TSB:一种多阶段 IPv6 路由表查找算法

李振强[1],郑东去[2],马 严[1,2]
(1. 北京邮电大学计算机科学与技术学院,北京 100876; 2. 北京邮电大学信息网络中心,北京 100876)

摘 要: 充分分析 IPv6 地址结构、IPv6 地址分配策略和 IPv6 骨干网路由表的特点后,将二叉树、段表和路由桶技术相结合,提出一种多阶段 IPv6 路由表查找算法. 和已有算法相比,提出的算法查找速度快、占用内存少、扩展性好、支持增量更新. 实验结果表明算法的软件参考实现在装有 P4 2.4GHz CPU,512M DDR333 内存和 Linux 操作系统的普通 PC 机上的查找能力可以到达 16MPPS(Million Packet per Second),这可以满足 10Gbps 80 字节 IPv6 最小包的线速转发. 对于当前 IPv6 骨干网 BGP 路由表,算法的参考实现只占用几百 K 字节的内存.

关键词: 算法; 路由查找; IPv6; 多阶段
中图分类号: TP393 **文献标识码**: A **文章编号**: 0372-2112 (2007) 10-1859-06

TSB:A Multi-Stage Algorithm for IPv6 Routing Table Lookup

LI Zhen-qiang[1], ZHENG Dong-qu[2], MA Yan[1,2]
(1. *School of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*;
2. *Network Information Center, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*)

Abstract: With the combination of binary tree, segment table and route bucket after sufficient and thorough analysis of the hierarchy of IPv6 address, IPv6 address allocation policy and the characteristics of real live IPv6 backbone BGP routing tables, we propose a multi-stage algorithm for IPv6 routing table lookup in this paper. Compared with previous algorithms, the proposed scheme performs faster, occupies less memory, scales better, and supports incremental update. The evaluation results show that the sample software implementation of the proposed algorithm can forward at a rate of 16MPPS (Million Packet per Second), or 10Gbps for 80-byte minimal IPv6 packets on a PC with Pentium4 2.4GHz CPU, 512M DDR333 memory, and Linux operating system. The sample implementation only needs several hundreds of kilobytes memory for the current real live IPv6 backbone BGP routing tables.

Key words: algorithm; routing lookup; IPv6; multi-stage

1 引言

IPv6 是下一代互联网 NGI(Next Generation Internet)的核心协议,和现在无处不在的计算机网络使用的网际协议—IPv4 相比,IPv6 最大的特点是它使用 128 位超长 IP 地址,如此巨大的地址空间几乎可以为地球上的每一粒沙子分配一个 IP 地址. 随着互联网的进一步发展,IPv4 地址短缺的问题变得越来越急迫,IPv6 也因此吸引了学术界和产业界的广泛关注和认可,特别是在欧洲和亚太地区,人口多,获得的 IPv4 地址少,对 IPv6 的研究、开发和部署非常积极. 目前世界上规模最大的纯 IPv6 网络,中国下一代互联网 CNGI(China Next Generation Internet)骨干网—CERNET2 已于 2004 年 12 月 25 日开通①. 随着 IPv6 被广泛认可和逐步部署,拥有丰富 IPv4 地址资源的美国对 IPv6 的态度也发生了变化. 美国国防部 2003 年 6 月宣布到 2008 年美国国防部的网络要全部升级到 IPv6②. 美国白宫管理和预算办公室 2005 年 6 月 29 日也宣布美国联邦机构到 2008 年 6 月必须使用 IPv6③. IPv6 的快速发展和广泛部署时期即将到来.

