基于非重叠前缀集合的并行路由查找系统

梁志勇, 徐 恪, 吴建平, 柴云鹏
(清华大学 计算机科学与技术系, 北京 100084)

摘 要: 快速的路由查找机制是高性能路由器设计的关键. 最长匹配查找是路由查找的难点所在. 本文提出一个并行路由查找系统. 它使用一种路由表划分方法, 可将路由表中的前缀划分为若干个集合, 集合内前缀没有重叠. 从而把路由表前缀的最长匹配查找转化为若干个集合内前缀的唯一匹配查找. 基于这种方法, 本文还提出一个通用的并行路由查找框架, 框架适用于大多数路由查找算法. 并行查找框架可简化查找算法的设计, 提高查找算法的速度. 使用二分查找算法, 并行查找系统可以达到 $\log_2(2N/B)$ 的查找复杂度 (N 为路由表前缀数目, B 为大于 4 的整数). 同时, 并行查找系统对 IPv6 也具有很好的扩展性.

关键词: 最长前缀匹配; 二分查找; 路由查找; 路由更新
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2004) 08-1272-05

Parallel Routing Lookup System Based on Non-Overlapping Prefix Sets

LIANG Zhi-yong, XU Ke, WU Jian-ping, CHAI Yun-peng
(Department of computer science and technology, Tsinghua University, Beijing 100084, China)

Abstract: Fast routing lookups are crucial for the forwarding performance of IP routers. Longest prefix match makes routing lookups difficult. This paper proposes a parallel routing lookup system that applies a method to partition a routing table. The method can divide all prefixes in a routing table into several prefix sets where prefixes don't overlap. Based on the method, this paper also presents a common parallel lookup framework that reduces "longest prefix matching" in all the prefixes to "only prefix matching" in several prefix sets. The framework can effectively simplify the design of lookup algorithms and improve lookup performance. The framework is suitable for most lookup algorithms. For simple binary search algorithm, our system can reach $\log_2(2N/B)$ lookup complexity (where N is prefix number in a routing table and B is an integer bigger than 4). Also, the system can be scaled to IPv6 easily.

Key words: longest prefix match; binary search; routing lookup; routing update

1 引言

高性能路由器中, 接口速率越来越高. OC-48(2.5Gbps)、OC-192(10Gbps)、甚至更高速率的接口已经在高性能路由器中得到应用. 在 OC-192 接口速率下, 路由器每秒钟需要转发 3000 万个分组(分组长度按 40 字节计), 进行 3000 万次路由查找. 快速路由查找技术成为提高路由器转发性能的关键.

Internet 初期采用基于类的地址结构, 这导致地址浪费和 Internet 全局路由表指数级增长. 1993 年, IETF 提出无类域间路由(CIDR)[1]的地址结构, 它在一定程度上解决了基于类地址结构的两个问题. 但它使得路由查找变为最长前缀匹配查找, 这增加了路由查找的复杂性. 下一代 Internet 采用 IPv6 地址结构[2], 地址长度从 32bits 扩展为 128bits, 这进一步增加路由查找的难度.

近年来, 研究人员提出了多种机制和算法解决路由查找的问题. 文献[3~8]基于不同的算法结构, 提出了多种路由查找算法. 最长前缀匹配是路由查找的难点所在. 由于前缀之间存在重叠, 传统查找算法(比如: Hash 查找, 二分查找等)不能直接应用于路由查找. 为了减少路由查找的复杂性, 研究人员提出一些方法把最长前缀匹配问题转化为唯一前缀匹配问题. Srinivasan 等在文献[5]中提出一种 leaf pushing 的技术. 二分支 trie 中, 具有路由信息的中间结点的存在导致路由前缀重叠. leaf pushing 技术通过把中间结点的路由信息扩散到叶子结点, 从而消除路由前缀间的重叠. 由于路由前缀不存在重叠, 最长前缀匹配查找转化为唯一前缀匹配查找. leaf pushing 技术可减少路由查找的复杂性, 但也存在两个问题: 1、导致大量重复路由信息. 中间结点的路由信息被扩散到多个叶子结点, 最初中间结点的一条路由扩展为多条路由. 2、更新复杂. 增加或删除中间结点的路由信息, 必须同时增加或删除多条扩展路由, 这大大增加了更新操作的复杂性. Waldvogel 等在文献[4]中提出另一种最长前缀匹配查找的转化方法. 方法按前缀长度把前缀分成若干个集合, 每个集合内的前缀是非重叠的, 集合内的查找为唯一前缀匹配查找. 查找时, 分别对各个集合进行查找, 然后从多个查找结果中, 选择长度最长的前

