Journal of Computer Applications
计算机应用,2013,33(5):1194 - 1196,1202
ISSN 1001-9081
CODEN JYIIDU
2013-05-01
http://www.joca.cn

文章编号:1001 - 9081(2013)05 - 1194 - 03
doi:10.3724/SP.J.1087.2013.01194

基于无冲突哈希表和多比特树的两级 IPv6 路由查找算法

杜　飞[1],董治国[2],苗　琳[3],庹宇鹏[1*]
(1. 中国科学院 信息工程研究所,北京 100093;　2. 中国核工业集团公司 中国核电工程有限公司,北京 100840;
3. 国家计算机网络应急技术处理协调中心,北京 100027)
(*通信作者电子邮箱 tuoyupeng@iie.ac.cn)

摘　要:为了提高 IPv6 的路由查找效率,根据 IPv6 路由前缀分布规律和前缀层次关系,提出了基于无冲突哈希表和多比特树的两级 IPv6 路由查找算法。该算法将地址前缀划分区间并按长度为 32,40,48 比特分别存储于 3 个哈希表中,剩下不足的前缀比特由多比特树存储,IPv6 路由查找时在无冲突哈希表和多比特树中两级查找。实验表明,该查找算法的平均查找路径数为 1.0 ~ 1.7,适用于高速的 IPv6 路由查找。

关键词:IPv6;路由前缀;无冲突哈希表;多比特树
中图分类号:TP393.03　　**文献标志码**:A

Two-stage IPv6 route search algorithm based on perfect-Hash table and multibit-trie

DU Fei[1], DONG Zhiguo[2], MIAO Lin[3], TUO Yupeng[1*]
(1. *Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093, *China*;
2. *China Nuclear Power Engineering Corporation Limited, China National Nuclear Corporation, Beijing* 100840, *China*;
3. *National Computer Network Emergency Response Technical Team Coordination Center of China, Beijing* 100027, *China*)

Abstract: Based on the analysis of the prefix length distribution and prefix level relationships of the routing table, a new IPv6 route search algorithm based on perfect-Hash table and multibit-trie was proposed to improve the efficiency of the IPv6 address search. The prefixes of the addresses divided into intervals were stored in three Hash tables according to the length of 32, 40 and 48, and the rest bits were stored in multibit-tries. IPv6 routing adopted a two-stage search. The first stage was in the perfect-Hash table and the second in the multibit-trie. The experimental results demonstrate that the average search path of the algorithm is 1.0 - 1.7, and it can be applied to the high performance IPv6 route search.

Key words: IPv6; routing prefix; perfect-Hash table; multibit-trie

0　引言

路由器作为 Internet 互连的核心设备,它的处理速度是网络通信的主要瓶颈之一,其性能则直接影响着网络互联的质量。高效的路由表查找算法是影响路由器转发效率的重要因素。随着 Internet 网络规模的扩大,路由表的大小与日俱增,随着无分类域间路由(Classless Inter Domain Routing, CIDR)[1]的引入,IP 地址查找从一个精确匹配的问题转变成为一个最优匹配的问题,必须采用最长前缀匹配(Longest Prefix Match, LPM)算法[2-3]解决,从而大大增加了 IP 地址查找的复杂性。特别是 IPv6 协议带来了巨大的地址空间和更长的地址格式,这些都对提高路由表查找算法效率提出了挑战。因此,必须研究适合 IPv6 的路由查找算法。

本文仅针对 IPv6 的全球单播地址进行讨论,提出了基于无冲突哈希表和多比特树的两级 IPv6 路由查找算法(perfect Hash Table Multibit-Trie, pHTMT),并利用一级索引结构和 3 个无冲突 Hash 表来存储 IPv6 的地址前缀,剩余的部分前缀由变步长的多比特树存储。实验表明:该算法支持大部分的 IPv6 单播地址,平均查找路径数为 1 ~ 1.7。

