文章编号:1001 - 2486(2005)05 - 0018 - 07

一种基于范围表示 B 树的大容量 IPv6 路由查表算法*

谭明锋,龚正虎,孙志刚
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:IPv6 具有巨大的地址空间,未来要面对的将会是海量 IPv6 路由表,而且 128 位的 IPv6 地址比 IPv4 需要更多的访存数。算法针对 IPv6 路由查找问题中的这两个难点,提出利用 B 树高度较低的优良性质,将前缀转化为范围表保存在 B 树中,并在结点内部利用分段范围比较树算法来减少访存次数和空间耗费。理论分析和实验表明,该算法能够以很好的性能支持 IPv6 海量路由表的查找。

关键词:IPv6;路由查表;B 树;大容量路由表;范围表示
中图分类号:TP393 **文献标识码**:A

An IPv6 Routing Lookup Algorithm for Large Route Tables Based on Range Representation B-tree

TAN Ming-feng, GONG Zheng-hu, SUN Zhi-gang
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The IPv6 routing lookup algorithms need to process huge route tables in the future owing to the huge address space of IPv6, and each lookup needs more memory accesses than IPv4 algorithms because of the 128 bits address. To solve these two difficult problems, this algorithm converts the prefix into ranges and stores them in a B-tree, then uses range fragment tree in the nodes to reduce the memory access and the storage requirement. Theoretical analysis and the experimental results indicate that the algorithm can support the high performance lookup for huge IPv6 route tables.

Key words: IPv6; routing lookup; B-tree; large route table; range representation

IPv6[1] 是未来 Internet 的发展方向,但是对于高性能路由查找,IPv6 比 IPv4 有更大的难度。首先,处理 128 位的 IPv6 地址和前缀所需的访存次数是 IPv4 的数倍。此外,虽然 IPv6 定义了比 IPv4 更合理的寻址结构,但 IPv6 能够"为地球上的每粒沙子都分配一个 IP 地址",因此即使进行了有效的路由聚集,也可以预见 IPv6 路由查找算法未来面对的将是海量路由表。

由于这两个原因,很多 IPv4 路由查表算法和优化技术不太适用于 IPv6 查表。例如,基于 Trie 树[2] 的 IPv4 算法在路由表稍大时,平均访存次数将超过 20 次[3];而对 IPv6,Trie 树的最大高度将远远超过 IPv4 的最大高度 32。而由于 IPv6 共有 128 种可能的前缀长度,前缀扩展等优化技术[4] 并不太适用。

针对上述难点,提出了基于范围表示 B 树的大容量 IPv6 路由查表算法 RR B-tree (Range Representation B-tree)。该算法利用 B 树高度小、更新简单的优秀性质将前缀转化为范围存储在 B 树中,避免了复杂的最长前缀匹配,获得较好的空间复杂度;在 B 树结点内部,用分段范围比较树获得时间和空间复杂度之间的平衡。理论分析和实验表明其在同类算法中具有更好的时空性能。

文中 W 表示 IP 地址长度,L 指机器字长,n 是路由总数,m 为 B 树的阶。

1 相关工作

目前几乎所有已有的 IP 路由查找算法都是针对 IPv4 路由查找而提出的,而其中的绝大多数不适于

* 收稿日期:2005 - 05 - 20
基金项目:国家自然科学基金项目(90104001);国家重点基础研究发展计划项目(2003CB314802);国家 863 高技术研究发展计划基金项目(2003AA115130)
作者简介:谭明锋(1976—),男,博士生。

IPv6,因此这里仅重点介绍几个适用于 IPv6 的算法。

Waldvogel 提出了在前缀长度维度上的二分搜索算法[5],该算法用不同的 Hash 表保存不同长度的前缀和预先计算出的标记,在搜索时按前缀长度和标记在各表中进行二分搜索。在使用理想的 Hash 函数时,其搜索时间复杂度为 $O(\log_2 W)$,更新复杂度为 $O(n\log_2 W)$,空间复杂度为 $O(\log_2 W)$。因此在理想情况下,算法复杂度对地址宽度较不敏感,适用于 IPv6 路由查找。但这些复杂度计算中忽略了如下几个问题:(1)理想的 Hash 函数非常难以获取;(2)需要较大存储空间以保证 Hash 较好的结果;(3)路由数目的增长会增加 Hash 碰撞;(4)算法的每次 Hash 查找需要比较前缀和 IP 地址所有有效位,因此需要多次访存。所以上述复杂度仅是理想复杂度,实际的时间耗费需要乘以一个较大的系数。