收稿日期: 2003-08-14; 修回日期: 2004-04-06
基金项目: 863 项目(No.2001AA121013); 教育部科学技术研究重点项目(No.02004); 自然科学基金重点项目(No.90104002); 973 计划项目(No.2003CB314801).

缀做为最终查找结果. 这种方法把最长前缀查找问题转化为若干个唯一前缀查找的问题. Mohammad 等在文献[10]中提出另一种基于出端口的前缀集合划分方法. 方法根据路由信息中出端口的不同把路由前缀划分为若干个集合, 在假设路由器没有连接共享链路的情况下, 集合内部的前缀是非重叠的. 查找过程与文献[4]类似, 先分别对各个集合进行查找, 然后从多个查找结果中, 选择最长的前缀. 由于假设路由器没有连接共享链路, 这种划分方法显然不具有普遍性.

本文的贡献包括三个方面. 第一, 本文提出一个划分前缀集合的方法. 方法基于二分支 trie 树结构把前缀分成若干个集合, 保证每个集合内的前缀没有重叠, 从而把对所有前缀的最长匹配查找转化为对若干个前缀集合的唯一匹配查找. 第二, 基于上面的前缀集合划分方法, 本文提出一个通用的并行路由查找框架, 框架适用于大多数路由查找算法. 框架并行的对多个前缀集合进行查找, 可以有效的提高路由查找算法的查找速度. 第三, 为了提高并行查找效率, 本文还提出一个平衡集合前缀数目的方法. 利用该方法, 各个集合内的前缀数目可以有效的得到平衡. 使用二分查找算法, 并行查找系统可以达 $\log_2(2N/B)$ 的查找复杂度.

2 并行路由查找的基本思想

2.1 二分支 trie 树

二分支 trie 树是一种基本的查找结构, 它具有二叉树结构, 树中每个结点最多有两个分支, 两个分支分别对应 bit 0 和 bit 1. 树中每个结点都代表一个 0/1 的 bit 串, bit 串与根结点到该结点经过的路径对应. 图 1 就是一个简单的二分支 trie 树, 树中存储了 a 到 j 十个前缀. 图 1 中结点的左分支代表 bit 0, 右分支代表 bit 1. 代表前缀的结点用灰色标出, 根结点到灰色结点的路径代表前缀的 bit 串, 比如: 灰色结点 b, 根结点到 b 结点的路径为 /010000, 它就是前缀 b 的 bit 串.

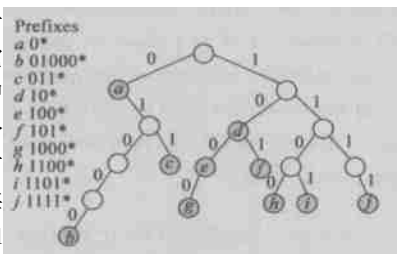

图 1 二分支 Tries 树

在二分支 trie 树中, 代表前缀的结点, 我们称为前缀结点, 如图 1 中所有的灰色结点, 它们存储着有效的路由信息; 不代表前缀的结点, 我们称之为非前缀结点, 如图 1 中所有的白色结点, 它们不具有有效的路由信息, 它们的存在只是满足查找结构的需要.