1　相关研究

目前 IPv6 的地址查找算法主要借鉴 IPv4 的地址查找方法[4-6],包括基于 Trie 树的方法、基于范围的查找(Search on Prefix Ranges)、基于前缀长度的二分查找(Binary Search on Prefix Lengths)以及基于三态内容寻址存储器(Ternary Content Addressable Memory,TCAM)的算法[7]等,表 1 给出了几种典型路由查找算法的性能分析(其中:W 表示前缀长度,N 表示路由前缀的数目)。

表 1　各种算法性能比较

算法	搜索复杂度	存储复杂度
Binary Trie	$O(W)$	$O(NW)$
k-bits Trie	$O(W/k)$	$O(2^kNW/k)$
Binary Search on Prefix Lengths	$O(\mathrm{lb}\ W)$	$O(N\ \mathrm{lb}\ W)$
k-way Range Search	$O(\log_k N)$	$O(N)$
TCAM	$O(1)$	$O(N)$

Waldvogel 等[6]提出了一种按前缀长度进行二分查找的算法,该算法将最长前缀匹配按前缀长度分解成一系列的精确匹配,并将前缀按长度分别存储在不同的 Hash 表中,地址查找时,按前缀长度对所有的 Hash 表进行二分查找。该算法查找的平均次数为 $O(\mathrm{lb}\ W)$,存储空间复杂度为 $O(N\ \mathrm{lb}\ W)$。该算法具有良好的扩展性,但引入了大量的标记来进行二分查找以避免回溯,因此增加了存储的复杂度。高莹等[2]提出

收稿日期:2012-11-06;**修回日期**:2012-12-18。
基金项目:国家 863 计划项目(2012AA012803,2013AA014703);中国科学院战略性科技先导专项(XDA06030200)。
作者简介:杜飞(1982 -),男,山西太原人,工程师,硕士,CCF 会员,主要研究方向:网络与信息安全;　董治国(1972 -),男,吉林农安人,高级工程师,主要研究方向:核电 DCS 系统级设备;　苗琳(1983 -),女,北京人,工程师,硕士,主要研究方向:网络与信息安全;　庹宇鹏(1984 -),男,河北廊坊人,助理研究员,博士研究生,CCF 会员,主要研究方向:网络协议识别与异常检测。

的分段 Hash 方法提出了针对不同长度前缀设计不同 Hash 算法的思想,并针对 IPv6 的地址结构做了分段;该算法性能较好,但设计比较复杂。王亚刚等[3]提出了基于哈希表和树位图的 IPv6 地址查找算法,依据 IPv6 的路由地址分布规律来优化查找路径,但对前缀覆盖和 Hash 查找的处理不足;基于哈希技术的查找算法对路由项的增减比较容易。Hash 函数的优点是时间复杂度为 $O(1)$, Hash 函数的缺点是发生冲突时比较耗时,一个 Hash 表只能处理同一长度的前缀。

步长大于 1 的 Trie 树可以称之为多比特 Trie 树(Multibit Trie)[8-10],若一棵多比特 Trie 树的步宽为 k, 则每个节点的最大分支数可为 2^k。二叉 Trie 树实际上就是查找步长为 1 的 Trie 树。步宽如果是固定的,则不能支持任意长度的地址前缀,需要对有些特定的前缀进行扩展。这样实现相对简单,但浪费存储空间。该算法的查找复杂度为 $O(W/k)$,存储复杂度为 $O(2^kNW/k)$。可见,步长 k 的选择要在查找复杂度和存储复杂度之间做折中,因此多分支 Trie 树结构不能独立作为 IPv6 路由查找算法,必须与其他技术相结合。

1.1　相关定义

定义 1　地址前缀 $P(p,l)$。表示长度为 l 的前缀 P,p 为该前缀的二进制比特串。

定义 2　子前缀 $P_s(p,l_s)$。设 $P(p, l)$ 表示长度为 l 的前缀 P,则将 P 中前 l_s 个比特($l_s < l$) 表示的前缀称为 P 的长度为 s 的子前缀。例如,对于前缀 P (1100101, 7) 而言,$P_s(P,3)$ 表示的前缀为 110。

