一种采用启发式分割点计算的包分类算法

陈　兵　　潘宇科　　丁秋林
(南京航空航天大学信息科学与技术学院　南京　210016)

摘　要：针对区域分割包分类算法存在的规则分布差异较大的缺陷，该文提出一种基于启发式分割点计算的区域分割包分类算法。首先依据规则集的分布规律进行分割点计算，然后再进行结构化建树。规则检索时间主要包括分割点匹配时间和分割点内规则的线性查找时间。该算法能够尽量将规则平分到各分割点，减少了规则分布的差异。仿真实验结果表明该算法降低了规则数增加对算法性能的影响，支持规则集的实时更新。

关键词：电信网络；包分类；区域分割；分类器
中图分类号：TP393　　**文献标识码**：A　　**文章编号**：1009-5896(2009)07-1594-06

A Heuristic Lookup Partition Algorithm for Packet Classification

Chen Bing　　Pang Yu-ke　　Ding Qiu-lin
(*Institute of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing* 210016, *China*)

Abstract: Regional partition packet classification algorithm is one of the most effective algorithms. To solve the unsymmetrical distribution problem of regional partition for packet classification, a heuristic lookup partition algorithm is proposed after analysing the character of rules' sets. The main research work on the algorithm in this paper includes two parts: the method of lookup partition to ensure the symmetrical distribution of rules, and the construction of decision tree based on the partition found. The time cost of this algorithm is composed by the time for lookup partition and the time for line-search in the partition found. The results of simulation show that the algorithm is not sensitive to the increasement of rules, meanwhile it supports the incremental updating on-line.

Key words: Telecommunication network; Packet classfication; Regional partition; Classfier

1　引言

包分类在网络技术领域中具有广泛的应用，是许多网络关键技术的基础，它关系到网络的控制、性能、安全、管理等多方面内容，包分类速度的快慢、功能的强弱等都直接影响到这些网络技术的性能[1,2]。

如果把分类规则的一个域看作一维空间坐标，则一条分类规则对应 d 维空间中的一个超长方形，一个规则集便是这个多维空间里有一定优先级的超长方形的集合，而输入的包头对应该多维空间里的一个点。这样包分类问题就成了多维空间点的定位问题。

区域分割法是一种基本的空间点定位方法，例如二维包分类可以看成长方形的区域分割。一个规则集内的不同规则一定有域值不同的域，所以通过对域分段就可以把规则分开，直到该段的规则数不多于 1。区域分割包分类算法正是利用这个特点建立算法的查找树结构，把原规则集分为各个叶子节点内的小规则集。对包进行分类时先查找到一个叶子节点，再从叶子节点所包含的规则中找到最佳匹配的规则。基本的区域分割算法如文献[3, 4]适于低维包分类。Gupta 在文献[5]中设计了 Hicuts 算法。Singh 等进行了改进，提出 Hypercuts 算法[6]，以进一步减少空节点。王永刚，颜天信等在文献[7, 8]中提出一种平分区域分割算法，并与其它算法进行了测试对比。平分法实现较为简单，但是不能保证规则能够尽量均匀分布到子空间，从而造成树的深度可能过大。文献[9] 采用 R-tree and R*-tree 来提高检索速度。这些算法不管如何变化，实际都是在区域分割算法的基础上给出一种具体的实现方法，而不同的实现方法可能带来较大的性能区别。所以优化实现是区域分割包分类算法的核心研究内容。本文结合空间几何算法思想和启发式算法思想，提出一种采用启发式分割点计算的区域分割包分类算法，通过对规则库进行分割点的动态计算，实现高效率的规则近似等分，减少了决策树的深度，提高

2008-05-16 收到，2009-02-17 改回
国家部级基金资助课题

了规则检索速度和吞吐量。

2　相关定义

定义 1　分类器C：所谓分类器是含有N个规则$\mathrm{R}_j(1 \le j \le N)$的集合，$C=\{\mathrm{R}_1,\mathrm{R}_2,\cdots,\mathrm{R}_N\}$，规则由3部分构成[10,11]：

