第 40卷 第 3期
2008年 3月

哈 尔 滨 工 业 大 学 学 报
JOURNAL OF HARBIN INSTITUTE OF TECHNOLOGY

Vol. 40 No. 3
Mar. 2008

适用于 GIDS报文分类的 P-HiCuts算法

龚 俭, 魏 薇, 周 鹏

(东南大学 计算机科学与工程学院, 南京 210096, E-mail: wwei@njnet.edu.cn)

摘 要: 针对 HiCuts算法在 NIDS应用上存在着空间异常膨胀和决策树不平衡性的问题,提出了一种 P-HiCuts算法. P-HiCuts(Pruned HiCuts)对原报文空间分组算法进行改进,采用覆盖规则上提和非均匀切分的技术解决原有问题,从理论上减小了决策树深度.实验结果显示,改进后决策树深度空间占用缩小到原来的 10%,分类速度也提升了 13.71%.

关键词: 报文分类;分类算法; NIDS; HiCuts; P-HiCuts

中图分类号: TP393 **文献标识码**: A **文章编号**: 0367-6234(2008)03-0448-05

P-HiCuts algorithm for GIDS parket classification

GONG Jian, WEI Wei, ZHOU Peng

(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China, E-mail: wwei@njnet.edu.cn)

Abstract: To overcome the unusual expansion of the space consumption and unbalance of the decision tree in the application of HiCuts algorithm in NIDS, a P-HiCuts algorithm is presented. P-HiCuts improves the original space-cutting algorithm through node-overlapped rules upward moving and asymmetrical space cutting. The experimental result shows that the new decision tree is shorter, its memory consumption is ten percents of the original one, and the system performance is promoted 13.71%.

Key words: packet classification; classification algorithm; NIDS; HiCuts; P-HiCuts

GIDS(Gigabit IDS)是指能处理千兆及更高流量的 NIDS报文分类是 GIDS必需的处理工作,设计 GIDS分类算法时,不仅要提高速度,还需要考虑到内存空间的限制,力求两者的平衡[1,2]. HiCuts是 1999年由 Gupta和 McKeown提出的一种灵活的报文分类算法[3],但算法在 NIDS应用上存在着空间异常膨胀和和决策树不平衡性的问题.

本文通过修改决策树的报文空间分组算法解决上述问题,从而获得了更好的系统性能.介绍了 HiCuts算法的主要思想,分析了算法的优势以及目前存在的问题,提出了改进思路和算法 P-HiCuts,通过理论和实验验证了算法的优越性.

收稿日期: 2005-09-28.
基金项目: 国家重点基础研究发展规划资助项目(2003CB314803);
国家自然科学基金资助项目(90104031);
江苏省网络与信息安全重点实验室资助项目(BM2003201).
作者简介: 龚 俭(1957—),男,博士,教授,博士生导师;
魏 薇(1982—),女,博士研究生.

1 HiCuts算法

1.1 HiCuts算法思想

HiCuts的基本思想是以每个报头域为一个层次,将报文空间逐层等距分组,生成报文分类决策树.当报文到达时,遍历决策树找到一个与之匹配的存储少量规则的叶结点,再使用线性查找算法,找到最佳匹配.树中每个节点 v都关联两个变量:报文子空间和规则子集.报文子空间记为 $Box(v)$,含义是节点 v代表的报文集合.规则子集记为 $R(v)$, $R(v) = \{$规则 $r \mid \exists$ 报文 $p \in Box(v)$, p满足 $r\}$, 表示 $Box(v)$可以满足的规则集合. $R(v)$的基数用 $|R(v)|$来表示.分组的终止条件是 $|R(v)|$小于预设筏值 $binch$或无报头域可以用于继续分组.

已知规则集与 K个报头域相关,报头域编号为 d_i $(1 \leq i \leq K)$.节点 v的 $Box(v) = \{L_1, \cdots, L_K\}$, L_i $(1 \leq i \leq K)$表示 $Box(v)$在 d_i上报文的取

值范围. 对 $Box(v)$ 在报头域 d上进行分组,将 v划分成 nc个子结点,则定义节点占用预算 $sm(v) = nc + \sum_{i=1}^{nc} |R(C_i)|$ (其中 C_i为 v的子节点). 分组算法 Cut1使用预设的空间衡量函数 $spmf()$,利用二分查找法启发式寻找合适的 nc, 使得节点占用预算 $sm(v)$尽量接近预设的空间 $spmf(|R(v)|)$.