定义 3　前缀区间。在路由表中,每一个前缀代表一个地址区间,包含一段连续的地址,称其为前缀区间。以地址长度为 5 的前缀为例,010 * 的起始点为 01000;结束点为 01011,区间范围表示[01000,01011]。

定义 4　前缀层次。路由表的前缀之间存在层次关系,即一个前缀的区间是另外一个前缀区间的子集或超集。前缀之间的这种包含关系就是它们之间的层次关系。层次关系反映了路由表被子网化的程度,最高层没有前缀,用 L0 表示;次高层用 L1 表示,最底层没有子集。

1.2　IPv6 前缀分布规律

本文以澳大利亚 potaroo[11] IPv6 骨干网 BGP 路由表前缀为例,分析该路由表的前缀分布规律。如图 1、图 2 所示,可以得出:

1)IPv6 路由前缀的长度主要集中在 32、40 和 48 比特,占前缀总量的 85.21% 以上,低于 16 比特和超过 64 比特的都比较少。前 16 比特共有 9 种不同的取值:2001,2002,2003,2400,2404,2601,2800,2a01,3ffe。这 16 比特的取值增长较慢,而其中 90% 以上的前缀又以 2001 作为开头。

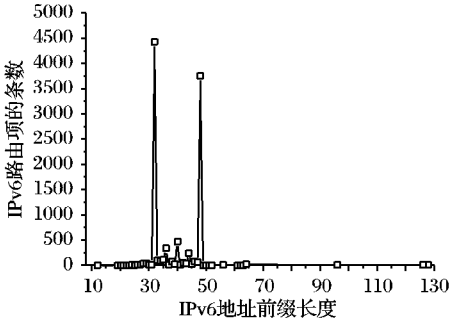

图 1　IPv6 前缀长度分布规律

2)路由前缀之间普遍存在着层次关系,根据路由表的统计,IPv6 的路由表最多包含 3 个层次,超过 70% 的前缀处于 L0 层。

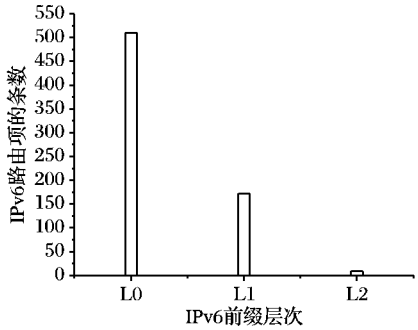

图 2　IPv6 前缀层次分布规律

2　pHTMT 路由查找算法

2.1　基于无冲突 Hash 表和 multibit-trie 的两级查找算法

针对上文对 IPv6 路由表分析的特点,本文提出了基于无冲突 Hash 表和 multibit-trie 相结合的两级 IPv6 路由查找算法。该算法主要针对骨干网上全球单播路由地址进行设计,因为前 3 比特为 001,采用 4 ~ 16 比特(IPv6 前缀均大于 16 比特)的值作为一个线性索引表,可包含 8 192 个表项。每个表项对应一个前缀区间,这是一个静态数组,可以进一步提高查找效率。将 17 ~ 64 比特的前缀根据层次关系和区间划分存储在 3 个无冲突 Hash 表中,分别用 HT32、HT40 和 HT48 表示,剩余不足 32、40 和 48 比特的部分采用 Multibit-Trie 存储。该算法的数据结构如图 3 所示。