(1)关于每个分组头K个字段的正则表达式$\mathrm{R}_j[i]$，$\mathrm{R}_j=(F_1^j,F_2^j,\cdots,F_d^j)$，$F_d^j$表示规则$\mathrm{R}_j$中的第$j$个字段。

(2)规则优先级$\mathrm{prior}(\mathrm{R}_j)$：表示$\mathrm{R}_j$在分类器$C$中的优先级，用来当一个分组$P$同时匹配多个规则时来决定哪个规则优先。

(3)行为$\mathrm{act}(\mathrm{R}_j)$：表示当匹配到该规则后应对分组所采取的操作，一般表示为permit或deny。只能选择一个动作进行处理。

对于接收到的分组P，$P=(P_1,P_2,\cdots,P_k)$可以看作是k维空间中的一个点。K维包分类问题就是要在规则中找到与点$(P_1,P_2,\cdots,P_k)$相匹配，并且优先级最高的规则R_m，即：$\mathrm{prior}(\mathrm{R}_m)>\mathrm{prior}(\mathrm{R}_j)$，$\forall j,m$, $1\le m\le N, 1\le j\le N$, and $\forall(1\le i\le k)$，P_i和$P_i[i]$相匹配，则称R_m是最佳匹配规则。表1给出一个分类器。

定义 2　分割域：每个划分后的子空间称为分割域，记为SubDomain。规则分布在分割域中，分割域可以继续裂变为更小的分割域，直到叶子节点内的规则数满足条件为止。

定义 3　分割点与分割函数：是指进行分割域划分的前缀表示。分割函数通过启发式搜索函数对分割域进行分割点查找。分割点记为S。

定义 4　叶子节点最大规则数：表示叶子节点内包含的最大规则数，记为LeafMaxRules。分割域作为是否继续裂变的阈值，一旦叶内规则数超过LeafMaxRules则继续进行裂变，否则停止。

定义 5　以前i个$\mathrm{bit}(b_0b_1b_2b_3\cdots b_i)$为前缀，定义以下变量：$\mathrm{Max}N_i$为不同前缀的数量。$N_i^j(1\le i\le 32,1\le j\le \mathrm{Max}N_i)$表示满足这些前缀的规则数。$\mathrm{Avg}_i(1\le i\le 32)$表示满足这些前缀的规则数的平均数，$\mathrm{Avg}_i=N_i^j/\mathrm{Max}N_i$。

3　启发式分割点计算

分割点用来进行区域划分，因此分割点的确定对于规则的均匀分布至关重要。启发式分割点计算是一种逐步求精的方法，根据规则库的特点，进行规则的前缀分解，对这些前缀比特位进行分类求和，统计其分布规律，从而找到合适的分割点。考虑到分割域不一定能够完全被分割点等分，如果两个子分割域内的规则数如果差异不大，则这种分割点是可以接受的，而不必继续扩大前缀的搜索位数。设