1.2　HiCuts算法生成实例

例 1(HiCuts报文分类决策树):报头域 d_1、d_2的取值范围都是 $[0, 100]$,规则集 R与 d_1、d_2相关,具体如表 1.

表 1　规则集 R

规则	d_1取值范围	d_2取值范围
r1	[0, 15)	[0, 40)
r2	[15, 25)	[20, 90)
r3	[0, 25)	[50, 80)
r4	[25, 100)	-
r5	[80, 100)	[0, 50)
r6	[80, 100)	[60, 100]

图 1为规则空间分布图,报头域 d_1、d_2分别对应报文空间的 X、Y轴. 各矩形框与 R中各规则的空间一一对应. 图 2是设 binch为 2时 HiCuts算法生成的报文分类决策树(假设通过设置 spmf使在 d_1上的分组数为 4, d_2上的分组数为 2).

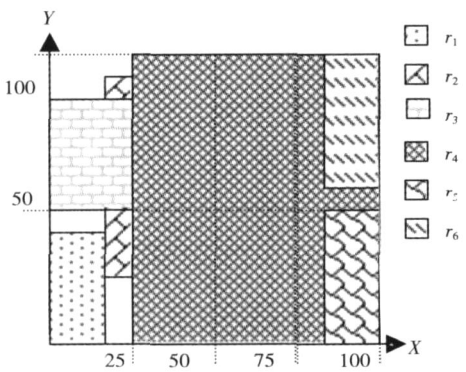

图 1　规则空间分布图

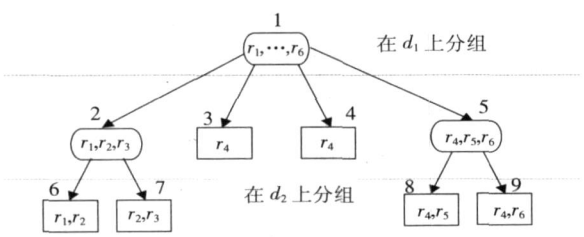

图 2　HiCuts报文分类决策树

2　HiCuts算法分析

2.1　HiCuts算法的优势和存在的问题

目前,其他报文分类算法主要有: Ternary Cam、ABV[4]、Cross-producting[5]、Grid of Tries[5]、RFC[1]、Hierarchical Tries、Set-Prunning Tries、Tuple Space Searching[6]. 而在 NIDS应用中,HiCuts算法与上述算法比较,在规则表达能力、分类速度和空间占用上具有明显的优势[7]. HiCuts算法已被应用于防火墙、路由器等领域并获得了良好的效果. 但是,实践表明,两种情况下,HiCuts算法存在问题:

1) 决策树空间异常膨胀. 该问题常见于报头域数量较少规则占规则总数的比重较大的情况.

2) 决策树平衡性较弱. 该问题常见于各规则在某些报头域的取值范围过于集中的情况.

2.2　决策树空间异常膨胀问题的分析

该问题的主因是报头域越少的规则,其规则空间越大,采用 Cut1逐层分组后,该类规则存在于较多节点的规则子集中. 规则子集基数的增大会引起节点的继续分组,最终导致空间异常膨胀. 为说明这一点,引入覆盖规则的定义和相关性质.

定义 1　(覆盖规则)节点 v上,∃规则 r, ∀报文 $p \in Box(v)$, p都满足 r,则称 r为 v的覆盖规则. 如例 1中,只与报头域 d_1相关的规则 r_4是节点 v_5的覆盖规则.

性质 1　(覆盖规则的继承性)∀节点 w, w是节点 v的子孙节点, ∀规则 r, r是 v的覆盖规则,则 r也是 w的覆盖规则. 例 1中, v_8、v_9是 v_5的子节点,则 r_4也是 v_8、v_9的覆盖规则.