Gupta 提出的最差性能受控的近似最优路由查找算法[6]利用 Huffman 编码树的变形算法组织整个路由表,让使用频繁的路由项更靠近树根,使算法的平均搜索性能近似达到二分法算法的最优值,而最差访存次数不大于最优访存次数加 2。其搜索时间复杂度只与路由命中率相关,与 W 无关,因此能够用于 IPv6 路由查表。但该算法存在以下问题:(1)绝大多数的网络设备都不使用或提供路由命中频率的统计功能,以免影响性能;(2)要求路由命中率改变时频繁地重新计算和建立整个数据结构;(3)匹配时,要比较所有有效位,需要较多的访存次数。该算法的理论意义大于实际意义,很少被实际使用。

Suri 提出的多路前缀值范围搜索树算法[7]首先将前缀转化为互不相交的范围表示,然后建立二叉树,并进一步将二叉树扩展成为 k 分支范围匹配树,以减小树高,提高搜索性能。该算法更新性能为 $O(k\log_k n)$,空间性能为 $O(nk\log_k n)$,因此仅与 n 和 k 有关,与 W 无关,能适用于 IPv6。算法搜索树的最小高度为 $\log_k n$,此时用二分法检索结点内 $k-1$ 个范围值的时间复杂度为 $O(\log_2 k)$,搜索时间复杂度为 $O(\log_2(k-1)\times\log_k n)\approx O(\log_2 n)$。但该算法在 n 增大时,时空性能下降较快。

2 算法描述

2.1 B 树的概念

B 树在算法中的作用是利用 B 树的优点,将路由项较为均匀地分散到 B 树的多个分枝上,从而控制树高和对存储空间的要求,并且使由前缀转化成的前缀的端点在 B 树中按照从小到大的顺序存储,以便对 IP 地址进行范围匹配。因此首先介绍 B 树[2]的概念。

定义 阶为 m 的一株 B 树,是满足下列性质的一株树:ⅰ)每个结点最多有 m 个儿子;ⅱ)除了根之外,每个结点的儿子数目至少是 $m/2$;ⅲ)根至少有两个儿子,除非它是一片叶;ⅳ)具有 k 个儿子的非叶结点含有 $k-1$ 个键码。

当一个键码找到相应的 B 树结点准备插入时,若该结点原键码数小于 $m-1$,则直接插入;否则若先挑出 $m-1$ 个原键码和新插入的键码中居中的一个,剩下的 $m-1$ 个键码按大小顺序放在分裂出的两个新结点中,最后将居中的那个键值插入父结点,如果父结点中键码的数目也达到了 $m-1$,那么重复分裂的过程,直到不再分裂结点,或是该分裂过程到达根结点,则将根结点分裂为两个 2 级结点,增加一个新的根作为其父结点,此时树的高度增加了 1。

对 B 树的定义中,并没有说明在 B 树结点的内部应如何对键码进行比较。当结点内的键码数目较多时,该问题的不同解决方法将会导致算法具有不同的性能。

2.2 前缀的范围表示和 B 树更新

算法通过范围比较为一个地址 x 找到小于等于 x 的最大前缀端点,从而找到它的最长前缀匹配,并找到对应的下一跳。首先考虑新插入的前缀 Rd 与已经插入的前缀 Ra、Rb 和 Rc 之间可能存在如图 1 所示的三种关系,设各前缀的端点是 $\mathrm{start}Rx$ 和 $\mathrm{end}Rx$($x=a,b,c,d$)。这三种情况的判断条件和插入 Rc 端点后对路由信息的修改和保存如表 1 所示,因此可得到前缀的范围插入算法 InsertBTree(R):