表1 分类器样例

	源 IP	目的 IP	源端口	目的端口	协议类型	优先级	行为
r0	131.215.175.144/31	170.144.33.176/32	0 : 65535	8080 : 8080	0x11/0xFF	18	deny
r1	131.49.181.0/26	209.180.201.160/31	0 : 65535	25:25	0x06/0xFF	17	permit
r2	131.49.181.64/28	170.144.188.144/28	0 : 65535	53:53	0x11/0xFF	16	deny
r3	131.49.0.0/17	91.158.163.21/32	0 : 65535	5729 : 5729	0x06/0xFF	15	deny
r4	131.49.181.90/32	170.144.188.0/23	0 : 65535	1025: 65535	0x06/0xFF	14	permit
r5	128.49.181.90/32	75.27.10.236/32	0 : 65535	0 : 65535	0x00/0x00	13	deny
r6	115.149.169.218/31	128.82.192.0/21	0 : 65535	1025: 65535	0x06/0xFF	12	deny
r7	138.9.29.165/32	123.42.0.0/15	0 : 65535	1025: 65535	0x06/0xFF	11	deny
r8	128.158.64.0/18	114.222.141.144/28	0 : 65535	0 : 65535	0x06/0xFF	10	deny
r 9	128.49.181.64/28	91.139.237.192/26	0 : 65535	0 : 65535	0x00/0x00	9	permit
r10	130.49.128.0/19	209.176.210.84/32	0 : 65535	0 : 65535	0x00/0x00	8	permit
r11	130.49.181.128/25	141.72.0.0/16	0 : 65535	0 : 65535	0x06/0xFF	7	permit
r12	20.165.248.128/28	27.90.128.0/17	0 : 65535	0 : 65535	0x00/0x00	6	deny
r13	121.128.0.0/13	85.176.100.178/31	0 : 65535	0 : 65535	0x00/0x00	5	deny
r 14	138.32.0.0/13	171.235.207.134/31	0 : 65535	0 : 65535	0x00/0x00	4	permit
r15	115.144.0.0/12	85.176.100.208/32	0 : 65535	0 : 65535	0x00/0x00	3	permit
r16	130.48.0.0/12	170.144.188.131/32	0 : 65535	0 : 65535	0x00/0x00	2	deny
r17	89.189.205.32/27	85.180.0.0/14	0 : 65535	0 : 65535	0x00/0x00	1	deny
r18	102.34.144.199/32	0.0.0.0/0	0 : 65535	0 : 65535	0x58/0xFF	0	permit

立一个调节因子λ，第 1 个接近Avg_i的前缀如果与Avg_i比值在调节因子范围内，则可认为找到了该分割域的分割点。在后面的仿真实验中，给出不同λ值对规则检索速度的影响。算法 1 给出了启发式分割点计算的过程。

算法 1 启发式分割点计算算法

i = length;//length 为初始的前缀位数

//初始化 N 为第 1 个前缀对应的规则数

cont: $N=N_i^1$;

//step 表示步长，如可设置为 2

step = 2;

//寻找第 1 个接近Avg_i的前缀

$j=1$;

while $(j<\mathrm{Max}N_i)$ {

$N=N+N_i^j$;

if $\left(\dfrac{N}{\mathrm{Avg}_i}\geq\lambda\right)$ exit

j++;

}

//若未找到，则增加前缀的 bit 数

if $(j==\mathrm{Max}N_i)$ {

i += step;

if($i\geq 32$) exit

goto cont;

}

4 基于启发式分割点计算的包分类算法

4.1 结构化建树

建树算法用于根据给定的规则集构造一棵决策树，是在预处理阶段完成，即在规则集建立之后，包过滤工作之前构造完毕。首先给出主要数据结构：

(1)定义 IP 地址及其前缀。

```
struct filt{
    unsigned int sa; // IP 地址
    int sa_len; //网络前缀
};
```

(2)定义规则索引链表。

```
struct filter{
    strunct filt  filts; //规则索引
    struct filter *next; //规则链表
};
```

(3)定义结构树中的节点链表。

```
struct node{
int flag; // 0：中间节点，1：叶子节点
unsigned int bit; //表示 IP 地址的前 bit 位为搜索匹配长度
unsigned int div; //分割点等于 div<<(32-bit)
struct filter *f1; //存放小于分割点的规则的索引
struct filter *f2; //存放大于等于分割点的规则的索引
struct node *left; //当前节点的左孩子节点
struct node *right; //当前节点的右孩子节点
};
```

