·移动互联与通信技术· 文章编号:1000-3428(2017)03-0099-06 文献标志码:A 中图分类号:TP393

基于代数决策图的路由查找算法

徐周波,胡 魁,常 亮,古天龙

(桂林电子科技大学 广西可信软件重点实验室,广西 桂林 541004)

摘 要: 为解决路由查找过程中路由表项数不断增加导致存储冗余大和查找效率低的问题,在代数决策图(ADD)的基础上,提出一种改进的路由查找算法。根据符号算法的特性对路由表项进行伪布尔函数表示,综合考虑路由表结构特征和符号算法的优势,基于 ADD 结构构建基于前缀的路由表,并给出路由表更新、删除、查找算法。通过国际项目管理协会提供的开源路由表进行实验仿真,结果表明该算法能够有效减少路由表操作时的内存访问次数,节省路由表存储空间。

关键词: 路由表;路由查找;代数决策图;符号算法;最长前缀匹配;伪布尔函数

中文引用格式: 徐周波,胡 魁,常 亮,等. 基于代数决策图的路由查找算法[J]. 计算机工程,2017,43(3):99-104.

英文引用格式: Xu Zhoubo, Hu Kui, Chang Liang, et al. Routing Lookup Algorithm Based on Algebraic Decision Diagram[J]. Computer Engineering, 2017, 43(3): 99-104.

Routing Lookup Algorithm Based on Algebraic Decision Diagram

XU Zhoubo, HU Kui, CHANG Liang, GU Tianlong

(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China)

【Abstract】 In order to solve the problem of high storage redundancy and low search efficiency caused by the increasing number of routing table entries in the proess of routing lookup, an improved routing lookup algorithm based on Algebraic Decision Diagram (ADD) is proposed. According to the characteristic of symbolic algorithm, routing table entries are expressed as a pseudo Boolean function. Considering the features of routing table structure and the advantages of symbolic algorithm, a prefix-based routing table is constructed by ADD structure, and the algorithm of routing table update, deleting and look-up is also given. With the open source routing table provided by International Project Management Association (IPMA), the experimental results show that the algorithm can effectively reduce the number of memory access in routing table operation and save the routing table storage space.

【Key words】 routing table; routing lookup; Algebraic Decision Diagram (ADD); symbolic algorithm; longest prefix matching; pseudo Boolean function

DOI: 10.3969/j.issn.1000-3428.2017.03.018

0 概述

目前,网络系统主要分为 3 类:电信网,有线电视网和计算机网。而随着电信和信息技术的发展,三网融合逐渐成为未来网络技术的发展方向。具体表现方式为各个网络之间相互联通、资源共享,能在统一的 IP 技术标准上为用户提供广播电视、语音和数字媒体等服务。这意味着 IP 协议将得到更广泛的应用,同时 IPv6 协议的应用将使 IP 地址的长度和数量都大幅度膨胀,这对路由器性能提出了新的要求,而优化路由表存储结构和路由查找算法是提升路由器性能的关键。总的来说,对路由查找算法的研究主要有2 个方向:硬件算法和软件算法。硬件算法[1-3]使用并行技术,具有结构简单、查找速度快的优点,但是存在存储空间大、查找过程功耗大、价格昂贵、扩展性能低的问题。软件算法占用的存储空间小,具有灵活性高的特点,本文主要研究软件查找算法。

基金项目: 国家自然科学基金(61262030,61572146,61363030);广西自然科学基金(2015GXNSFAA139285,2014GXNSFAA118354);广西可信软件重点实验室基金;广西高等学校高水平创新团队及卓越学者计划项目。

作者简介: 徐周波(1976—),女,副教授、博士,主研方向为符号计算、智能规划;胡 魁,硕士研究生;常 亮,教授、博士;古天龙,教授、博士生导师。

收稿日期: 2016-03-14 **修回日期:** 2016-04-15 **E-mail:** xzbli_11@guet.edu.cn

软件实现的路由查找算法主要有基于地址前缀长度和基于 Trie 树 2 类。基于地址前缀长度的路由查找算法利用前缀长度不同的特性,将路由查找表(Lookup Table,LUT)分成多个集合,典型算法有哈希查找算法[4]、二分查找算法[5]。这类算法优点是查找时间只与前缀长度有关,但是实际应用中路由表是需要动态更新的,这样就很难找到一个能一直满足条件且没有冲突的哈希函数[6]。基于 Trie 树结构的路由查找算法[7-10]根据前缀的二进制值的每一个比特位构建节点,每个节点中都能够保存路由转发信息,典型的 Trie 树查找算法有路径压缩 Trie 树、多分支 Trie 树等。基于 Trie 树的算法能够利用公共前缀减少存储空间,并且算法简单易于实现,但是其查找过程需要大量的内存访问,同时算法中可能有回溯操作影响查找效率[11]。