从二分支 trie 树结构, 可以看出它的两个重要性质:

(1) 一个前缀结点是另一个前缀结点的祖先, 则这两个前缀结点代表的前缀必然是重叠的, 比如图 1 中的 c 结点和其祖先 a 结点.

(2) 两个前缀是重叠的, 则它们对应的前缀结点必是祖先和子孙结点的关系. 比如图 1 中的前缀 d 和前缀 f, 它们代表的地址空间是重叠的. 在二分支 trie 树中, 前缀结点 d 显然是前缀结点 f 的祖先结点.

从上述两个性质可以看出, 前缀结点间祖先和子孙的关系与前缀重叠是等价的. 由此, 我们考虑, 如果把所有前缀结点分成不同的集合, 集合内的前缀结点不存在祖先和子孙的关系, 则集合内的前缀结点代表的前缀也就不存在重叠. 所有前缀也就被划分为几个无重叠前缀的集合.

2.2 非重叠前缀集合的划分

首先我们定义 trie 树中新的结点属性, 结点高度.

定义 1 结点高度 从当前结点到多个子孙叶子结点的多条路径中, 选择一条包含最多前缀结点的路径(如果有多条满足条件的路径, 任选一条), 路径中包含前缀结点的数目(不包括当前结点)称为当前结点的结点高度. 我们用 height 表示结点高度.

图 1 中的 d 结点, 从 d 结点到叶子结点共有 2 条路径: d 到 g、d 到 f. 其中 d 到 g 路径上包含两个前缀结点(e 和 g), d 到 f 的路径只包含一个前缀结点(f), 所以 d 结点的结点高度为 2.

引入结点高度之后, 我们就可以根据结点高度对前缀结点进行划分.

定义 2 高度集合 具有相同高度的前缀结点所代表的路由前缀构成的集合, 称为一个高度集合. 我们用 HeightSet(h)表示 height= h 的前缀结点所代表的前缀组成的高度集合, h 称为高度集合的高度.

按照定义 2, 图 1 中的路由前缀可以分成下面几个高度集合:

HeightSet(0) = {b、c、g、f、h、i、j}, 集合高度为 0;

HeightSet(1)= {a、e}, 集合高度为 1;

HeightSet(2)= {d}, 集合高度为 2;

根据高度集合的定义以及二分支 trie 树的性质, 我们可以推导出下面三个和高度集合相关的定理.

定理 1 高度集合中的所有前缀都是非重叠的.

定理 2 对于某个高度集合, 给定一个 IP 地址, 此高度集合或者存在唯一与该 IP 地址匹配的前缀, 或者不存在与之匹配的前缀.

定理 3 一个 IP 地址与高度集合 HeightSet(i)中前缀 $prefix_i$ 匹配, 同时也与高度集合 HeightSet(j)中的前缀 $prefix_j$ 匹配(i X j), 如果集合高度 i< j, 则 $prefix_i$ 的长度长于 $prefix_j$ 的长度, 否则 $prefix_i$ 的长度短于 $prefix_j$ 的长度[X].

由定理 1 可知, 高度集合中不存在重叠的前缀. 我们把路由表中的所有前缀划分为若干个高度集合, 也就把所有前缀划分为了若干个非重叠前缀集合.

2.3 并行查找

得到无重叠前缀集合后, 进行路由查找时, 我们可以采用并行查找技术, 提高查找速度.

并行查找具有下面几个优势: (1)并行在多个高度集合中进行查找, 可以提高路由查找速度. 使用相同的查找算法, 并

X 以上三个定理, 由于篇幅原因, 本文未给出证明. 如需要, 可从作者主页获得.

行在多个前缀集合中的查找速度高于在一个前缀集合中的查找速度. (2) 高度集合内无重叠前缀, 可简化查找算法设计. 由于无重叠前缀, 高度集合内的查找为唯一匹配前缀查找, 这大大减少了路由查找复杂性, 传统的查找算法可以直接应用(比如: 二分查找法等). (3) 可简化数据结构, 降低查找算法的存储复杂度和更新复杂度. 对于最长匹配, 路由查找算法不得不采用复杂冗余的数据结构, 这导致算法存储空间和更新复杂度的提高. 而对于唯一匹配, 路由查找算法可以使用比较简单的数据结构, 在存储空间和更新速度上都可达到比较好的指标.