算法 2 给出了决策树的建立过程。

算法 2 建立决策树 Recursion

```
Struct node *Recursion(struct filter*filters, double rate, int start, int step, int LeafMaxRules)
{
struct node *nodes = new node;
//寻找分割点point及对应的掩码长度length
point = find_breakpoint(filters,& length);
//获取分割点的IP地址
point_ip = point<<(32-length);
nodes->div = point;
nodes->bit = length;
//filters初始为所有的规则索引
while (filters){
    filt = 对应的规则的IP域;
    if (filt.sa_len>=length){//规则的网络前缀长度大于等于length
        if (filt.sa<point_ip){//规则的IP值小于分割点的 IP
        将规则的索引添加到小于point_ip的索引链表filters_lchild;
        count_l++; //左链表长度+1
        }
        else {//规则的IP 值大于等于分割点的IP
        将规则的索引添加到大于等于point_ip的索引链表filters_ rchild;
        count_r ++;//右链表长度+1
        }
    }
    else {//规则的网络前缀长度小于length
        if (max(filt.sa)<point_ip){//匹配的IP的最大值小于分割点IP
        将规则索引添加到小于point_ip的链表filters_lchild;
        count_l++; //左链表长度+1
        }
        else if (min(filt.sa)>=point_ip){//匹配的IP的最小值大于等于分割点IP
```

```
                将规则索引添加到大于等于point_ip的链表filters_rchild;
                count_r++; //右链表长度+1
            }
            else {//分割点在规则匹配的IP值的最大值和最小值之间。
                将规则的索引添加到filters_lchild和filters_rchild;
                count_l++; //左链表长度+1
                count_r++; //右链表长度+1
            }
        }
        filters = filters->next;
        }//end of while
        If (count_l>LeafMaxRules or count_r> LeafMaxRules) {
            nodes->left=Recursion(filters_lchild, rate, start, step, LeafMaxRules);
            nodes->right=Recursion(filters_rchild, rate, start, step, LeafMaxRules);
            nodes->flag=0;
        }
        else {
            nodes->f1 = filters_lnode;
            nodes->f2 = filters_rnode;
            nodes->flag = 1;
        }
    return nodes;
    }
```

根据该算法对表 1 的规则建立的决策树如图 1 所示。

4.2 规则检索

算法的查找过程分为两步：首先搜索决策树找到叶子节点，然后在该叶子节点所包含的子规则集范围内进行线性查找，即逐个比较，从而找到最佳匹配规则。查找树的时间复杂度为 $O(D)$，其中 D 为查询树的平均深度；而线性查找的逐个比较所需时间为 $O(dM)$，M 是线性查找时的规则数(即 LeafMaxRules)，d 表示维数。所以整个规则检索的时间复杂度为 $O(D+dM)$。在利用单域化线性查找后，可适当加大 M 的值，使树的深度(尤其是最大深度)减小，从而改善算法的时间特性。后面的仿真实验给出 M 对规则检索的影响。

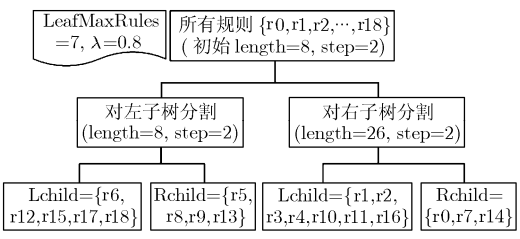

图 1 决策树

4.3 规则更新

算法的更新速度是算法应用场合的重要判据之一。区域分割算法的更新分为两种情况：

(1)新增规则。过程分两步：第 1 步是查找结构树找到相应的分割点，这个查找过程与包分类的查找过程类似；更新过程的第 2 步是更改所查到的叶子节点内的规则，如果该规则加入后造成叶内节点数大于 LeafMaxRules，则调用分割点算法对该节点进行分割，生成新的左右子树。

(2)删除规则。首先找到分割点，在该节点内线性查找完全相同的规则，有完全相同的规则就删除之，再检查该节点的规则数是否为 0，为 0 则把该节点也删除，若没找到完全相同的规则，则删除过程结束。

无论是新增规则还是删除规则，不需要重新构建整个决策树，因此，规则更新速度快。

5 仿真结果

华盛顿大学应用研究实验室开发的 Class Bench[10]是目前较为权威的规则库的产生和测试平台。该平台对网络服务提供商和其他研究机构提供的 12 个真正的规则库集合进行了统计分析得到 12 个参数文件，据此可以产生不同的规则库用于结构化建树，同时也可以产生相应的测试规则集。本文从 500 个规则集开始产生，一直到 10000 条规则逐个进行测试。在测试环境为 Intel P4，3.2 GHz (CPU)，1 GB DDR2 内存的平台下测试了采用启发式分割点计算的包分类算法。