有序二叉决策图(Ordered Binary Decision Diagram,OBDD)是布尔函数的一种规范表达形式,具有高紧凑性和易操作的特点,并在符号模型检验、符号优化等领域得到广泛应用。结合符号算法,文献[12]中提出一种结构紧凑的基于符号二叉决策图(Ordered Binary Decision Diagram,BDD)的路由表结构和算法,该算法能够利用符号技术知识共享的特性约减路由表项中的冗余信息,达到减小路由表存储空间的目的。但是该算法也存在局限性,使用 BDD 来表示路由表需要对端口号进行二进制编码,对每个编码位都建立一个 BDD,这反而会在一定程度上增加节点数量和提高算法操作复杂度。相对于 BDD,代数决策图(Algebraic Decision Diagram,ADD)是 BDD 的扩展形式,除了具有 BDD 结构紧凑的特性,还具有描述伪布尔函数的能力。利用该图可以表示整个路由表,且能对路由表存储空间进一步优化。因此,本文将 ADD 结构引入到路由查找算法中,进一步减小路由表的存储空间,优化路由查找性能。

1 相关知识

1.1 最长前缀匹配

IETF(The Internet Engineering Task Force)于 1993 年提出无类域间路由(Classless Inter-domain Routing,CIDR)结构,传统路由查找算法不再适用,最长前缀匹配(Longest Prefix Matching,LPM)成为路由查找算法的核心[13]。最长前缀匹配是在互联网协议(IP)中查找最长前缀路由的算法,假设储存 n 条路由表项的路由表中存在 m 条表项与目的 IP 地址匹配($0 \leqslant m \leqslant n$),那么子网掩码最长的表项称为最长前缀路由。

路由器中有一个记录目的 IP 地址和下一跳端口对应关系的数据库,即路由表。路由表项前缀长度范围是 0 bit ~ 32 bit,表 1 是一个简单的前缀路由表。路由表中包含路由前缀(Prefix)和下一跳端口(Next-hop Port,NHP)。表 1 中第 1 条路由表项{01*,2}表示前缀长度为 2,下一跳端口为 2,掩码形式表示为{64/2,2}。

表 1 前缀路由表

前缀	下一跳端口
01*	2
010*	1
10*	2

最长前缀匹配的过程就是从路由表中匹配最优的路由表项的过程。对一个传入的数据包,路由器提取目的 IP 地址并与路由表中的所有表项进行比较,选择匹配的最长前缀所对应的下一跳端口作为这个数据包的输出端口,如图 1 所示。