3　并行路由查找框架

311　总体结构

为实现并行查找, 我们提出一个并行路由查找框架. 框架总体结构如图 2 所示, 框架分为 4 个部分: 分发器、bank 单元、选择器和 nexthop 映射表.

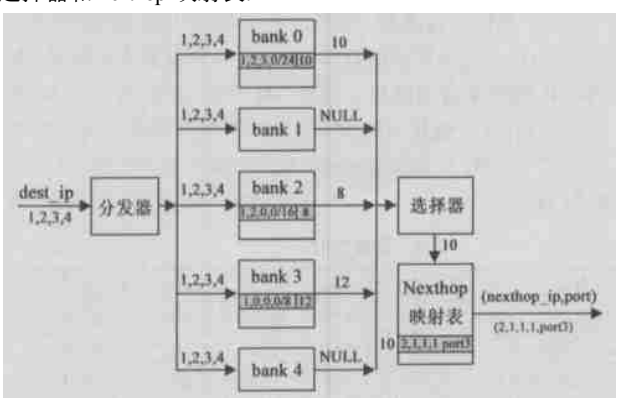

图 2　并行路由查找框架

分发器负责把目的 IP 地址分发至多个 bank 单元, 多个 bank 单元可并行进行查找. Bank 负责在内部前缀集合中进行唯一匹配查找. 每个 bank 存储一个高度集合, 编号小的 bank 对应高度低的高度集合, 比如: bank 0 对应 HeightSet(0), bank 1 对应 HeightSet(1), , , bank 4 对应 HeightSet(4). Bank 单元共有五个, 这是因为实际 Internet 路由表[9]最多可以划分为 5 个高度集合. Bank 的数目也是可以扩充的, 后面会对此进行讨论. Bank 内不存储真正的转发信息(下一跳 IP, 出端口), 转发信息存储在 nexthop 映射表中, bank 只保存转发信息在 nexthop 映射表中的单元索引(index).

选择器负责从多个 bank 的多个匹配结果中选择高度最小集合的匹配结果作为最终的查找结果, 在图 2 框架中, 高度最小集合对应着编号最小的 bank.

nexthop 映射表存储真正的转发信息, 每个映射表单元存储一个(下一跳 IP, 出端口). Nexthop 映射表根据输入的映射表单元索引, 输出对应单元存储的(下一跳 IP、出端口) 转发信息.

具体来讲, 在图 2 中对目的地址 1. 2. 3. 4 进行查找. 首先分发器将 1. 2. 3. 4 分发给五个 bank(bank0 到 bank4). 五个 bank 单元并行进行查找, 其中三个 bank 存在与 1. 2. 3. 4 匹配的前缀(图 2 用灰色条框标出): bank 0 中的 1. 2. 3. 0/ 24, bank 2 中的 1. 2. 0. 0/ 16, bank 3 中的 1. 0. 0. 0/ 8. bank 0 把 1. 2. 3. 0/ 24 对应的映射表索引 10, 做为查找结果送到选择器, bank 2 和 bank 3 分别把映射表索引 8 和 12 送到选择器, 选择器选出编号最低 bank 的查找结果, bank 0 的查找结果 10 被选中. 访问映射表的第 10 个单元, 得到最终的转发信息(2. 1. 1. 1, port3).

312　Bank 内的查找算法

bank 内的前缀不存在重叠, 这大大简化了查找算法的设计. 在后面实验中, 我们采用简单的二分查找算法实现 bank 内的查找.