证明　∵w是 v的子孙节点,

$\therefore Box(w) \subset Box(v)$.　(1)

∵r是 v的覆盖规则,

$\therefore \forall p \in Box(V)$, p满足 r　(2)

根据式(1)、(2), $\forall p \in Box(v)$, p满足 r

根据定义 1, r是 w的覆盖规则.

性质 2　(覆盖规则的膨胀性)当节点的覆盖规则总数超过 $binch$,将造成该节点下决策子树的节点数按照子树深度的指数增长.

证明　假设节点 v的覆盖规则数为 t, $t > binch$. 根据性质 1,节点子孙的覆盖规则数必定不小于 t,因此分组需持续到无报头域可用为止. 设以 v为根节点的决策子树深度为 H, l_i $(1 \le i \le H)$表示每层的分组数(节点的分组数主要与预设空间上限公式 $spmf(|R(v)|)$有关). 则该决策子树的节点数为 $\sum_{i=1}^{H-1} \prod_{j=1}^{i} l_j$.

性质 3　(覆盖规则的报头域相关性)规则相关报头域越少,它成为节点覆盖规则的概率就越高.

证明　设分类报头域数量为 D,其中与规则 r相关的有 d个,若算法以等概率的方式挑选报头域进行分组(HiCuts有专门的算法挑选分组用的

报头域,实际报头域的挑选与规则集本身结构相关.为方便讨论,本文假设规则集以等概率的方式挑选报头域进行分组),则 r在第 K层是覆盖规则的概率为 C_{D-d}^{K-d}/C_D^K.而由性质 1可知,若 r在第 K层是覆盖规则,那么只要 r在第 i层 $(1 \leq i \leq K)$成为覆盖规则即可.所以,设 r在第 K层成为覆盖规则的概率为 $P_d(K)$,则

$$C_{D-d}^{K-d}/C_D^K = \sum_{i=1}^{K} P_d(i),$$

$$\therefore P_d(K) = \sum_{i=1}^{K} P_d(i) - \sum_{i=1}^{K-1} P_d(i) = C_{D-d}^{K-d}/C_D^K - C_{D-d}^{K-d-1}/C_D^{K-1} \quad (d \leq K \leq D).$$

可以证明,当满足 $d \geq 2$且 $K \leq D/2$时,$P_d(K)$在 d上递减.

因此,若规则集中有 M条规则的相关报头域总数为 d,则在第 K层的覆盖规则数 NUM_K的期望值 $E(NUM_K) = M * P_d(K)$.根据性质 3,期望值将随着 d的减小而增大,在 M上线性增长.当 $NUM_K \geq binch$,根据性质 2,K层的子树节点数将以 $(d - K)$的指数增长.

2.3 决策树平衡性较弱问题的分析

该问题产生的主因是规则集在报头域的取值范围集中在某些区域上,因此 Cut1等距分组后,各子节点规则子集基数不等,进而影响决策树的平衡性.已知 HiCuts算法的分类速度与匹中叶节点的深度成反比,因此,决策树的非平衡性将会导致决策树的分类速度不稳定,在某些时刻出现分类速度较慢的情况.

3 P-HiCuts算法

3.1 NDS规则特征

对 Snort、天阗 DS、Dragon DS、NFR ND-100的规则库的研究发现,DS规则集存在着两种情况:

1) DS规则集涉及的报头域超过 10个,但每规则使用的报头域相对较少.以 Snort2.0规则集为例,报头域数量为 2～3个的规则占总数的 88.95%.

2) DS规则集在某些报头域上的取值范围分布明显不均,以 TCP规则为例,其中超过 60%的规则在宿端口上的取值范围低于 1 024,这主要是因为攻击基本上是针对知名服务端口[8].

试验表明,比较其他应用中同等规模的规则集[9],由 DS规则集产生的 HiCuts决策树(见第 5节验证实验)空间占用明显偏大,平衡性较不理想.

3.2 改进思路

改进 1(覆盖规则上提).对空间异常膨胀的问题,最简便的方法是通过增大 binch来增加节点的规则容纳能力,减少分裂的次数.但是 binch过大将导致分类速度的下降,极限情况下,算法将退化到 Linear Search.因而,比较合适的方法是:在预编译生成决策树时,提出节点的覆盖规则集合,不再参与分组(例如将图 2中 v_5的覆盖规则 r_4提出),以此来抑制由该类规则引起的空间指数膨胀.在执行报文分类工作时,直接匹中所在节点相关的覆盖规则集合,然后再执行后续操作.