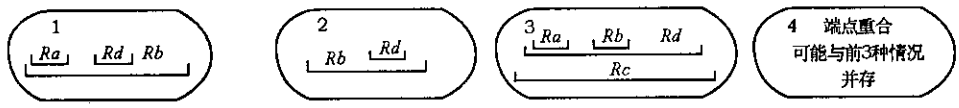

图1 新插入的前缀与旧有前缀之间的关系

Fig.1 The relationship between the newly inserted prefix and the previously inserted prefixes

表1 四种关系的判断以及相应路由信息的修改与保存

Tab.1 The criterion of the four relationships and the corresponding operations on the route information

关系	判断条件	路由信息的修改和保存
1	end*Rd* 的前趋是 start*Rd*，且 start*Rd* 的前趋是某个前缀终点	start*Rd* 和 end*Rd* 都存储 *Rd* 的下一跳信息 end*Rd* 另需拷贝 end*Ra* 中保存的 *Rb* 的下一跳信息
2	end*Rd* 的前趋是 start*Rd*，且 start*Rd* 的前趋是某个前缀起点	start*Rd* 和 end*Rd* 都存储 *Rd* 的下一跳信息 end*Rd* 另需拷贝 start*Rb* 中保存 *Rb* 的下一跳信息
3	end*Rd* 的前趋不是 start*Rd*	start*Rd* 和 end*Rd* 都存储 *Rd* 的下一跳信息 end*Rd* 另需拷贝 end*Rb* 中保存的 *Rd* 的下一跳信息 修改 *Rd* 直接覆盖的所有第一层前缀范围的终点，把原保存的 *Rd* 下一跳信息换为 *Rc* 下一跳信息
4	start*Rd* 或 end*Rd* 与某端点重合	比较 *Rd* 与该端点对应前缀的长度，取最长者

算法描述1 前缀的范围插入算法 InsertBTree(*R*)

B树初始状态 插入了下一跳为空(0)的全0前缀的端点，即全0和全1两个端点。

参数 路由 *R*，包含前缀 pre，下一跳 T

(1)将 *pre* 转化为范围表示 start*R* 和 end*R*。

(2)按照B树的搜索规则找到要插入的结点，按照B树的插入规则将这两个端点插入。

(3)根据表1对路由信息进行处理。

(4)用3.2节中的B树结点内部构造算法 buildNode()构造被修改的结点。

而删除是插入的逆过程，首先找到相应的结点和结点内要删除的端点，按B树规则删除端点，有可能要合并结点，并进行附加的信息修改工作。而相应的路由查找算法如下：

算法描述2 IP地址的路由查找算法 SearchBTree(*A*)

参数 IP地址 *A* 返回 *A* 的下一跳

(1)按B树的搜索规则和结点内查找算法 SearchNode(*A*)找到比 *A* 小的最大端点 *p*。

(2)若 *p* 是起点，则 *A* 的下一跳就是 *p* 自身的下一跳。

(3)若 *p* 是终点，但 *A* 与 *p* 相等，则 *A* 的下一跳就是 *p* 自身的下一跳。

(4)若 *p* 是终点，但 *A* 大于 *p*，则 *A* 的下一跳是 *p* 附加保存的下一跳。

图2给出了一个伪路由表的例子，本文用它来说明算法数据结构的建立和使用。

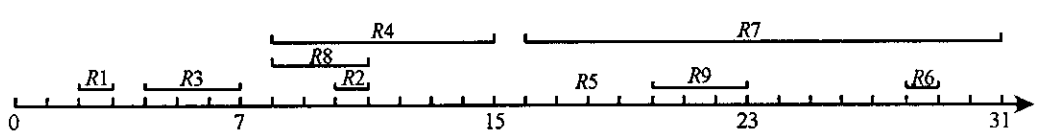

图2 示例路由在数轴上的范围

Fig.2 The example routes showed by the ranges on the axis

图3显示了这些路由插入阶为5的B树中后的宏观数据结构，可以看到由于B树的多分支特性，树高仅为2。其中每个结点中的各个单元中四元组的含义是：<端点值，起点(*s*)/终点(*e*)标志位，本路由

下一跳索引,附加记录的下一跳索引 >,若下一跳索引为 0 则未保存下一跳。这里细实线、粗实线、粗虚线分别为地址 $9(01001)_2$、$11(01011)_2$ 和 $24(11000)_2$ 的查找路径。