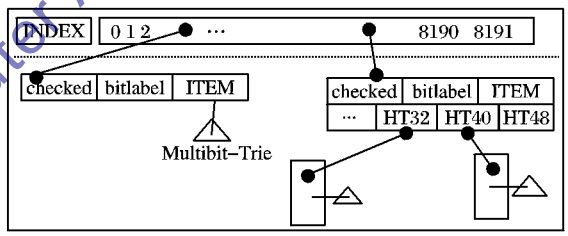

图 3　两级算法数据结构

2.2　路由表项的添加算法

初始化路由表为空,包括索引表的指针。在添加路由表项之前,对路由表前缀构建层次树。以 A:2001:300::/32; B:2001:300:900::/40; C:2001:300:903::/48 为例,其层次树的前缀区间包含关系为 A 为 L0 层,B 为 L1 层,C 为 L2 层,即 A 前缀区间包含 B,B 包含 C。添加算法如下:

1)以前缀的第 4 ~ 16 比特的值作为索引,在索引表中得到该表项的子区间。

2)假设表项前缀为 x,如果其前缀小于 32 比特,则将其剩余的 $x-16$ 以步宽 *skip* 中的值存储在多比特树中;若前缀 x 为 32、40、48 的区间点,则将其存储在对应的无冲突 Hash 表中;如果 $32 < x < 40$,则将 $x-32$ 的前缀存储于对应表项的多比特树中;对于 $40 < x < 48$ 和 $48 < x < 64$ 的情况,做类似的处理。

3)查看其对应的层次,如果属于最低层,则在其上一级层次的 Hash 表中存储其值。以表项 B 为例,应该存储在 HT40 中,但其属于 A 的子集,因此表项 B 的前 32 比特也存储在 HT32 中。

对于存储在 Multibit-Trie 中的前缀来说,采用固定步宽的策略,步宽 *skip* 的值和树的根节点 *root* 均存储于 hashitem 中,Multibit-Trie 中存储比特串的最大长度为 8 或者 16。

2.3　IPv6 地址的查找算法

IPv6 的地址查找算法如下:

1)以前缀的第 4 ~ 16 比特的值作为索引,得到该表项的子区间。

2)查看其 checked 值,若为 0 表明该区间中前缀长度不存在超过 32 比特的值,在对应的多比特树中查找并返回下一跳的值;若为 1,转到 3)。

3)大部分前缀长度为 32 比特,因此以 32 比特作为入口地址来查找,如果在 HT32 中找到该表项,并且对应的多比特树为空,前缀层次为最底层,则前缀长度为 32 比特,返回下一跳的值。

a)若对应多比特树非空,前缀层次为最底层,则最长前缀可能在对应的多比特树中,在多比特树中继续查找并返回结果;

b)若对应多比特树为空,前缀层次为非最底层,则可能有更长的前缀在 HT40 中,在 HT40 表为入口地址查找;

c)如果在 HT40 中找不到该表项,则在 HT48 表为入口地址查找。

4)在三个 hash 表中均未找到匹配的表项,则匹配默认路由。

在查找算法中,对于在 Multibit-Trie 中的前缀,步宽 *skip* 取值依据不同的前缀而不同,深度不超过 4,由于前面的前缀分布在不同的索引区间,因此保证了 Multibit-Trie 结构的简洁性。

2.4 无冲突 Hash 表的构建

该算法中,性能的高效与否取决于无冲突 Hash 表的构建,在本文中,采用动态加载因子和双 Hash 框架来解决这一问题。如图 4 所示:$Hash1$ 定位索引数组的位置,其中包含两个值,一个为 $Hash2$ 的算法因子 α,另外一个值为存储 *item* 的数组指针。当发生冲突时,即当 $x \neq y$,$Hash1(x) = Hash1(y)$ 且 $Hash2(x + \alpha) = Hash2(y + \alpha)$,调整算法因子 α 使得 $Hash2(x + \alpha) \neq Hash2(y + \alpha)$,即使第二层的 Hash 不发生碰撞。为了提高查找速度,$Hash1$ 和 $Hash2$ 的计算复杂度不能太高,应该尽可能地简单高效,算法因子 α 每个表项都不相同,经验值取不同的素数。无冲突 Hash 表的查找时间复杂度为 $O(1)$,存储空间复杂度为 $O(n)$,其中 n 为存储元素的个数,存储开销是线性增长的。实验中内存消耗和理论中略有偏差,但在可接受的范围之内。由于前面的索引数组对前缀的分布有了区间的划分,进一步减少了无冲突 Hash 表的内存消耗。