128 位的 IPv6 地址给路由器的设计和制造提出了挑战,特别是对路由表查找算法. 在数据平面,IPv6 路由器收到 IPv6 数据包后仍然要在路由表中做最长前缀匹配 LPM(Longest Prefix Matching)以决定数据包的转发端口. 然而,现有的路由表查找算法[1]大都针对 IPv4 的 32 位地址设计,要么不能扩展到 IPv6,要么扩展到 IPv6 后占用的内存急剧增加或者查找性能大大降低. 因此,我们必须研究 IPv6 的特点,专门针对 IPv6 128 位的 IP 地

收稿日期:2006-07-31;修回日期:2007-08-19

① http://www.edu.cn/cernet10/
② http://www.usipv6.com/2003arlington/presents/MariynKraus.pdf
③ http://nationaljournal.com/about/technologydaily

址设计高效的路由表查找算法.

已有的 IPv6 路由表查找算法大致可以分为多比特查找树(trie)算法[2,3]、使用硬件 TCAM 的算法[4]、基于前缀长度的二分查找算法[5]、转换为包分类问题的算法[6]等.多比特查找树(trie)算法有时也被称为分段算法,它把 128 位的 IPv6 地址分为长度相同或者不同的若干分段并建立段表(多比特查找树),然后在这些段表上进行查找.分段算法的查找性能和分段的层数成反比,内存消耗和分段的层数成正比,为了平衡内存消耗和查找性能之间的矛盾,文献[3]中使用了复杂的压缩算法.使用硬件 TCAM 来完成 IPv6 LPM 的查找复杂度为 $O(1)$,一次查找只需一次访存时间,但是 TCAM 集成度低、芯片面积大、功耗大、价格昂贵,而且对路由更新的支持也比较复杂.基于前缀长度的二分查找算法的查找性能为 $O(W/M * \log W)$ 次哈希,M 是机器的字宽,W 是 IP 地址的长度,对于 IPv4,W 等于 32,对于 IPv6,W 等于 128.这样,在 32 位机上,使用基于前缀长度的二分查找算法来完成 IPv6 的 LPM,最多需要 $128/32 * \log 128 = 28$ 次哈希.但是,适用于各种 IPv6 前缀长度的哈希函数在文献中未见报道,且哈希由于哈希冲突的存在,其最坏情况下的性能难以预测,使用哈希的路由器在最坏情况下也难以保证"小包线速".通过切割 IPv6 地址,文献[6]将 IPv6 的路由查找问题转换成了多维包分类问题,并使用 Pankaj Gupta 等提出的 RFC(Recursive Flow Classification)包分类算法完成了 IPv6 的 LPM.如果采用硬件并行加流水的实现方式,文献[6]在查找性能上可以到达一次 IPv6 LPM 只需一次访存.但是由于复杂的预处理,文献[6]的内存消耗较大且更新困难,基本上是一种静态算法,路由更新时需要重构整个数据结构.

总之,现有的 IPv4/IPv6 路由表查找算法还不能满足 IPv6 对路由查找的需要.为了改善这种不利情况,本文研究了 IPv6 的地址分配策略,专门针对 IPv6 地址结构和路由表的特点提出一种多阶段 IPv6 路由表查找算法—TSB 算法(binary Tree,Segment table,route Bucket).该算法将二叉树(binary tree)、段表(segment table)和路由桶(route bucket)技术相结合,而不是像已有的 IPv6 查找算法那样采用单一的技术.TSB 算法具有查找速度快、占用内存少、扩展性好、支持增量更新的特点.软件参考实现的实验结果表明该算法在装有 P4 2.4GHz CPU,512M DDR333 内存和 Linux 操作系统的普通 PC 上的查找性能就能达到 16MPPS(Million Packetper Second).即使 IPv6 网络中全部是 80 字节的最小包,该算法也能满足 10Gbps 线速转发的需求.至于内存占用,对于当前 IPv6 骨干网 BGP 路由表,参考实现只占用几百 K 字节的内存.

2 IPv6 地址结构、地址分配策略和骨干网路由表的特点

2.1 IPv6 地址结构

RFC3587 规定的最新 IPv6 全球单播地址结构如图 1 所示,它包括 3 个部分:n 比特的全球路由前缀、64-n 比特的子网标识和 64 比特的接口标识.全球路由前缀标识一个站点,子网标识用于标识站点内的一个子网,接口标识是用 EUI-64 格式来标识的子网内的一个网络接口.通常用于路由的只有全球路由前缀和子网标识,共 64 比特.因此,IPv6 骨干网 BGP 路由表中很少有前缀长度超过 64 的路由表项(详见 IPv6 骨干网路由表的特点部分).