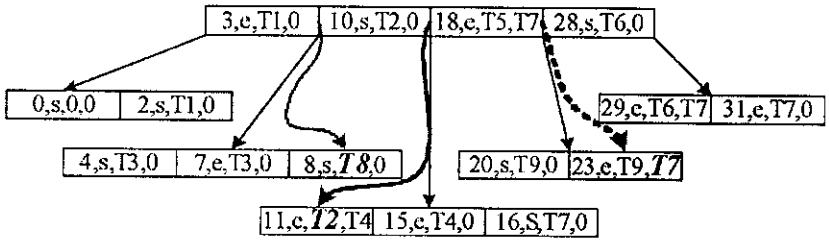

图 3 算法宏观结构和查找过程示例

Fig.3 Overview of *RR* B-tree and examples of the search processes

2.3 B 树结点中数据结构的组织

由于往往不需要比较端点值的全部位即可判断出与目标 IP 地址之间的大小,因此算法采用分段比较来提高 B 树结点内部的搜索性能。

算法描述 3 结点内部构建分段范围比较树算法 buildNodeTree()

(1)将结点中所有 y 个键值分为 W/L 段,令 i 从第 1 段至第 W/L 段。

(2)如果第 i 段刚好有 y 个不同的值,则结束。

(3)如果有多个键码在该段的值相等,则记录这些具有相同段值的键码。

(4)对记录了多个键码的段值,用第 $i+1$ 段的段值再区分,直到每个段值只与一个键码相关联。

(5)重复(2)至(4)步递规建立范围比较树。

这里用图 3 中的根结点和最左结点作为例子,假设 $W=5$,$L=2$,则建立的范围比较树最多有 3 级。实际上,分段范围比较算法能够对前缀信息进行压缩。如图 4 右图所示,该结点第一级中仅保存了 11 这一个值,而第二级则将这两个端点区分开来。因此原来 2 个端点的第 1、2 位共需要 4 位,现在只需要 2 位即可。在路由数目巨大、存储在同一个结点内的端点值的若干高位相同的情况下,该方法所耗费的存储空间甚至比线性存储还要少得多。此外为了能够进行增量更新,在叶结点还需保存那些不作为分枝索引的位。

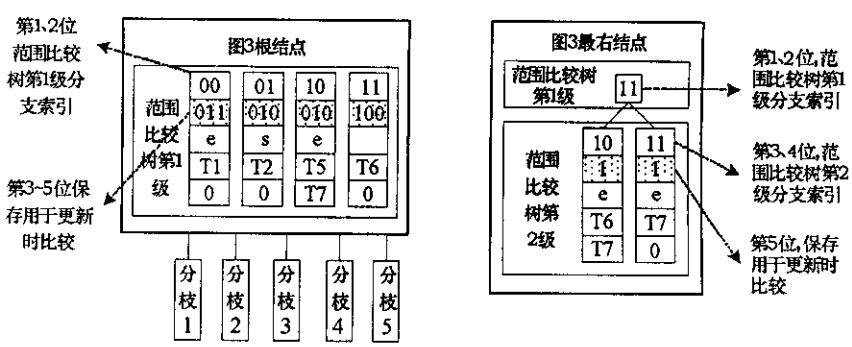

图 4 结点内部构建分段范围比较树内部结构和信息压缩原理示意

Fig.4 The building process and the information compress effect of the range fragment tree in the nodes

3 算法性能分析

设路由数为 $n=2^a$,因此最多有 2^{a+1} 个端点。另设 B 树阶为 $m=2^{b+1}$,因此 B 树除根之外的每个结点中的键值数最多为($2^{b+1}-1$),最少为(2^b-1),子结点数最多为 2^{b+1},最少为 2^b。下面求性能下限。

3.1 *B* 树结构性能

设B树的树高在最差情况下为 $h_{\max}$,此时B树各级的结点的数目为分别为:$2, 2^{b+1}, 2^{2b+1}, \cdots, 2^{(h-1)b+1}$,结点总数

$$\sum_{i=1}^{h_{\max}} 2^{(i-1)b+1} = 2\frac{2^{bh_{\max}}-1}{2^b-1} \tag{1}$$

每个结点内部存储了(2^b-1)个键值,所以根据总键值相等的关系,有

$$2^{a+1} = 2n = 2\frac{2^{bh_{\max}}-1}{2^b-1} \times (2^b-1) = 2(2^{bh_{\max}}-1) \Rightarrow h_{\max} \approx \frac{a}{b} = \frac{\log_2 n}{b} \tag{2}$$