改进 2(非均匀分组).对决策树平衡性较弱的问题,可以采用非均匀分组的方法,以各规则在报头域上取值范围的边界值作为切分点(图 2中 d_1上的切分点为 15、25、80).区域规则数越多,规则范围边界值出现的可能也会越大,这样该区域的切分次数就会增大,从而使分组后每个区间的规则集基数趋于平均.

3.3 P-HiCuts算法

根据改进思路,对 HiCuts的 Cut1算法加以改进,得到的 Cut2算法.

输入:

1) 需要分组的节点 v;

2) 报头域 d.

输出:

1) 分组的节点数 nc';

2) 子结点 $\{C_1, \cdots, C_{nc'}\}$.

方法:

1) 计算节点 v的覆盖规则集合 $Ronr(v)$,保存到节点 v中.计算 $R(v)$与 $Ronr(v)$的差集 $R'(v)$;

2) 清空集合 P,取 $R'(v)$各规则在报头域 d上的取值范围的边界值插入到 P中.完成后,$P = \{e[1], \cdots, e[i], \cdots, e[nc'+1]\}$,$e[i]$表示 P中的点,$nc'+1$为 P中点的数目;

3) 以 P作为 d的切分点集,切分后形成连续区间集 $L = \{S_1, \cdots, S_{nc'}\}$,其中 $S_j = [e[j], e[j+1]]$,$(1 \leq j \leq nc')$;

4) 创建 nc'个 v的子节点 C_t $(1 \leq t \leq nc')$,$Box(C_t) = \{L_1, \cdots, L_i', \cdots, L_K\}$,$L_i' = S_t$;

5) 返回 nc'、$\{C_1, \cdots, C_{nc'}\}$.

3.4 P-HiCuts生成实例

例 2(P-HiCuts报文分类决策树).规则集和 $binch$与例 1一致,这里分组的终止条件是 $|R'(v)|$小于预设筏值 $binch$或无报头域可以用于继续分组.生成的决策树如图 3,图中每个节点由二元组 $(Ronr(v), R'(v))$组成,$Ronr(v)$为节点 v的覆盖规则集合,$R'(v)$为本节点除去覆盖规则集合后参与进一步分组的规则集.

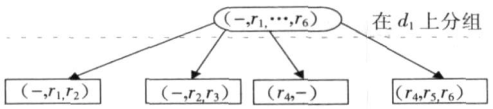

图 3　P-HiCuts报文分类决策树

4　P-HiCuts算法分析

4.1　P-HiCuts决策树深度分析

通过例 2可以看出,覆盖规则上提可以抑制分组,而使用非均匀分组来替换等距分组可以使分组后子节点规则集基数趋于平均.因此 P-HiCuts生成的决策树深度较低.为证明此点,引入区间规则集定义及性质.

定义 2　(区间规则集)规则 r在报头域 d上的取值范围为 L_d,则 $\forall$区间 $l \subseteq L_d$,称 r满足 r规则集 R在 d的区间 l上规则集 $Sub(R, d, l) = \{$规则 $r \mid \exists\, l' \subseteq l,\ l'$满足 $r,\ r \in R\}$.

性质 4　(区间规则集的含义)已知节点 v、w,w是 v在报头域 d上分组后得到的子节点,w对应了 d上的区间 l 由区间规则集定义可知,$Sub(R(v), d, l) = R(w)$.

性质 5　(区间规则集的规则包含性)报头域 d的取值范围为 L_d,规则集 R_1, R_2,且 $R_1 \subseteq R_2$,则 $\forall\, l \subseteq L_d$, $Sub(R_1, d, l) \subseteq Sub(R_2, d, l)$.