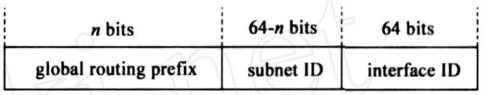

图 1 IPv6 全球单播地址结构

2.2 IPv6 地址分配策略

IPv6 的地址分配采用分级结构,参见图 2.管理全球 IP 地址分配的最高机构是 IANA(Internet Assigned Number-Authority),它将/23 的 IPv6 地址分配给 RIR(Regional Internet Registries).RIR 再将/32 的 IPv6 分配给下级地址分配机构 ISP(Internet Service Provider)或者 LIR(Local Internet Registries).最终用户 EU(End Users)可以根据需要向 ISP/LIR 申请/48、/64 或者/128 的 IPv6 地址.目前全球共有 5 个 RIR:ARIN、RIPE、APNIC、AfriNIC 和 LACNIC,分别负责北美地区、欧洲地区、亚太地区、非洲地区和拉美地区的 IP 地址分配.虽然 IPv4 的地址分配也采用同样的分级结构,但 IPv6 在地址分配上吸取了 IPv4 地址分配的经验教训,IPv6 128 位的超大地址空间和灵活的地址结构(参见图 1)也为 IPv6 制定更好的地址分配策略提供了可能.为了更好保证 IPv6 地址分配的可聚合性和节约性,IPv6 明确规定了各级地址分配机构可以分配的前缀长度,IAB/IESG 在 RFC3177 中还对 IPv6 的地址分配提出了建议,ARIN、RIPE 和 APNIC 也联合制定了 IPv6 地址分配策略[7].这使得 IPv6 路由表呈现出更好的层次性(详见 IPv6 骨干网路由表的特点部分),不同的层次有不同的特点,适合用不同的数据结构来组织,本文提出的多阶段路由表查找算法正是利用了这一点.

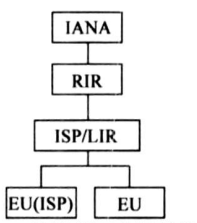

图 2 地址分配分级结构

2.3 IPv6 骨干网路由表的特点

Li 等在文献[6]中分析了最新的 IPv6 骨干网路由表的特点,现在把和本算法相关的特点总结如下.(1)

目前 IPv6 路由表中的路由条目数不超过 1000 条. 这一方面是因为 IPv6 的实际部署才刚刚起步,另一方面是因为如前所述的 IPv6 灵活的地址结构和良好的地址分配策略使得 IPv6 路由能够更好的聚合. (2) 前缀长度分布特点. IPv6 路由表中的路由前缀以前缀长度为 32、48 和 64 的为最多,这反映了前面所述的 IPv6 的地址分配策略. 虽然 IPv6 的地址长度为 128 位,但 IPv6 路由表中前缀长度大于 64 的路由表项非常少,并且没有前缀长度小于 16 的路由表项. 这样,IPv6 地址的前 16 位可以做精确匹配,后 64 位因为数量非常少也应该采用不同的查找方式,比如线性查找或者使用少量的 TCAM.

除文献[6]中发现的这些特点外,通过分析,我们还发现目前 IPv6 骨干网路由表中路由前缀的前 16 位共有 9 种不同的取值:2001、2002、2003、2400、2404、2610、2800、2A01 和 3ffe ①,且绝大多数路由前缀都以 2001 开头,以其它数值开头的路由前缀非常少,一个路由表中只有几条. 进一步我们发现,以 2001 开头的路由前缀其第 17-32 位有很高的区分度,即第 17-32 位相同的路由前缀非常少. 如图 3 所示,第 17-32 位相同的路由前缀一般只有几条,最多也不过十几条,目前还没有任何前缀的占了将近一半.