除了二分查找算法, 其他查找算法也同样适用于并行查找框架, 比如: 多分支 trie 树, Hash 查找, multiway 查找等等. 由于本文主要是讨论并行查找系统, 和这些算法相关内容讨论超出了本文讨论范围, 请参照其他相关文献.

4　并行查找系统的改进

411　Bank 间前缀数目的平衡

并行查找框架中, 选择器必须在所有 bank 都完成查找后才可选出最终的查找结果, 所以多个 bank 并行查找速度取决于查找速度最慢的 bank. bank 间前缀数目分配的不平衡会导致各个 bank 查找速度的不同. 比如, 对于二分查找算法, 查找复杂度为 $\log_2 2N$, bank 内前缀数目 N 直接影响查找速度. 而对于另一些查找算法(如 trie 树, Hash 表), 它们的查找复杂度与前缀数目 N 无关, 但 N 也会间接影响它们的查找速度.

为了平衡 bank 间前缀数目, 我们提出一种前缀平衡方法. 方法利用二分支 trie 树中一类特殊的前缀结点))) 孤儿结点.

定义 3　孤儿结点　如果一个叶子结点, 从它到根结点的路径中, 不存在除它之外其他前缀结点, 则此叶子结点称为孤儿结点.

从定义中可以看出, 孤儿结点是一类特殊的前缀结点, 它的结点高度为 0, 并且所有的祖先结点都不是前缀结点. 图 1 中的叶子结点 h, i, j 都是孤儿结点, 从它们到根结点的路径上不存在除自己外的其他前缀结点.

定理 4　孤儿结点代表的前缀与其他所有前缀非重叠.

由定理 4 可知, 孤儿集合中的前缀与所有前缀都不重叠, 我们可以把孤儿集合中的前缀加入到任何的 bank 内, 都不会影响 bank 内前缀的不重叠关系.

孤儿结点可以很好的平衡 bank 间的前缀差异, 这点我们可在后面的实验中看出.

412　扩展性

图 2 并行查找框架中共有 5 个 bank, 每个 bank 存储一个高度集合. 为了进一步改善并行查找的性能, 我们可以增加 bank 数目, 使用多个 bank 存储一个高度集合.

Bank 数目的增加, 使得 bank 内前缀减少, bank 内查找算法设计变得更为简单. 当 bank 数目等于前缀数目时, 每个 bank 内只存储一个前缀, bank 的查找只需要一次 IP 地址比较. 这时并行查找框架与 TCAM 具有类似的结构. 但对所有前缀的并行比较代价是昂贵的, TCAM 技术的高成本、大功耗、低集成度就是最好的体现. 与 TCAM 技术相比, 我们提出并行

查找框架优势在于, 可以根据需要, 灵活调整 bank 数目, 以满足不同的查找速度、成本、功耗、集成度等指标.

并行查找框架基于 IPv4 地址结构提出, 但它也适用于 IPv6. 对于 IPv6 路由表, 路由数目和路由分布都会有所变化, 因此高度集合的划分可能会有所不同. 为此将并行查找系统应用于 IPv6 时, 需要对存储高度集合的 bank 的数目进行调整.

5 路由更新

并行路由查找框架中, bank 间的前缀存在着约束关系, 在某个 bank 内删除或增加一个前缀, 可能需要在其他 bank 内相应删除或增加一些前缀.

路由更新与路由查找都需要访问路由表, 慢速更新会直接影响查找的性能. 我们采用下面两种技术改善更新性能.

(1) 并行更新

多个 bank 的并行查找必须同时进行, 如果某个 bank 进行前缀更新操作, 其他 bank 则无法进行查找. 依次对不同的 bank 进行前缀更新, 会导致并行查找长时间不能进行. 为了减少 bank 的更新时间, 我们可以把多个对不同 bank 的前缀更新操作并行处理, 同时对多个 bank 进行前缀更新操作.

(2) 缓存更新

由于多个 bank 可以并行执行前缀更新操作, 我们采用缓存技术来提高更新效率. 对于五个 Bank, 每个 bank 维护一个更新操作请求队列. 当每个队列中都有操作请求时, 才并行的对五个 bank 执行一次前缀更新.