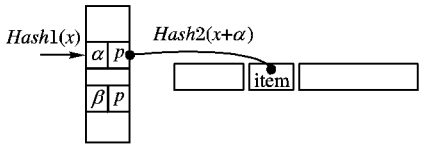

图 4 无冲突 HASH 表算法结构

3 算法性能分析

为了评价算法的性能,本文以查找路径长度为指标,定义一次二进制比特串的移动为一个路径单位,可以是 16 个比特,也可以是 4 个比特,依据不同的位置而不同。当真实的 IPv6 单播地址的前缀与路由表前缀分布相近时,算法的平均查找路径长度 L = 索引路径 + 概率 P_{L0} * HT32 路径 + 概率 P_{L1} *(HT32 路径 + HT40 路径) + 概率 P_{L2} *(HT32 路径 + HT40 路径 + HT48 路径) + 概率 P_{trie} * Multibit-Trie 路径。将索引的查找路径设为 1,Hash 操作的查找路径为 b,根据前缀分布规律,32、40 和 48 比特的前缀概率为 43.43%、4.61% 和 36.87%,多比特树的查找路径为 r,修正参数 C。则平均查找路径估算近似为:

$$L = 1 + 73.8\% * 0.4343b + 24.89\% *(0.4343 + 0.0461)b + 1.3\% *(0.4343 + 0.0461 + 0.1509)b + 0.1509r + C = 1 + 0.4511b + 0.1509r'$$

其中 $0.1509r' = 0.1509r + C$,由于前缀层次反映了子网化的程度,所以随着 IPv6 的进一步深入应用,子网化程度加深,使得 L0、L1 和 L2 的概率趋于相等,此时的查找路径为:

$$L' = 1 + 33.3\% * 0.4343b + 33.3\% *(0.4343 + 0.0461)b + 33.3\% *(0.4343 + 0.0461 + 0.1509)b + 0.1509r + C = 1 + 0.5880b + 0.1509r'$$

对比 L 和 L' 可知,该算法在 IPv6 的路由查找上具有很好的稳定性。随着子网化的深入,对查找性能的影响较小。对于静态索引数组而言,只有当 IPv6 的规模比较大时,其查找效率相对于占用的内存优势才会体现出来。相对二分查找 Hash 表和采用 XOR-Folding 技术的 Hash 表,在查找性能上无冲突 Hash 表有很大的优势,但初始化时间相对较长。

4 实验及结果分析

为了验证该算法的性能,实验环境为内存 16 GB,CPU 24 核 2.4 GHz,CentOS5.5 操作系统。实验数据采用了真实路由表数据和模拟数据。

为了验证算法的性能,通过比较平均查找时间和内存消耗来分析结果。表 2 中的数据显示了基于 LC-trie 的算法、HTMT(有冲突的 Hash 表)和 pHTMT 算法在查找时间上的对比实验结果。路由表的数据前缀分布规律和层次结构与 potaroo 路由器中的前缀分布规律相吻合。测试数据来自骨干网中的真实 IPv6 地址(2 546 个)和随机产生的 IPv6 地址(65 536 个)。

表 2 三种算法平均查找时间对比 ns

算法	2 546 个真实 IPv6 地址	65 536 个随机 IPv6 地址
LC-trie	82	83
HTMT	37	40
pHTMT	20	31

由表 2 中数据可以看出,无论真实 IPv6 还是随机 IPv6,LC-trie 算法的平均查找时间相对稳定,而 HTMT 和 pHTMT 对于 IPv6 地址的前缀分布比较敏感,但其平均查找时间均比 LC-trie 算法要好。由于 pHTMT 算法没有冲突发生,因此比 HTMT 的效果要好。