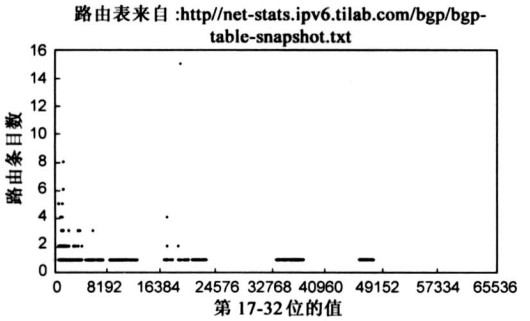

图 3　第 17-32 位的区分度

3　多阶段 IPv6 路由表查找算法

3.1　数据结构

充分利用前文所述的 IPv6 地址结构、IPv6 地址分配策略和 IPv6 骨干网路由表的特点,本文设计的 IPv6 路由表查找算法采用 3 阶段查找方法,如图 4 所示.

第 1 阶段是以 2001 作为根节点的二叉树,它由路由前缀的前 16 位组成,因为 IPv6 路由表中没有前缀长度小于 16 的路由表项,所以在二叉树上只需要做精确匹配. 以 2001 作为二叉树的根节点是因为目前 IPv6 网络上的流量的 IPv6 地址绝大部分以 2001 开头,这样网络上的大部分流量在二叉树上只需要一次匹配,提高了算法的平均查找性能.

第 2 阶段是段表或者路由桶. 对路由桶的定义和讨论见下文对第 3 阶段的介绍. 当路由条目数小于某个阈值(比如 64)时用路由桶组织,否则用段表组织,阈值的具体数值由算法的查找效率和算法所占内存之间的平衡决定. 第 2 阶段采用两种不同的数据结构,充分利用了 IPv6 路由表的特点,降低了算法的内存消耗:因为绝大多数路由前缀都以 2001 开头,所以 2001 链接段表;以其它值开头的路由前缀非常少,所以链接路由桶(如 2800,2002 等). 段表的大小是 16 比特,共有 2^{16} 项,由路由前缀的第 17-32 位组成. 这样组织段表不仅效率高,而且具有一定的前瞻性. 效率高是因为路由前缀的第 17-32 位具有很高的区分度(参见图 3),因此链接在段表上的路由桶都不会太大,这样就节约了内存消耗,提高了算法的查找性能. 具有前瞻性是因为第 17-32 位的值中虽然有很多还没有路由前缀,但我们仍然采用 16 位的段表,这样内存消耗只有 64K,将来有新的路由前缀加入时,数据结构也不用调整.

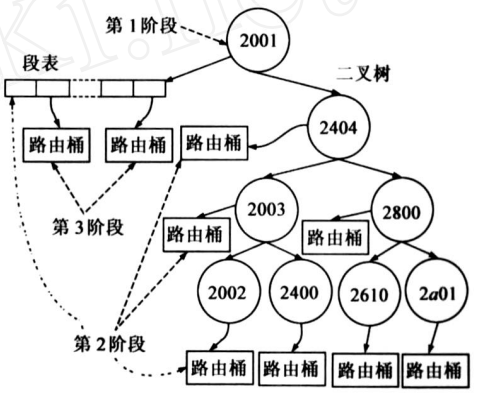

图 4　3 阶段 IPv6 路由表查找算法数据结构示意图

第 3 阶段是链接在段表上的路由桶. 路由桶是路由表项的集合,根据路由桶中路由表项数目的不同,路由桶可以采用不同的组织方式. 当路由桶中只有几条路由时,线性查找一般即可满足要求;当路由桶中有十几条到几十条路由时,可以采用基于前缀区间的二分查找[8]或者基于前缀长度的二分查找[5]. 路由桶链接的位置不同,关心的路由前缀的比特位置也不同:第 2 阶段中的路由桶关心路由前缀的第 17-128 比特,第 3 阶段中的路由桶关心路由前缀的第 33-128 比特.