因此最差情况下结点总数为

$$2\frac{2^{bh_{\max}}-1}{2^b-1} = 2\frac{n-1}{m/2-1} = 2\frac{2n-2}{m-2} \approx \frac{4n}{m} = 2^{a-b+2} = 8\frac{n}{m} \tag{3}$$

作为比较,在最好情况下,每个结点有 2^b 个分支,所以最差情况下总结点数为

$$\sum_{i=2}^{h_{\min}} 2^{(b+1)i} = \frac{2^{(b+1)}[2^{(b+1)h_{\min}}-1]}{2^{(b+1)}-1} \approx 2^{(b+1)h_{\min}} \tag{4}$$

每个结点中都存了$(2^{b+1}-1)$个键值,并且同样有

$$2^{a+1} = 2n = 2^{(b+1)h_{\min}}(2^{b+1}-1) \Rightarrow h_{\min} \approx \frac{\log_2 \frac{2^{a+1}}{2^{(b+1)}-1}}{\log_2 2^{(b+1)}} \approx \frac{a-b}{b+1} \tag{5}$$

因此结点数为

$$2^{(b+1)h_{\min}} = 2^{(a-b)} = 2n/m \tag{6}$$

因此结点数是最差情况的 $1/4$。而最大树高和最小树高的差值为

$$h_{\max} - h_{\min} = (a+b^2)/[b(b+1)] \approx 1 + a/[b(b+1)] \tag{7}$$

算法在更新时可能需要修改多个结点,并在最差情况下需要分裂或合并 $h_{\max}$ 个结点,但是实际上平均需要分裂或合并的结点数很少,因为在构造整株树时的分裂总数,恰好是非根结点的总数减去树高 h。如果总共有 p 个结点,则根至少有一个键码,因此至少有 $1+(2^b-1)(p-1)$ 个键码,所以有:

$$2n = 2^{a+1} \geqslant 1 + (2^b-1)(p-1) \Rightarrow p \leqslant 1 + \frac{2^{a+1}-1}{2^b-1} \tag{8}$$

所以分裂的总数小于 $1 + \frac{2^{a+1}-1}{2^b-1} - \frac{a}{b}$,而总共插入了 n 个路由,因此每个路由平均插入时长生的结点分裂数 S 满足下列条件:

$$S \leqslant \frac{1 + \frac{2^{a+1}-1}{2^b-1} - \frac{a}{b}}{n} = \frac{\frac{2n-1}{2^b-1} + \left(1 - \frac{a}{b}\right)}{n} \approx 1/(2^b-1) \tag{9}$$

3.2 *B* 树结点内部性能

假设某次搜索中,各个级别上的键码数分别为 $x_1, x_2, \cdots$,所以在结点内部进行搜索时的访存次数为 $\sum_{i=1}^{W/L} \log_2(x_i)$,且在最差情况下结点内部有 $m-1$ 个键值,因此有 $\sum_{i=1}^{W/L} x_i = m-1$。当所有的 x 相等时,所需要的访存次数最大。令 $W/L = S$,则

$$S \times \log_2[(m-1)/S] = S[\log_2(m-1) - \log_2 S] \approx S[(b+1) - \log_2 S] \tag{10}$$

因此该算法的最差性能比二分法的最好性能 $\log_2(m-1) \times S \approx S(b+1)$ 还要少 $S\log_2 S$ 次访存。特别是对于 $L=32$、$W=128$ 的典型情况,仅在每个结点内就比二分法大约减少至少 8 次访存。

3.3 性能结论

根据上述分析可得到如下结论:ⅰ)从(2)式可看到,算法最差时间复杂度与路由数的对数成正比,且B树的阶 m 较大时比例系数很小;ⅱ)从(3)式可看到,算法的最差性能与前缀数目成正比,而当 m 较大时,比例系数较小;ⅲ)算法进行搜索时,在最好情况下的树高比最差情况下树高少 $1 + a/[b(b+$