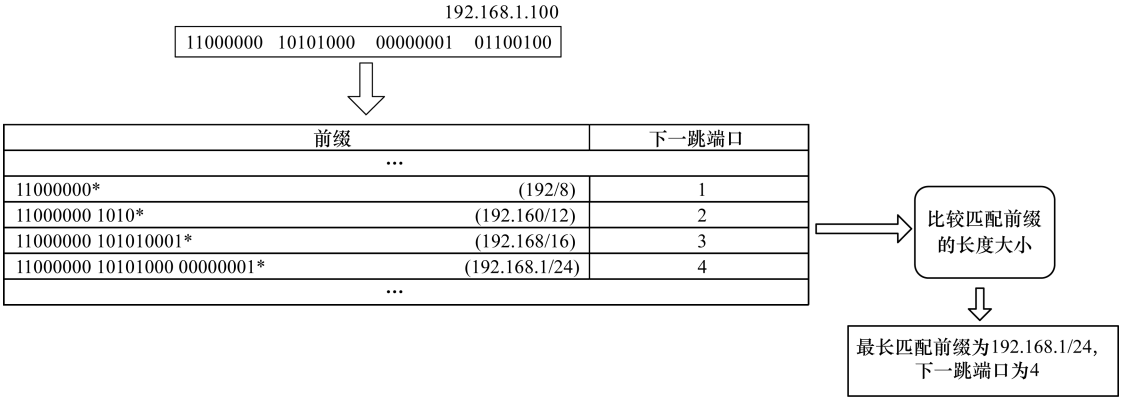


图 1 最长前缀匹配

图 1 中传入数据包的目的 IP 地址为 192.168.1.100,路由表中有 4 条路由与之匹配:192/8,192.160/12,192.168/16 和 192.168.1/24,其中匹配项中前缀最长的路由为 192.168.1/24,该条表项就是数据包下一跳的最优匹配项,端口 4 即为下一跳输出端口。此外,如果路由表中没有任何一条与输

入目的 IP 地址匹配的表项,那么数据包从默认端口输出,前缀长度默认为 0。

1.2　代数决策图

结合代数决策图(ADD)[14]紧凑性高、易操作的特点构建路由表能够在很大程度上删除冗余节点,降低路由表所需存储空间。同时,ADD 对伪布尔函数的描述能力使其在 IPv6 协议上同样适用。

一个 ADD 就是表示基于变量 $x_1,x_2,\cdots,x_n$ 的一簇伪布尔函数 $f_i:\{0,1\}n\to S$ 根节点的一个有向无环图,它满足:

1)S 为 ADD 代数结构的有限值域,且 $S\subseteq\mathbb{Z}$。

2)ADD 中有 3 类节点:根节点,内部节点和叶子节点。没有父节点或者输入边的节点称为根节点;没有子节点或者输出边的节点称为叶子节点,叶子节点集合为 T,对 $\forall t\in T$,均被标识为值域 S 中的一个元素 $s(t)$;其他的节点则称为内部节点。

3)ADD 中的每一个非叶子节点 u 都具有四元组属性(f^u,var,$left$,$right$),其中,f^u 表示节点 u 所对应的一个伪布尔函数分量,若 u 是根节点,则 f^u 表示整个伪布尔函数;var 表示当前节点所对应变量的取值;$left$ 是指 var 取值为 0 时所指向的左分支节点,通常用虚线来表示左分支;$right$ 是指 var 取值为 1 时所指向的右分支节点,通常用实线来表示右分支。

4)在 ADD 结构的路径中,每个变量 x_i 都会至少出现一次。

5)在给定变量序 $\pi:x_1<x_2<\cdots<x_n$ 下,每条路径中变量的次序都必须符合该次序。

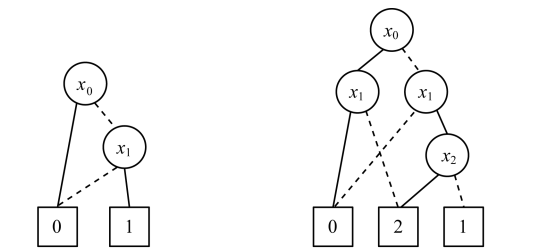
由 ADD 结构构建路由表,输出弧表示目的 IP 地址,使用合并、删除规则进一步减少冗余节点数量。同时,在网络路由器中下一跳端口的数量一般不会超过 256 个[10],路由表中的任何一个下一跳端口都可以使用一个叶子节点表示。例如:IPMA(International Project Management Association)提供的 MAE-West 路由表中的下一跳端口数量不超过 64 个。

2　基于符号 ADD 的路由查找算法设计

2.1　路由表构建

从表 1 中可知路由表项是二元结构{Prefix,NHP},假设路由表项的前缀二进制长度为 i。按照固定变量序 $x_0<x_1<\cdots<x_i$ 使用伪布尔变量从左到右逐一表示路由表中的前缀二进制值,如果当前二进制位为 0,那么用 $r_i=x_i'$ 表示;如果二进制值为 1,那么用 $r_i=x_i$ 表示。前缀对应的下一跳端口 $v\in V$(V 是下一跳端口集)表示叶子节点,这样每一条前缀路由表项都可以使用伪布尔函数 $f(x_0,x_1,\cdots,x_i,v)=r_0\wedge r_1\wedge\cdots\wedge r_i\wedge v$ 表示,其中 $i\in[0,32]$。