提出的算法在不同的阶段分别使用了树(Tree)、段表(Segment Table)和路由桶(Bucket)三种数据结构,因此我们把它叫做 TSB 算法.

3.2　路由查找

路由查找的伪代码如图 5 所示. 第 2 行首先将默认路由的下一跳信息存入 BMP(Best Matching Prefix),然后

① 3FFE::/16 原来分配给 6bone 用于 IPv6 的测试,因为 IPv6 的实际部署已经开始,RFC3701 声明 3FFE::/16 范围内的地址到 2006 年 6 月停止使用,以 3FFE 开始的 IPv6 地址将不再出现在互联网中. 本文也不再讨论以 3FFE 开始的地址.

提取目的 IPv6 地址的第 1-16 比特在二叉树上查找(第 3-4 行),如果查找失败就返回默认路由(第 5 行),否则记录当前的 BMP(第 6 行),并根据匹配节点的指针类型继续匹配.如果匹配节点没有链接,就返回当前记录的 BMP(第 7 行);如果匹配节点链接路由桶,就用目的 IPv6 地址的第 17-128 比特在路由桶中查找并返回查找结果(第 8-11 行);如果匹配节点链接段表,就用目的 IPv6 地址的第 17-32 比特的值定位到段表中的相应表项,并记录当前的 BMP(第 13-15 行);如果段表表项没有链接路由桶就返回当前 BMP(第 16 行),如果段表表项链接路由桶,就用目的 IPv6 地址的第 33-128 比特在路由桶中查找并返回查找结果(第 17 - 18 行).

```
1  search TSB(dstIP){
2     BMP = next hop of default route;
3     key1 = the 1-16 bits of dstIP;
4     node = searchBinaryTree(key1);
5     if (! node) return BMP;
6     BMP = node.nexthop;
7     if (! node.ptr) return BMP;
8     if (the type of node.ptr is bucket){
9       key2 = the 17-128 bits of dstIP;
10      searchBucket(node.ptr, key2);
11    }
12    else{
13      key3 = the 17-32 bits of dstIP;
14      ptr = node.ptr + key3;
15      BMP = ptr->nexthop;
16      if (! ptr->bkt) return BMP;
17      key4 = the 33-128 bits of dstIP;
18      searchBucket(ptr->bkt,key4);
19    }
20 }
```

图 5 TSB 算法的路由查找伪代码

如 3.1 节所述,路由桶的组织方式由路由桶中的路由条数决定,并不采用单一的组织方式,这样就既能保证算法的查找性能,又能降低算法的内存消耗.当路由桶中只有几条路由时,可以采用简单的线性表结构和线性查找方式.当路由桶中有十几条到几十条路由时,可以采用基于前缀区间的二分查找[8]或者基于前缀长度的二分查找[5].

3.3 路由更新及算法的可扩展性

TSB 算法能够满足增量更新的要求,即路由更新时只用更新局部数据结构而不用从头重构整个数据结构.由前所述,TSB 算法不需要复杂的预处理,添加路由表项 R 时,首先找到 R 应该添加的位置,这有三种可能.(1) R 应该添加到二叉树中,由二叉树数据结构的特点,这很容易实现.(2) R 应该添加到段表中,由于可能涉及前缀扩展(当 R 的前缀长度小于 32 时,需要将其扩展到 32 位),这种情况比较复杂.假设 R 的前缀长度为 L,$17 \le L \le 32$,R 的第 17 到 L 比特的数值为 V.首先把段表中从第 2^V 项起连续 2^{32-L} 项中存储的前缀长度 L' 和 L 相比较,如果 $L \ge L'$,就用 L 替换 L',并把 R 存储到段表表项中,否则检查下一项.(3) R 应该添加到路由桶中,根据路由桶的组织方式将 R 添加到相应的位置,如果加入 R 后路由桶中的路由条目数达到阈值,就将路由桶改用段表组织.