1)],例如当路由表大小为 4G,$m = 32$ 时,仅相差两级,算法在最好情况下的空间耗费为在最差情况下的 1/4,而实验表明算法平均性能较为接近最好性能。每次路由更新需要分裂的结点数目少于 $1/(2^b - 1)$,分裂次数很小,更新性能较高。

因此,该算法性能对路由的增长和地址宽度不敏感,适于大容量 IPv6 路由表。而且当 B 树的阶取较为合适的值时,可使算法需要访问的结点数保持在一个很小的水平,从而获得较好的搜索性能。例如选择 $m = 32$,当由表项达到 4G 个之多的时候,算法最多仅需要访问 6 ~ 8 个 B 树结点即可。

4 实验结果

我们对 IPv6 路由表[8]进行了试验。需要指出的是,该 IPv6 路由表是 IPv6 实验网的实验用路由表,因此存在以下问题:(1)前缀数目很少,仅有 893 个,因此不能反映算法在大容量 IPv6 路由表环境中的优势;(2)该路由表分布较为畸形,893 个前缀有 762 个前 16 位为 0x2001,有 128 个为 0x3ffe,而前 32 位为 0x20010250 的就有 109 个。因此其不能准确衡量算法性能,仅具有参考价值。

我们用适用于 IPv6 的范围比较二分法进行比较,这种方法将前缀转化为范围表示,然后进行标准的二分法查找,由于对 IP 前缀的长度不敏感,因此可适用于 IPv6 的路由查找。此时每个路由所保存的信息为:(128 位端点值,5 位前缀长度,16 位下一跳索引),共 19 个字节。因此通过实验和计算我们可得到表 2 的数据与 RR B-tree 算法进行比较。

表 2 算法对真实小容量 IPv6 的性能($m = 32, L = 32, W = 128$)

Tab.2 RR B-tree's performance to process real small IPv6 route table ($m = 32, L = 32, W = 128$)

前缀数目	B 树高度	结点内范围比较树最大高度	耗费结点	耗费内存	搜索平均访存次数	更新平均访存次数	二分法在最好情况下的最小访存数	二分法更新平均访存次数	顺序存储所有路由时的存储空间
893	4	3	88	34.0KB	20.1	23.6	40	54.6	16.97KB

为了衡量算法对大容量 IPv6 路由表的性能,我们生成了大容量的 IPv6 路由表和报文流对算法进行测试。由于目前 IPv6 处于试运行阶段,所获得的试验数据较少且仅能作为参考,而且路由查表问题所关注 IPv6 路由表分布和 IPv6 报文流特征等都不明确。虽然 RFC2460[1]、RFC3513[9]、RFC3587[10] 等说明了 IPv6 的寻址结构和地址格式等,但只在 RFC1887[11] 中给出了 IPv6 地址的一些分配原则和方法。而这些原则和方法在目前 IPv6 尚未大规模应用的情况下难以在未来的真实 IPv6 路由表分布之间找到对应关系。因此如同 Radix Trie 树算法[4]在开始用于路由查找时不对路由表分布进行假设一样,这里也不对 IPv6 路由表和报文做前提假设,即 按照上述 RFC 规定的寻址方案随机生成合法的 IPv6 全球可聚集单播地址的路由和报文流。

表 3 算法对大容量 IPv6 的性能($m = 32, L = 32, W = 128$)

Tab.3 RR B-tree's performance to process huge IPv6 route tables($m = 32, L = 32, W = 128$)

前缀数目	B 树高度	结点内范围比较树最大高度	耗费结点	耗费内存	搜索平均访存次数	更新平均访存次数	二分法在最好情况下的最小访存数	二分法更新平均访存次数	顺序存储所有路由时的存储空间
128K	4	4	11.3K	3.9M	20.05	43.5	72	89.2	2.4M
1M	5	4	82.9K	25.8M	24.03	64.7	84	95.3	19M
8M	6	4	624.1K	191.5M	29.2	95.2	96	101.8	152M
64M	7	4	4.7M	1.52G	35.7	157.6	108	107.1	1.22G
512M	8	4	34.7M	10.8G	42.6	279.2	120	113.9	9.73G