证明　∵根据定义 2,

$\forall\, r \in Sub(R_1, d, l), \exists\, l' \subseteq l$,$l$满足 r,　(3)

∵$R_1 \subseteq R_2$　, $r \in R_1$,

∴$r \in R_2$.　(4)

∴由式 (3)、(4) 可知,l 满足 r, $r \in R_2$,

∴$r \in Sub(R_2, d, l)$

∴根据 r的任意性,$Sub(R_1, d, l) \subseteq Sub(R_2, d, l)$.

性质 6　(区间规则集的区间包含性)报头域 d的取值范围为 L_d,规则集 R,则 $\forall\, l_1 \subseteq L_d$,若 $l_2 \subseteq l_1$, $Sub(R, d, l_2) \subseteq Sub(R, d, l_1)$.

证明　∴$\forall\, r \in Sub(R, d, l_2), \exists\, l \subseteq l_2$, l满足 r,

又 ∵$l_2 \subseteq l_1$,

∴$l \subseteq l_1$,

∴$r \in Sub(R, d, l_1)$,

∴根据 r的任意性,$Sub(R, d, l_2) \subseteq Sub(R, d, l_1)$.

性质 7　(区间规则集的最小切割集)在规则 R报头域 d上分组,使用 Cut2算法分组后得到区间的集合 L_1.任取一种分组算法 C,设 L_2为 C得到区间的集合. $\forall$区间 $l \in L_1$, $\exists$ 区间 $l' \in L_2$,使得 $Sub(R, d, l) \subseteq Sub(R, d, l')$.

证明　∵采用规则边界分组

∴$\forall\, l \in L_1$, $\forall\, l_2 \subseteq l$, $Sub(R, d, l) \equiv Sub(R, d, l_2)$,

∵$\exists\, l' \in L_2 \wedge l' \cap l \neq \Phi$,取 $l_3 = l' \cap l$,

∴$Sub(R, d, l) \equiv Sub(R, d, l_3)$,

∵$l_3 \subseteq l'$,

∴根据性质 6, $Sub(R, d, l) \equiv Sub(R, d, l_3) \subseteq Sub(R, d, l')$.

定理 1　(P-HiCuts深度降低定理)对于同样的规则集 R, HiCuts生成的决策树深度不小于 P-HiCuts生成决策树深度.

证明　设 R相关的报头域数量为 D,分组算法挑选报头域的次序相同, P-HiCuts生成的决策树为 T_1, HiCuts为 T_2.设 T_1的根节点为 t_1, T_2的根节点为 t_2.

∵规则集相同, $R(t_1) \equiv R(t_2) \equiv R$,　(5)

设在层 $K(2 \leq K \leq D-1)$上, $\forall\, T_1$上的节点 v, $\exists\, T_2$上的节点 w, $R(v) \subseteq R(w)$.　(6)

若 $|R(w)| \leq binch$, w、v分组终止;

若 $|R(v)| \leq binch$, $|R(w)| > binch$, v分组终止;

若 $|R(v)| > binch$, w, v继续分组,设分组报头域为 d, v分组后的区间集为 L_1, w为 L_2,

∵根据性质 7, $\forall$区间 $l_1 \in L_1$, $\exists\, l_2 \in L_2$, $Sub(R(v), d, l_1) \subseteq Sub(R(v), d, l_2)$,　(7)

∵根据式 (6)、性质 5, $\forall$区间 l, $Sub(R(v), d, l) \subseteq Sub(R(w), d, l)$,　(8)

∴根据式 (7)、(8), $\forall$区间 $l_1 \in L_1$, $\exists\, l_2 \in L_2$, $Sub(R(v), d, l_1) \subseteq Sub(R(v), d, l_2) \subseteq Sub(R(w), d, l_2)$,　(9)

设 l_1对应 T_1上节点 v_1, l_2对应 T_2上节点 w_1,

∴根据 (9)、性质 4, $R(v_1) \subseteq R(w_1)$,

∴根据 l_1的任意性可知,在 $K+1$层上, $\forall\, T_1$的节点 v_1, $\exists\, T_2$上的节点 w_1, $R(v_1) \subseteq R(w_1)$,　(10)

∴根据式 (5)、(10),设在层 $K(1 \leq K \leq D)$上, $\forall\, T_1$上的节点 n, $\exists\, T_2$上的节点 m, $R(n) \subseteq R(m)$,

∴根据分组条件易知, T_2的深度不小于 T_1.