在固定变量序 $x_0<x_1<\cdots<x_i$ 下,每个表项使用一个伪布尔函数表示。以表 1 中第 1 条表项为例,伪布尔函数表示为 $f(x,v)=2\cdot x_0'\cdot x_1$,其中 x_0 是前缀最高有效位,根据变量序将 $f(x,v)$ 香农分解可得到 ADD 结构如图 2(a)所示。图中叶子节点 0 为默认下一跳端口,对于路由表中不存在的路由表项,系统会选择默认端口作为输出端口;若没有默认端口,则直接丢弃。表 1 中共有 3 条路由,伪布尔函数表示为 $f(x,v)=2\cdot x_0'\cdot x_1+x_0'\cdot x_1\cdot x_2+2x_0\cdot x_1'$,同样对其香农分解构造 ADD 表示如图 2(b)所示。



(a)$f(x,v)=2\cdot x_0'\cdot x_1$的ADD　(b)$f(x,v)=2\cdot x_0'\cdot x_1+x_0'\cdot x_1\cdot x_2+2x_0'\cdot x_1$的ADD

图 2　路由表项的 ADD 表示

从图中可以看出构建如表 1 所示的路由表仅使用 7 个节点,基于 ADD 结构构建的路由表有效利用知识共享的特性,降低了路由表的存储空间,最坏情况下的节点个数为 2^n(n 表示 ADD 的深度),但实际使用中远小于该数。

2.2　路由查找算法设计

基于 2.1 节提出的路由表构建方法,给出本文提出的基于符号 ADD 的路由查找算法。整个算法分为 3 个部分:1)路由表的删除和更新;2)路由表的化简;3)查找下一跳端口。

定义 1　若已知具有相同变量序 $\pi:x_1<x_2<\cdots<x_n$ 的布尔函数 f,g 和 h,则 $ITE(f,g,h)=f\cdot g+f'\cdot h$[14]。

如表 2 所示,实际的路由表由三元素组成:目的 IP 地址,IP 地址掩码(Mask)和下一跳端口(NHP),IP 和 Mask 构成前缀[15]。

表 2　MAE-EAST 路由表项

目的 IP 地址	IP 地址掩码	下一跳端口
100663296	8	3223958001
151126016	16	3223957901
76152832	17	3223957870
…	…	…

路由表构建的基本思想是将路由表项转化成 ADD 表示,继而将生成的 ADD 添加进路由表中。所以,算法第 1 步是将路由表项转化为 ADD,下面

给出路由表项转化为 ADD 算法的伪代码。

算法 1 路由表项转化为 ADD 的算法 *PrefixMaskToAdd*

输入 *ip*,*mask*,*port*

输出 *out*

```
1. 添加叶子节点 port;
2. for(index 从 0 到 mask){
3.       添加索引为 index 的变量 var;
4.       if(ip 的第 index 位为 0)
5.          var = ITE(var,1,0);
6.       out = out ∧ var;
7.   }
8.   return out;
```

算法 1 输入参数 *ip* 是 32 位的目的 IP 地址,*mask* 为前缀掩码,*port* 为下一跳端口。假设路由表项的伪布尔函数为 $f(x)=x_0'\cdot x_1\cdot x_2$,按照变量序从 x_0 位开始添加节点,若第 *index* 个变量为 x_i',则对其进行 *ITE*(x_i,1,0)运算得到 *var'*,如算法 1 第 5 行所示。每层节点连接弧的 0 分支表示目的 IP 地址对应伪布尔函数中的 x_i',反之连接弧的 1 分支对应 x_i,将各个变量依次合取最终得到该条表项的 ADD 表示。