从该表 2 和表 3 中我们可以看到,算法的搜索性能比标准的二分法高大约 3 倍,更新性能与二分法性能基本持平,而空间性能则基本上同顺序存储路由项所需要的空间相当,而且算法的性能随着规则数

目的快速增长减小得很慢,因此算法适用于大规模的IPv6路由查表和路由表管理。

5 结束语

IPv6是未来Internet的发展趋势,本文针对IPv6路由查找这一重要问题中所特有的路由表容量巨大和长地址这两个难点进行研究,提出了基于范围表示B树的IP路由查找算法,算法利用B树的优点,在B树结点内部使用分段范围比较树对长地址进行处理,使算法对大容量、超大容量IP路由表具有良好的空间性能和搜索时间性能,而且通过控制B树的参数,可以对算法的时空性能进行一定调整,从而适应不同环境中的路由表大小。

IP路由查找和报文分类的性能是互联网络性能的关键问题之一,我们一直在对其进行研究[12~15]。未来我们在该领域的研究将重点着眼于如下几个方面:对RR B-tree算法并行化硬件化研究;研究利用新的器件对算法进行优化;研究在分布式、并行式、集群式等新型路由器体系结构中并行和流水地进行路由查表和报文分类分类的理论、算法和关键技术等。

参考文献:

[1] Deering S, Hinden R. RFC 2460: Internet Protocol, Version 6 (IPv6) Specification[S]. http://www.ietf.org/rfc/rfc2460.txt., Dec. 1998.

[2] Srinivasan V, Karlsson G. Fast IP Lookups Using Controlled Prefix Expansion[J]. ACM Transactions on Computer Systems, 1999, 17: 1-40.

[3] Ruiz-Sanchez M A, Biersack E W, Dabbous W. Survey and Taxonomy of IP Address Lookup Algorithms[J]. IEEE Network, 2001, 15: 8-23.

[4] Knuth D E. 计算机程序设计艺术第3卷 排序和查找(第2版)[M]. 苏运霖,译. 北京:国防工业出版社, 2003, 458-478.

[5] Degermark M, Brodnik A, Carlsson S, et al. Scalable High Speed IP Routing Lookups[C]. Proceedings of ACM Sigcomm, 97, New York: ACM Press, 1997, 3-14.

[6] Gupta P, Prabhakar B, Boyd S. Near-optimal Routing Lookups with Bounded worst Case Performance[A]. Proceedings of IEEE Infocom. 2000[C], New York: IEEE Xplore Press, Mar. 2000: 1184-1192.

[7] Suri S, Varghese G, Warkhede P R. Multiway Range Trees: Scalable IP Loolup with Fast Updates[R]. Washington Univ. Technique Report, 1999: 99-128.

[8] 中国Cernet中心. 中国Cernet中心全球IPv6路由表[DB]. http://bgpview.6test.edu.cn/datav6/, Feb. 2005.

[9] Deering S, Hinden R. RFC 3513: Internet Protocol Version 6 (IPv6) Addressing Architecture[S]. http://www.ietf.org/rfc/rfc3513.txt, Apr. 2003.

[10] Deering S, Hinden R, Nordmark E. RFC 3587: IPv6 Global Unicast Address Format[S]. http://www.ietf.org/rfc/rfc2460.txt, Aug. 2003.

[11] Rekhter Y, Li T. RFC 1887: An Architecture for IPv6 Unicast Address Allocation[S]. http://www.ietf.org/rfc/rfc2460.txt, Dec. 1995.

[12] Tan M F, Gong Z H. High Speed IP Lookup Algorithm with Scalability and Parallelism Based on CAM Array and TCAM[A]. Proceedings of IEEE ICC 2004[C], New York: IEEE Xplore Press, 2004, 2: 1085-1089.

[13] 谭明锋, 龚正虎. 基于ASIC实现的高速可扩展并行IP路由查找算法[J]. 电子学报, 2005, 33(2): 209-213.

[14] 彭元喜, 唐玉华, 龚正虎. 基于压缩NH表的高速IP路由查找算法的研究[J]. 电子学报, 2002, 30(2): 196-200.

[15] 彭元喜, 龚正虎. 基于LSOT的高速IP路由查找算法[J]. 计算机学报, 2002, 25(1): 106-111.