6 性能评价

6.1.1 前缀数目的平衡

我们选取一个实际的路由表 mae2west 路由表[13]. 我们先对路由表中的前缀进行划分, 然后我们利用孤儿结点平衡 bank 内前缀.

bank 的数目我们分别选择 B= 5 和 B= 10. B= 5 时, 每个高度集对应一个 bank. B= 10 时, 我们根据高度集合包含前缀数目的比例确定对应 bank 的数目. HeightSet(0) 对应 5 个 bank, HeightSet(1) 对应 2 个 bank, HeightSet(2) 到 HeightSet(4) 分别对应一个 bank.

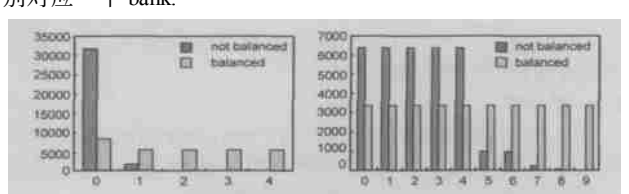

图 3 mae2west 路由表

图 3 黑色线条表示未平衡前 bank 的前缀数目, 灰色线条表示平衡后 bank 的前缀数目. 当 B= 5, 未平衡前 bank 间前缀数目相差很大. bank 0 的前缀数目最多达到几十 K, bank 4 的前缀数目最少只有几个. 平衡后, bank 间前缀数目差异变小, bank 1 到 bank 4 的四个 bank 内前缀数目相当. 但 bank 0 与其他四个 bank 内的前缀数目还是存在一定差异. 这主要是因为 HeightSet(0) 中的前缀数目大于总前缀数目的 1/5, bank 0 内即使不加入孤儿集合的前缀, 它的前缀数目也必然是大于其他几个 bank 的. 当 B= 10 时, 未平衡前存储相同高度集合的多个 bank, 它们包含的前缀数目是基本相当的. 但存储不同高度集合的 bank 之间, 它们包含的前缀数目差异是比较大的. 平衡后, bank 间前缀数目差异基本被消除了, 所有 bank 内的前缀数目都基本相当. 这主要是因为 HeightSet(0) 用 5 个 bank 存储(bank 总数的 1/2). 虽然 HeightSet(0) 内包含的前缀比较多, 但还远远未达到所有前缀的 1/2. 所以利用孤儿集合可以消除所有 bank 内前缀数目的差异.

6.1.2 并行查找框架的查找性能

并行查找框架可以提高路由查找算法的查找速度. 我们对二分查找算法在并行查找框架的查找速度进行了测试. 测试平台为 PC 机, PC 的 CPU 为 PIII 933MHz, 内存为 256M DRAM. 测试使用的路由表为 5 个实际的路由表[9].

为了模拟并行查找, 我们分别对各个 bank 分别进行查找, 然后从中选出最慢的查找时间, 作为并行查找的时间. 查找的 IP 地址集合为随机生成的 500 个 IP 地址. 对每个 IP 地址进行查找时, 查找 1000 次, 然后取平均值算出一次查找时间. 表 1 中的数据为对 500 个 IP 地址查找所需的时间, 时间单位为 ms.

表 1 查找时间

路由表	Mae2east	Mae2west	Aads	Paix	Pacific Bell
路由数目	22443	33953	30716	16880	42348
binary search on intervals	8.231	8.562	8.395	8.162	9.454
Parallel binary search(B= 5)	7.321	7.341	7.330	7.261	7.921
Parallel binary search(B= 10)	6.790	6.800	6.795	6.279	7.331
Parallel binary search(B= 500)	3.710	4.220	3.766	3.705	4.456

对于所有前缀的查找, 由于是最长匹配查找, 不能直接应用二分查找算法, 我们采用文献[3] 中提出的 binary search on intervals 算法. 从表 1 数据可以看出, 并行路由查找框架可以明显改善查找算法的性能. 并且 bank 数目越大, 并行查找速度越快. 当 B= 500, parrallel binary search 与 binary search on intervals 算法相比, 查找速度提高 100% 以上.