2.2.1 路由表删除和更新

路由表会随着路由协议的改变而不断更新,在更新路由表时,若该条路由表项已经存在,则需要将其删除,下面给出本文提出的路由表删除算法。

算法 2 路由表删除算法 *deleteEntrie*

输入 *ip*,*mask*

输出 *m_fun*

```
1.   tmp = m_fun;
2.   for(index 从 0 到 mask){
3.       fatherNode = tmp;
4.       if(ip 二进制第 index 位不为 0){
5.          if(tmp 是叶子节点)
6.             break;
7.          direction = 1;
8.          tmp 跳向左边弧指向的节点;
9.       }else{
10.         if(tmp 是叶子节点)
11.            break;
12.         direction = 0;
13.         tmp 跳向右边弧指向的节点;
14.      }
15.  }
16.  if(tmp 是叶子节点){
17.      if(终节点值非 0){
18.      entrie = prefixMaskToAdd(ip,mask,port);
19.      AddMinus(m_fun,entrie);}
20.  }else{
21.      if(direction)
22.         删除 fatherNode 左边弧指向的节点;
23.      else
24.         删除 fatherNode 右边弧指向的节点;
25.  }
```

算法 2 中 *m_fun* 是路由表根节点,路由表初始下一跳端口为默认跳(默认跳为 0)。在删除路由表项之前首先判断当前目的 IP 地址前缀是否存在,具体分为 3 种情况:

1)前缀指向叶子节点且端口号为默认跳,则路由表中没有相应路由信息,结束返回。

2)前缀指向叶子节点且端口号非默认跳,将该条路由从路由表中删除,如算法 2 第 19 行所示。

3)前缀指向中间节点,后面所有节点均需要删除,根据 *fatherNode* 记录的父节点位置和 *direction* 记录的输出弧方向对后面的节点递归删除,如算法第 21 行~第 24 行所示。

在更新路由表项时,首先判断当前路由表项是否存在:若存在,则调用删除算法删除当前路由表项;然后调用 *PrefixMaskToAdd* 函数生成表项的 ADD 表示 *g*;最后使用 $f=f\vee g$ 操作将该条路由表项 *g* 添加进路由表 *f*。

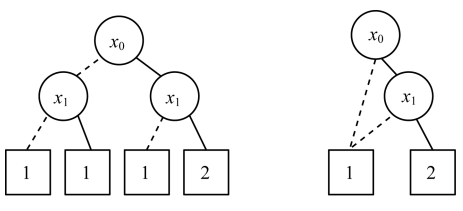
2.2.2 路由表化简

使用本文提出的路由表删除和更新算法可以用 ADD 表示每一条路由表项。然而,根据符号算法的规则能够对路由表进行更深层次的化简,进一步减少路由表节点个数,如表 3 所示。

表 3 路由表化简

编号	前缀	下一跳端口
1	0*	1
2	10*	1
3	11*	2

根据上文所述的构建方法,表 3 中路由表的伪布尔函数表示为 $f(x,v)=x_0'x_1+x_0'x_1'+x_0x_1'+2x_0x_1$,转化成 ADD 结构如图 3(a)所示使用了 7 个节点。将相同节点进行合并删除多余的节点,最终路由表仅使用 4 个节点,对应的伪布尔函数化简结果为 $f(x,v)=x_0'+x_0x_1'+2x_0x_1$,如图 3(b)所示。



(a)路由表化简前 (b)路由表化简后

图 3 路由表化简的 ADD 表示

比较化简前后节点数量可知,本文提出的路由表结构能够有效删除冗余节点,使路由表结构更加紧凑,减小系统内存占用。当使用 32 bit 的 IP 地址时,对 MAE-EAST. 971030 中的路由表进行构建仅需要 46 015 个节点,相比于理论上最坏情况的 80 亿个节点上限是微乎其微的。

2.2.3　路由查找算法

基于 2.1 节提出的路由表建模方法,本文构建了一个符号 ADD 结构的路由表。路由查找算法是在 ADD 路由表基础上的下一跳端口的匹配求解过程。为了方便说明,下面给出基于符号 ADD 的路由查找算法。

算法 3　基于符号 ADD 的路由查找算法 *GetPort*

输入　*ip*

输出　*port*

```
1.  for(index 从 0 到 IP_LENGTH){
2.      if(ip 二进制第 index 位不为 0){
3.          if(是叶子节点)
4.              break;
5.          跳向左边弧指向的节点;
6.      }else{
7.          if(是叶子节点)
8.              break;
9.          跳向右边弧指向的节点;
10.     }
11. }
12. 输出 tmp 节点值 port;
```