表 3 中的数据显示了 LC-trie、HTMT 和 pHTMT 算法在内存占用上的对比实验结果,路由表中的路由条数分别为 2 546、16 384 和 65 535。

表 3 内存占用对比 KB

算法	2 546 条路由	16 384 条路由	65 536 条路由
LC-trie	1 034	6 380	22 576
HTMT	2 774	3 156	8 243
pHTMT	1 799	2 132	4 660

由实验数据可以看出,规模比较小时,HTMT 和 pHTMT 的内存占用比 LC-trie 大,这是由于使用了一个 8 192 项的索引表。路由规模小的时候,索引表占用内存的比例较大,但它是静态的,不会随路由规模的扩大而改变。因此,当路由表规

(下转第 1202 页)

将提供哈希函数、SFC 和 DHT Overlay 的封装类，以实现哈希函数、SFC、DHT Overlay 的灵活替换；2）保证已知组合服务数据键的时效性。

参考文献：

[1] 邓水光，黄龙涛，尹建伟，等. Web 服务组合技术框架及其研究进展[J]. 计算机集成制造系统，2011，17(2)：404－412.

[2] ZENG L，BENATALLAH B，NGU A H H，*et al*. QoS-aware middleware for Web services composition[J]. IEEE Transactions on Software Engineering，2004，30(5)：311－327.

[3] YU T，ZHANG Y，LIN K-J. Efficient algorithms for Web services selection with end-to-end QoS constraints[J]. ACM Transactions on the Web，2007，1(1)：1－26.

[4] YE X，MOUNLA R. A hybrid approach to QoS-aware service composition[C]// Proceedings of the 6th IEEE International Conference on Web Services. Washington，DC：IEEE Computer Society，2008：62－69.

[5] CHENG R，SU S，YANG F，*et al*. Using case-based reasoning to support Web service composition[C]// Proceedings of the 6th International Conference on Computational Science. Washington，DC：IEEE Computer Society，2006：87－94.

[6] STOICA I，MORRIS R，LIBEN-NOWELL D，*et al*. Chord：a scalable peer-to-peer lookup protocol for Internet applications[J]. IEEE/ACM Transactions on Networking，2003，11(1)：17－32.

[7] MAYMOUNKOV P，MAZI D. Kademlia：A peer-to-peer information system based on the XOR metric[C]// Proceedings of the 1st International Workshop on Peer-to-Peer Systems. London：Springer-Verlag，2002：53－65.

[8] ZHANG Y，LIU L，LI D，*et al*. DHT-based range query processing for Web service discovery[C]// Proceedings of the 7th IEEE International Conference on Web Services. Washington，DC：IEEE Computer Society，2006：87－94.

[9] TANG Y，XU J，ZHOU S，*et al*. A lightweight multidimensional index for complex queries over DHTs[J]. IEEE Transactions on Parallel and Distributed Systems，2011，22(12)：2046－2054.

[10] MOKBEL M F，AREF W G. Space-filling curves[M]// SHEKHAR S，XIONG H. Encyclopedia of GIS. Heidelberg：Springer，2008：1068－1072.

[11] CHEN X，ZENG H，WU T. Decentralized orchestration with local centralized orchestration for composite Web services[C]// Proceedings of the 11th International Conference on Parallel and Distributed Computing，Applications and Technologies. Washington，DC：IEEE Computer Society，2010：255－260.

[12] MOKBEL M F，AREF W G. Irregularity in high-dimensional space-filling curves[J]. Distributed and Parallel Databases，2011，29(3)：217－238.