7 总结

本文提出了一个并行路由查找系统. 它采用简单有效的方法, 把路由表划分为若干个没有重叠的前缀集合. 从而把路由查找从最长前缀匹配查找转化为若干个唯一前缀匹配查找. 唯一匹配查找可以并行进行, 基于此系统还采用一个通用的并行查找框架. 并行查找框架适用于大多数路由查找算法. 为了提高并行查找系统的查找性能, 我们使用了平衡 bank 前缀数目, 增加 bank 数目的方法. 为了提高并行查找系统的更新性能, 我们采用了并行更新和缓存更新的技术.

对于未来的工作, 我们正考虑在下面两个方面做进一步的研究: (1) 如何用硬件实现并行路由查找系统; (2) 如何在并行路由查找系统中, 实现更有效的路由查找算法.

参考文献:

[1] Fuller V, et al. RFC1519, 1993. Classless inter2domain routing (CIDR): an address assignment and aggregation strategy[S].

[2] Hinden R, Deering S. RFC 2373, 1998. IP version 6 addressing archi2tecture[S].

[3] Lampson B, Srinivasan V, Varghese G. IP lookups using multiway and multicolumn search [J]. IEEE/ACM Transactions on Networking, 1999, 7(3):324- 334.

[4] Waldvogel M, Varghese G, Turner J, Plattner B. Scalable high speed IP routing lookups[A]. Proceedings of SIGCOMM. 97[C]. Cannes, France: ACM, 1997. 25- 36.

[5] Srinivasan V, Varghese G. Fast IP lookups using controlled prefix ex2pansion[J]. ACM Transactions on Computer Systems, 1999, 17(1):1- 40.

[6] SiberCore company. SiberCAM Products[DB/OL]. http://www.siber2core.com/products_siberCAM.htm, 2002.

[7] Desai M, Gupta R, Saxena K, Samant V. Reconfigurable finite2State machine based IP lookup engine for high2Speed router[J]. IEEE Jour2nal on Selected Areas in Communications (JSAC), 2003, 21(4):501 - 512.

[8] Sangireddy R, Somani A K. High2speed IP routing with binary decision diagrams based hardware address lookup engine[J]. IEEE Journal on Selected Areas in Communications (JSAC), 2003, 21(4):513- 521.

[9] Michigan University, Merit Network. Internet Performance and Analysis Project[DB/OL]. http://www.merit.edu/~ipma, 2001.

[10] AkhbarizadehM J, NouraniAn M. An IP packet forwarding technique based on partitioned lookup table[A]. Proceedings of IEEE Internation2al Conference on Communications[C]. New York: Institute of Electrical and Electronics Engineers Inc, 2002. 2263- 2267.

作者简介:

梁志勇　男, 1978 年 11 月生于内蒙古海拉尔, 2003 年毕业于清华大学计算机科学与技术系, 获工学硕士学位, 主要研究方向为计算机网络体系结构, 路由查找算法及其评价. Email: lzy@csnet1.cs.tsinghua.edu.cn.

徐　恪　男, 1974 年 12 月生于江苏省洪泽, 清华大学计算机系博士, 讲师, 主要研究领域为计算机网络体系结构, 高性能路由器体系结构, 分布式实时操作系统, 系统性能评价, 算法分析与设计. Email: xuke@tsinghua.edu.cn.

吴建平　男, 1953 年 10 月生于山西省太原, 清华大学计算机系教授, 博士生导师, 主要研究领域为计算机网络体系结构, 计算机网络协议测试, 形式化技术.

柴云鹏　男, 1983 年 1 月生于内蒙古海拉尔, 现为清华大学计算机系本科生, 主要研究方向为路由查找算法及其评价.