删除路由表项和添加路由表项类似,也有三种情况,下面以删除路由表项 R 为例进行说明.(1) 当 R 需要从二叉树中删除时,由二叉树数据结构的特点,这很容易实现.(2) 当 R 需要从段表中删除时,假设 R 的前缀长度为 L,R 的第 17 到 L 比特的数值为 V.此时,$17 \le L \le 32$,只需更新段表中从第 2^V 项起连续 2^{32-L} 项中存储的路由信息,并检查删除 R 后段表中的路由条目数是否小于阈值,如果小于阈值就将段表改用路由桶组织.(3) 当 R 需要从路由桶中删除时,根据路由桶的组织方式首先找到 R,然后将其删除.如果删除 R 后路由桶中没有任何路由条目,就将路由桶也删除.

由 TSB 算法添加和删除路由表项的过程可以看出该算法具有良好的可扩展性和对未来路由表的适应性.添加路由表项时,如果二叉树节点链接的路由桶中的路由条目数达到 3.1 节所述的阈值,就把路由桶改用段表组织;如果段表链接的路由桶中的路由条目数达到 3.1 节所述的阈值,也可以将第 3 阶段的路由桶改用段表来组织,这样就建立了多级段表结构.可见 TSB 算法不仅适用于目前 IPv6 骨干网路由表的查找,而且对未来 IPv6 网络的发展有很好的适应性.

4 实验结果

为了验证 TSB 算法的性能,我们在 Linux 操作系统下用 C++ 开发了 TSB 算法的软件原形系统,并和最新算法 RST[6]和 LPFST[9]以及经典算法 Patricia[10]和 LC-trie[11]的性能进行比较.测试用 PC 机安装 Red Hat Enterprise Linux Advanced Server4.0 操作系统,硬件配置为 512M DDR333 内存,一个 P4 2.4GHz CPU.

实验中所用的 IPv6 骨干网路由表的规模如图 6 所示,目前这些路由表中的路由条目数都不足 1000 条.

为了比较算法的查找性能,对于每个路由表我们都产生 100 万个数据包来进行测试.这 100 万个验证数据包不都是随机生成的,其中 80 %是随机从相应路由表中选择一条路由条目,然后据此产生验证包,这样的验证包至少能够匹配上路由表中的一条路由;另外 20 %的验证包是纯粹随机产生的,这样的验证包可能匹配上路由表中的路由条目,也可能只匹配默认路由.路由查找的性能比较结果见图 7.由图 7 可以看出本文提

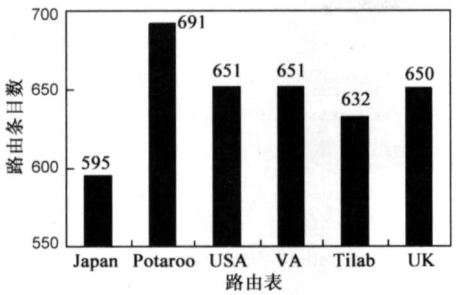

图 6　实验中所用的 IPv6 骨干网路由表的规模

出的 TSB 算法的查找速度最快,大约是 RST 算法的 5 倍,Patricia 算法的 4 倍. TSB 算法的软件参考实现的查找性能就能够达到 16MPPS,这样即使 IPv6 网络中全部是 80 字节的最小包,TSB 算法也能满足 10Gbps 线速转发的需求.

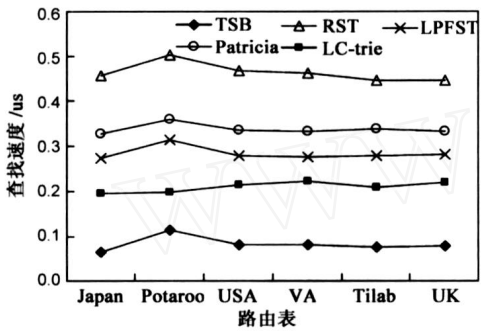

图 7　用真实路由表测试的查找速度比较结果

表 1 是内存消耗的比较结果. 对于当前 IPv6 骨干网 BGP 路由表,TSB 算法的软件参考实现占用的内存不超过 400K 字节. RST 算法由于使用了复杂的数据结构和预处理,不但不支持增量更新,而且占用的内存也较多,大约是 TSB 算法的 1.5 倍. 由于 TSB 算法中至少要使用一个段表组织以 2001 开始的路由表项,这个段表占用的内存为 64K * 段表表项的大小,所以 TSB 算法占用的内存比 LPFST、Patricia 和 LC-trie 要多.