(1)本算法与线性查找算法、区域平分和 Hyper Cuts 的对比如图 2 所示。此时启发式分割点计算算法的 λ=0.8，LeafMaxRules=50。从图 2 可以看出，启发式分割点算法对规则数的增加不敏感，且搜索时间和 HyperCuts 相当，但优于区域平分和线性搜索算法。

(2) λ 对本算法搜索时间的影响如图 3 所示。λ 取值分别为 0.9，0.8，0.6 和 0.5，LeafMaxRules=50。从图中可以看出，λ 取值为 0.9 或 0.8 时，搜索时间较小。

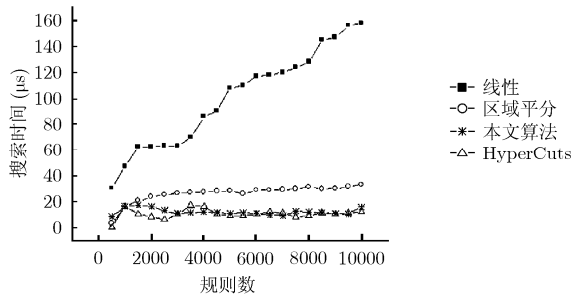

图 2 4 种算法的平均搜索时间

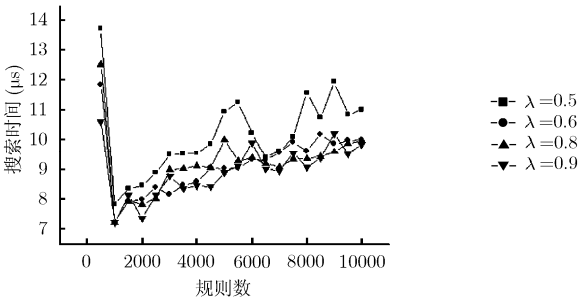

图 3 不同 λ 下的平均搜索时间

(3)LeafMaxRules 对本算法搜索时间的影响如图 4 所示。LeafMaxRules 对应图中的 M，取值分别为 20，50，80，100，125，150，λ 为 0.8。结果显示 LeafMaxRules=50 时规则搜索时间较小。

(4)算法的空间占用对比如图 5 所示。由图中可见，线性查找算法的平均空间占用最少，启发式分割点搜索算法和区域平分算法的空间占用基本相同，其空间复杂度为 $O(dN)$ (d 为维数，N 为规则数)，HyperCuts 占用空间较大。

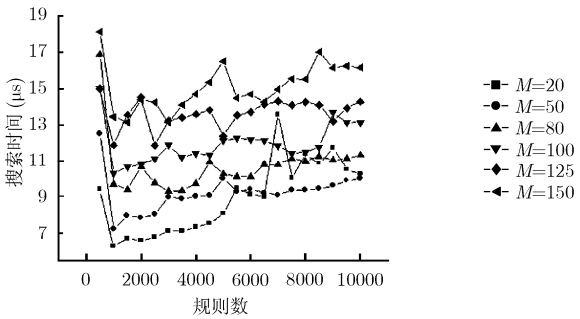

图 4 不同 LeafMaxRules 下的平均搜索时间

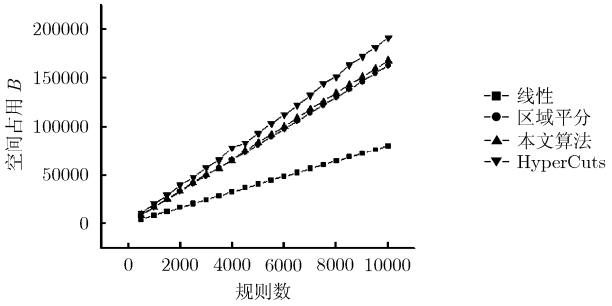

图 5 4 种算法的平均占用空间

6 结束语

基于区域分割法实现的包分类算法是目前较为有效的分类算法之一。在区域分割思想的基础上算法有很多种实现可能，且不同实现可能有较大的性能差别，所以优化实现是区域分割包分类算法的核心研究内容。本文所做的工作是在区域分割的基本原理上，通过启发式动态分割点计算，实现了高效率的规则近似等分，通过与其它区域分割包分类算法比较，结果显示：