在路由查找算法中,节点中 index 中记录当前节点在固定变量序中的索引位置。如算法 3 第 7 行所示,根据当前路由节点的 *index* 获取 IP 中相应位的值 *ip*[*index*],具体操作如下:

1)若当前节点是终节点,则跳向步骤 4)。

2)若当前节点不是终节点,且 *ip*[*index*] = 1,则跳向右孩子,返回步骤 1)。

3)若当前节点不是终节点,且 *ip*[*index*] = 0,则跳向左孩子,返回步骤 1)。

4)取出叶子节点值并返回。

如算法 3 所示,一次深度搜索就可以完成查找下一跳端口值,不需要任何回溯。

3　测试与结果分析

本文提出的路由查找算法在 VS2012 环境下编写,并在内存为 4 GB 的 Windows7-64 bit 操作系统下运行。实例验证时使用了 IPMA 提供的 MAE-EAST 和 AADS(www. merit. edu/ipma)2 个开源路由表作为输入数据分别构建路由表。实验结果与基于 BDD 结构构建的路由表所需节点数进行对比,如表 4、表 5 所示。

表 4　AADS 路由表节点数

路由数	BDD	ADD
1 000	1 784	1 270
5 000	10 038	7 707
10 000	16 965	13 378
15 000	22 251	17 840
20 328	29 212	23 511

表 5　MAE-EAST 路由表节点数

路由数	BDD	ADD
1 000	2 770	1 589
5 000	10 699	6 211
10 000	21 601	12 771
15 000	32 317	26 538
38 470	68 374	45 971

表 4、表 5 分别统计了 2 种开源路由表在使用本文提出的基于 ADD 的路由查找算法和基于 BDD 的路由查找算法构建路由表所需的节点总数。从表中数据可知,基于 ADD 的算法生成的节点数较基于 BDD 的算法生成的节点数明显减少,而且随着路由表项的增多,节点数量的差距逐渐变大。所以,随着路由表项的增多,本文提出的基于 ADD 的路由查找算法的节点递增速度更缓慢,在大型路由表中 ADD 更加适用。

表 6 给出了本文基于 ADD 的路由查找算法与基于 BDD 的路由查找算法在 MAE-EAST 路由表中查找下一跳端口的节点比较次数,前者查找过程中的比较次数较后者平均减少了 76.9% 。因此,本文基于 ADD 的路由查找算法的性能显著优于基于 BDD 的路由查找算法。

表 6　MAE-EAST 路由表中节点比较次数

路由数	BDD	ADD	减少比例/%
5 000	390 861	87 724	77.56
10 000	903 635	204 923	77.32
20 000	1 858 808	433 608	76.67
38 470	2 773 962	662 879	76.10

为对算法有更加直观的认识,本文算法与常用的快速路由查找算法进行比较,使用 MAE-EAST 路由表作为比较数据,比较结果如表 7 所示。可以看出,本文提出的基于符号 ADD 的算法的存储容量为 191 KB,相比较其他路由查找算法的存储性能有较大提升。因此,本文提出的算法能够更加有效地节省空间,尤其适用于大型的路由表以及对资源利用

率要求较高的场合。

表 7 算法存储性能比较 KB

算法	算法占用的存储空间
基于二进制 Trie 树的路由查找算法[7]	7 720
基于路径压缩 Trie 树的路由查找算法[9]	3 500
基于多分支 Trie 树(深度为 5)的路由查找算法[8]	660
基于前缀长度的二分查找法[5]	1 600
文献[16]算法	220
本文算法	191

4 结束语

本文在使用符号 BDD 技术改进传统路由表结构的基础上,根据路由表特征结合 ADD 结构紧凑和能够表示伪布尔函数的优势,提出了基于符号 ADD 结构的路由表模型。结合符号技术给出了相应的路由查找算法,该算法不需要前缀扩展和排序等预处理操作,并且查找时没有回溯。实验结果表明,相比基于 BDD 的路由查找算法,本文提出的基于 ADD 的路由查找算法能大幅减少路由表节点数和路由查找时对内存的访问次数,同时使路由查找时的比较次数平均减少 76.9% ,并且能节省存储空间。下一步将采用多分支的符号技术,缩减路由表深度,在保证路由表存储空间较小的前提下进一步提高算法效率。

参考文献

[1] Sun Y, Egi N, Shi G, et al. Content-based Route Lookup Using CAMs[C]//Proceedings of 2012 IEEE Global Communications Conference. Washington D. C., USA: IEEE Press, 2012: 2677-2682.

