硕士研究生. 研究方向为数字集成电路设计.