(1)与 HyperCuts 相比，二者搜索时间接近，但是本算法空间需求较 HyperCuts 少；

(2)与颜天信等人改进的区域分割法(平分区域法)相比，本算法生成的决策树深度小，搜索时间少，空间需求相当。

(3)分割点计算的调节因子 λ 选择 0.8 或者 0.9 时，本算法的搜索时间较少。

(4)叶内节点最大规则数选择 50 时，本算法的搜索时间较少，搜索时间曲线较为平滑。

参 考 文 献

[1] Taylor D E. Survey and taxonomy of packet classification techniques[J]. *ACM Computing Surveys*, 2005, 37(3): 238–275.

[2] 陆晟, 龚俭. 一种新的高维报文分类算法—无相交树算法[J]. 计算机学报, 2003, 26(11): 1502–1509.
Lu Sheng and Gong Jian. A multi-dimension packet classification algorithm: Nonintersection trie[J]. *Chinese Journal of Computers*, 2003, 26(11): 1502–1509.

[3] Buddhikor M M, Suri S, and Waldvogel M. Space decomposition techniques for fast layer-4 switching[C]. Proc.of Conf. on Protocols for High Speed Networks. Salem, Massachusetts, USA,August 1999: 25–41.

[4] Feldtnan A and Muthukrishnan S. Tradeoffs for packet classification[C]. Proc. of Infocom. Israel, March 2000: 193–202 .

[5] Gupta P and Mckeown N. Packet classification using hierarchical intelligent cuttings[J]. *IEEE Micro*, 2000, 20(1): 34–41.

[6] Singh S, Baboescu F, Varghese G, and Wang J. Packet classification using multidimensional cutting. Proceedings of ACM SIGCOMM[C]. Karlsruhe, Germany, 2003: 238–275.

[7] 颜天信, 王永刚等. 并行区域分割包分类算法[J]. 小型微型计算机系统, 2005, 26(11): 1898–1902.
Yan Tian-xin and Wang Yong-gang, *et al*.. Packet classifi-cation using parallel regional partition[J]. *Mini-Micro*

Systems, 2005, 26(11): 1898–1902.

[8] 王永刚, 石江涛等. 网络包分类算法仿真测试与比较研究[J]. 中国科学技术大学学报, 2004, 34(4): 400–409.
Wang Yong-gang and Shi Jiang-tao, *et al*.. Simulated testing and comparison of algorithms for packet classification [J]. *Journal of University of Science and Technology of China*, 2004, 34(4): 400–409.

[9] Masatoshi Kakiuchi, Naoto Morishima, and Hideki Sunahara. Design and implementation of a parameter filter based on virtual bounding rectangles[J]. *Systems and Computers in Japan*, 2007, 38(13): 92–103.

[10] 田立勤, 林闯等. 基于 IXP1200 的快速报文分类算法的设计与实现[J]. 计算机研究与发展, 2003, 40(11): 1616–1625.
Tian Li-qin and Lin Chuang, *et al*.. Design and implementation of fast packet classification based on IXP1200[J]. *Journal of Computer Research and Development*, 2003, 40(11): 1616–1625.

[11] Kounavis M E, Kumar A, Yavatkar R, and Vin H. Two stage packet classification using most specific filter matching and transport level sharing[J]. *Computer Networks*: *The International Journal of Computer and Telecommunications Networking*, 2007, 51(18): 4951–4978.

陈　兵：　男，1970 年生，副教授，从事计算机网络和信息安全研究.

潘宇科：　女，1984 年生，硕士生，从事计算机网络研究.

丁秋林：　男，1934 年生，博士生导师，主要研究方向为系统集成.