[13] KAZMI I，BUKHARI S F Y. PeerSim：an efficient scalable testbed for heterogeneous cluster-based P2P network protocols[C]// UkSim 2011：the 13th International Conference on Computer Modelling and Simulation. Washington，DC：IEEE Computer Society，2011：420－425.

[14] 张龙昌，邹华，杨放春. 一种基于多 QoS 注册中心和模型异构的 Web 服务选择算法[J]. 电子与信息学报，2011，33(1)：168－174.

[15] 张莹，黄厚宽，杨冬，等. 基于 Chord 的带有 QoS 的语义 Web 服务发现方法研究[J]. 电子与信息学报，2009，31(3)：711－715.

（上接第 1196 页）

模较大时，pHTMT 算法在内存消耗方面具有一定优势。理论上来说无冲突 Hash 表要比常规的 Hash 表占用更多的内存，但由于索引表的作用，进一步弱化了无冲突 Hash 表对内存的占用。

表 4 分析 pHTMT 算法的平均查找路径，路由表的规模为 2546 条，测试数据为真实的 IPv6 地址，路由表的层次树结构 L0 分别占 70%，60% 和 50%。

表 4 pHTMT 算法的平均查找路径对比（路由表规模 2546）

数据	L0 占用率/%	总查找路径数	平均查找路径数
真实数据	70	3810	1.49647
	60	3849	1.51178
	50	4159	1.63354
随机数据	70	3827	1.50314
	60	4405	1.73016
	50	5093	2.00004

由表 4 可知：

1）随着 L0 的占用率降低，平均查找路径数随着降低，但下降程度不会太大，因此随着路由规模扩大，复杂性提高，该算法依然可用；

2）对于随机数据的平均查找路径要比真实的效果差些，这是由查找入口的选择每次都是 HT32 所导致的。

5 结语

本文把无冲突 Hash 表与多比特树算法相结合，提出了 pHTMT 算法。该算法能够结合 IPv6 路由前缀分布规律和前缀层次树的特点，并采用完美 Hash 技术处理冲突，对 IPv6 路由查找有较好的效果。随着子网化的深入、路由表规模的扩大，算法具有良好的稳定性。

参考文献：

[1] 陈蹊，赵跃龙. 多分枝 Trie 树路由查找算法研究[J]. 电子设计工程，2010，18(3)：4－5，8.

[2] 高莹，王贺明，陈强. 采用分段哈希方法的 IPv6 路由查找算法研究[J]. 计算机工程与设计，2010，31(22)：4790－4793.

[3] 王亚刚，杜慧敏，杨康平. 使用 Hash 表和树位图的两级 IPv6 地址查找算法[J]. 计算机科学，2010，37(9)：36－39，80.

[4] 孙庆南，鲁士文. 一种改进的二分法 IPv6 路由查找算法[J]. 计算机工程，2006，32(18)：35－38.

[5] 崔尚森，张白一. 一种基于哈希表和 Trie 树的快速 IP 路由查找算法[J]. 计算机工程与应用，2005，41(9)：156－158.

[6] WALDVOGEL M，VARGHESE G，TURNER J，*et al*. Scalable high speed IP routing lookups[J]. ACM SIGCOMM Computer Communication Review，1997，27(4)：25－36.

[7] EATHERTON W. Hardware based Internet protocol prefix lookups[D]. St. Louis：Washington University，1999.

[8] EATHERTON W，VARGHESE G，DITTIA Z. Tree bitmap：Hardware/software IP lookups with incremental updates[J]. ACM SIGCOMM Computer Communication Review，2004，34(2)：97－122.

[9] LI Y K，PAO D. Address lookup algorithms for IPv6[J]. IEEE Proceedings of Communications，2006，153(6)：909－918.

[10] HUSTON G. Analyzing the Internet's BGP routing table[J]. The Internet Protocol Journal，2001，4(1)：2－15.

[11] AS2IPv6 BGP Table Statistics [EB/OL]. [2012－09－12]. http://bgp.potaroo.net/v6/as2.0/index.html.