[2] Zheng Kai, Hu Chengchen, Lu Hongbin. A TCAM-based Distributed Parallel IP Lookup Scheme and Performance Analysis[J]. IEEE/ACM Transactions on Networking, 2006, 14(4): 863-875.

[3] Kuo Fangchen, Chang Yeim-kuan, Su Cheng-chien. A Memory-efficient TCAM Coprocessor for IPv4/IPv6 Routing Table Update[J]. IEEE Transactions on Computers, 2014, 63(9): 2110-2121.

[4] 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.

[5] Gupta P, Prabhakar B, Boyd S. Near-optimal Routing Lookups with Bounded Worst Case Performance[C]// Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies. Washington D. C., USA: IEEE Press, 2000: 1184-1192.

[6] 范富明,李念军,雷升平,等. 基于哈希表与多比特树的路由查找算法[J]. 计算机工程,2015,41(9):63-67.

[7] 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.

[8] Srinivasan V, Varghese G. Fast Address Lookups Using Controlled Prefix Expansion[J]. ACM Transactions on Computer Systems, 1999, 17(1): 1-40.

[9] Yilmaz P A, Belenkiv A, Uzun N, et al. A Triebased Algorithm for IP Lookup[C]//Proceedings of Global Telecommunications Conference. Washington D. C., USA: IEEE Press, 2000: 593-598.

[10] Kumar S, Becchi M, Crowley P, et al. CAMP: Fast and Efficient IP Lookup Architecture[C]//Proceedings of 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems. New York, USA: ACM Press, 2006: 51-60.

[11] 谭明锋,高 蕾,龚正虎. IP 路由查找算法研究概述[J]. 计算机工程与科学,2006,28(6):77-80.

[12] Sangireddy R, Somani A K. High-speed IP Routing with Binary Decision Diagrams Based Hardware Address Lookup Engine[J]. IEEE Journal on Selected Areas in Communications, 2003, 21(4): 513-521.

[13] 朱云洁,朱凌众. LPM 路由查找算法及其应用[J]. 硅谷,2010(10):127.

[14] 古天龙,徐周波. 有序二叉树决策图及应用[M]. 北京:科学出版社,2009.

[15] 徐 恪,徐明伟,吴建平,等. 路由查找算法研究综述[J]. 软件学报,2002,13(1):42-50.

[16] 张 琦,金胤丞,李 苗,等. Trie 树路由查找算法在网络处理器中的实现[J]. 计算机工程,2014,40(1):98-102.

编辑 陆燕菲

(上接第 98 页)

[8] Anastasi G, Conti M, Francesco M D, et al. Reliability and Energy Efficiency in Multi-hop IEEE 802. 15. 4/ZigBee Wireless Sensor Networks[C]//Proceedings of IEEE Symposium on Computers & Communications. Washington D. C., USA: IEEE Press, 2015: 336-341.

[9] 杨 帆. 基于智能感知的小区集中供暖分户计量系统的设计与实现[D]. 镇江:江苏大学,2013.

[10] 纪均衡. 浅谈 GSM 与 ZIGBEE 模块在智能家居方面的应用[J]. 中国科技博览,2014(3):290.

[11] 张同翰,王正彦,袁 双,等. 基于 CC2530 的 ZigBee 的智能家居设计[J]. 工业控制计算机,2015,28(4):75-76.

[12] 张 睿,王建中. 基于 CC2530 的无线温度传感网络的设计[J]. 杭州电子科技大学学报,2014,34(3):87-90.

[13] Holt A, Huang C Y. OpenWRT[M]. Berlin, Germany: Springer, 2014.

[14] 陶文寅. 基于 OpenWrt 开源系统的无线视频监控智能车设计[J]. 单片机与嵌入式系统应用,2015,15(10):68-71.

[15] 曹为华,凌 强,张 雷,等. 基于 OpenWrt 系统路由器的模式切换与网页设计[J]. 微型机与应用,2015,34(23):91-94.

编辑 陆燕菲