表 1　用真实路由表测试的内存占用比较结果(单位:KB)

	TSB	RST	LPFST	Patricia	LC-trie
Japan	387.91	603.12	36.48	33.21	35.65
Potaroo	396.36	713.22	40.80	36.20	40.68
USA	390.11	631.23	39.72	35.87	39.34
VA	390.15	637.91	39.72	35.78	39.23
Tilab	389.56	645.49	39.00	35.11	37.83
UK	390.09	633.29	39.42	35.78	39.20

由图 6 可以看出目前 IPv6 骨干网路由表的规模都比较小. 为测试算法的可扩展性和对未来 IPv6 网络的适应性,我们用 V6Gene[12] 生成了不同大小(从 1K 到 200K)的 IPv6 路由表,用它们对算法的性能进行测试. 图 8 是包含 100K 条表项的路由表的前缀长度的分布.

用生成的路由表测试的查找性能的比较结果如图 9 所示. RST 算法的扩展性较差. 当路由表中的路由前缀超过 2000 条时,RST 算法的预处理时间显著加长,占用的内存急剧增加,这使得算法无法进行实际测试. 因此,这部分没有使用 RST 算法进行性能比较. 图 9 说明,当路由表的大小不断增大时,TSB 算法的查找速度始终是最快的. 因此 TSB 算法的可扩展性最好,对未来 IPv6 网络有着良好的适应能力.

用生成的路由表测试的内存占用的比较结果见图 10. 从中可以看出,TSB 算法的可扩展性最好,当路由表中的路由表项的数量超过 100K 时,TSB 算法占用的内存最少. 当路由表达到 200K 条时(目前 IPv4 骨干网路由表的规模),TSB 算法仅占用内存 9M 字节,其中主要是被多级段表所消耗.

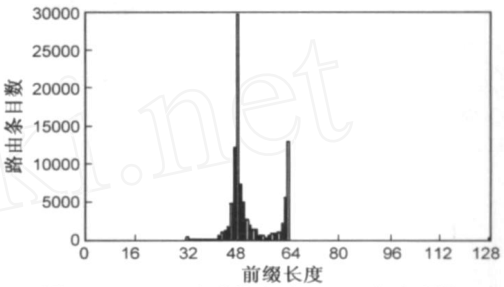

图 8　V6Gene 生成的包含 100K 条表项的路由表的前缀长度的分布

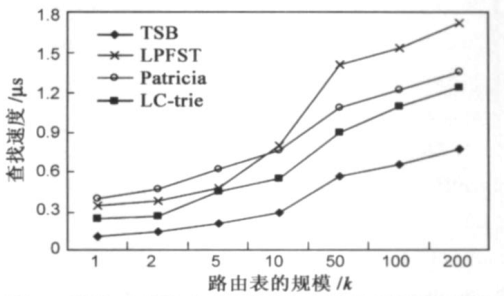

图 9　用生成的路由表测试的查找性能比较结果

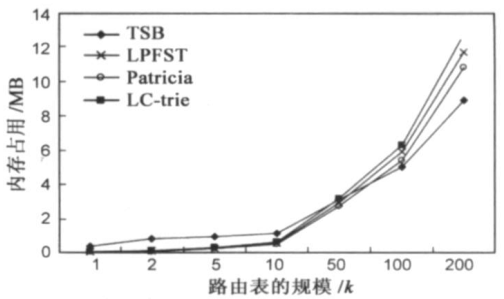

图 10　用生成的路由表测试的内存占用比较结果

5　结论

充分分析了 IPv6 地址结构、IPv6 地址分配策略和 IPv6 骨干网路由表的特点,将二叉树、段表和路由桶技术相结合,提出了一种多阶段 IPv6 路由表查找算法 — TSB 算法. 和已有算法相比,提出的算法查找速度快、占用内存少、扩展性好、支持增量更新. 实验结果表明算

法的软件参考实现在装有 P4 2.4GHz CPU,512M DDR333 内存和 Linux 操作系统的普通 PC 机上就能达到 16MPPS 的查找能力.对于当前 IPv6 骨干网 BGP 路由表,参考实现占用的内存也只有几百 K 字节.

参考文献:

[1] M A Ruiz-Sanchez, E W Biersack, W Dabbous. Survey and taxonomy of IP address lookup algorithms[J]. IEEE Network, 2001,15(2):8 - 23.

[2] M Zitterbart, T Harbaum, D Meier, D Brokelmann. Efficient routingtable lookup for IPv6[A]. In Proc. IEEE High PerformanceCommunication Systems[C]. Chalkidiki, Greece: IEEE Press,1997. 1 - 9.

[3] 姚兴苗,李乐民.一种快速 IPv6 路由查找方案[J].计算机学报,2005,28(2):214 - 219.
X Yao, L Li. A fast IPv6 route lookup scheme[J]. Chinese Journalof Computers,2005,28(2):214 - 219. (in Chinese)

[4] T Hayashi, T Miyazaki. High speed table lookup engine for IPv6 longest prefix match[A]. In Proc IEEE GLOBECOM[C]. Rio deJaneiro,Brazil:IEEE Press,1999. 1576 - 1581.

[5] M Waldvogel. Fast Longest Prefix Matching: Algorithms, Analysis,and Applications[D]. Swiss: Swiss Federal Institute of Technology,2000.

[6] Z Li, X Deng, H Ma, Y Ma. Divide-and-conquer:a scheme for IPv6 Address longest prefix matching[A]. In Proc IEEE Workshop on High Performance Switching and Routing[C]. Poznan, Poland:IEEEPress,2006. 37 - 42.

[7] APNIC, ARIN, RIPE NCC. IPv6 Address Allocation and AssignmentPolicy[OL]. http://www.ripe.net/ripe/docs/ripe-267.html. Jan 2003.

[8] B Lampson, V Srinivasan, G Varghese. IP lookups using multiway and multicolumn search[A]. In Proc IEEE INFOCOM[C]. San Francisco,USA:IEEE Press,1998. 1248 - 1256.

[9] L Wuu, K Chen, T Liu. A longest prefix first search tree for IP lookup[A]. In Proc. IEEE ICC[C]. Seoul, Korea:IEEE Press, 2005. 989 - 993.

[10] K Sklower. A Tree-based packet routing table for berkeley unix[A]. In Proc. 1991 Winter USENIX Conference[C]. Dallas,USA: UsenixAssociation,1991. 93 - 99.

[11] S Nilsson, G Karlsson. IP-address lookup using LC-tries[J]. IEEE Journal on Selected Areas in Communications,1999,17(6):1083 - 1092.

[12] K Zheng, B Liu. V6Gene:A scalable IPv6 prefix generator for route lookup Algorithm Benchmark[A]. In Proc IEEE AINA'06[C]. Vienna,Austria:IEEE Press,2006. 147 - 152.

作者简介:

李振强 男,1976 年 6 月出生于河北省沙河市,博士.主要研究领域为路由查找和包分类算法、网络安全等. E-mail:lizq@buptnet.edu.cn

郑东去 男,1979 年 6 月出生于江苏省丹阳市,硕士.主要研究领域为路由查找算法. E-mail:zhengdq@buptnet.edu.cn

马 严 男,1955 年 9 月出生于北京市,教授,博导.主要研究领域为下一代网络协议及应用. E-mail:mayan@bupt.edu.cn