4.2　P-HiCuts决策树复杂度分析

理论上,节点的最大分组数与规则总数 N成正比,决策树深度 ≤规则相关报头域的数目 D.因此 P-HiCuts的最坏时空复杂度与 HiCuts算法相同:最坏时间复杂度为 $O(D)$,最坏空间复杂度为 $O(N^D)$.根据定理 1, P-HiCuts生成的决策树深度较小.根据复杂度,深度的减小将使占用空间呈指数减小,速度也得到线性的提升.

5 实验验证

实验使用四台服务器作为流量发生器,模拟流量使用的报文采用分层抽样[10]技术从实际网络收集,报文长度为 350～420. 检测主机与流量发生器组成星状网络.

入侵检测主机的分类算法交替采用 HiCuts算法和 P-HiCuts算法,规则取自 snort2.2,共 920条. 表 2是使用两种算法后决策树深度和空间占用的比较. 从表 2可知,采用 P-HiCuts算法,决策树平衡性得到改进,叶节点深度落差维持在 2以下. 较 HiCuts算法,决策树深度减少了 2. 理论上,空间占用压缩比应为 $920^2 = 846\ 400$,实际压缩比(binch设置为 2)为 180.2/16.4 = 10.99,远低于理论值,这是因为通常节点的分组数远低于最坏情况下的规则总数.

表 2 决策树内存占用与深度比较

决策树	占用内存/MB		树的深度	
	P-HiCuts	HiCuts	P-HiCuts	Hicuts
2	16.4	180.2	3～5	3～7
3	15.7	130.1	3～5	3～6

实验中,通过调节流量发生器软件参数,保持到达入侵检测主机的总流量由四点均匀产生. 受到硬件环境的限制,总流量只能达到 1.5 G左右. 本文以入侵检测系统在模拟网络流量下的丢包率来量化系统性能,表 3是分别采用 HiCuts和 P-HiCuts后的系统丢包率.

表 3 系统丢包率比较

总流量/GB	丢包率/%	
	HiCuts	P-HiCuts
0.96 ～ 1.01	0	0
1.12 ～ 1.13	3.37	0
1.19 ～ 1.22	9.8	4.5
1.41 ～ 1.45	21.2	10.4
1.51 ～ 1.53	37.3	19.1

NIDS的处理速度 =(1 -丢报率) ×流量,从表 3可知,使用 HiCuts算法的 NIDS速度极限在 78.8% ×1.41 ×353 kpps = 392.21 kpps,使用 P-HiCuts的 NIDS极限值在 89.6% ×1.41 ×353 kpps = 445.97 kpps,速度提升了 (445.97 - 392.21)/392.21 = 13.71%. 根据第 4节分析,算法的速度与决策树深度成反比,因此速度的理论提升值为 (1/5 - 1/7)/1/5 = 28.6%,理论和实际数据的差异主要是由于 NIDS的报文分类工作和报文检测工作是以串联方式进行,随着流量的上升,报文检测成为系统的性能瓶颈,制约了分类速度. 随着流量的上升,NIDS检测进程、报文获取进程及分类进程在 CPU、总线、I/O端口等资源上竞争加剧,反而降低了系统性能,例如,在流量达到 1.51 G时,处理速度只能达到 80.9% ×1.51 ×353 kpps = 431.22 kpps.

6 结 论

1) P-HiCuts可以通过覆盖规则上提来抑制空间异常膨胀问题.

2)非均匀分组使切分后各节点规则集基数趋于平均,从而增强了决策树的平衡性,降低了树的深度,因而获得了超过 10倍的空间压缩比,系统速度也提升了 13.71%,促进了 GIDS的整体性能的提升.

参考文献:

[1] GUPTA P, MCKEOWN N. Packet classification on multiple fields[C]//Proc ACM SIGCOMM, Computer Communication Review. Cambridge: [s n], 1999: 147 - 160.

[2] OVERMARS M H, STAPPEN A F VAN DER. Range searching and point location among fat objects[J]. Journal of Algorithms, 1996, 21(3): 629 - 656.

[3] GUPTA P, MCKEOWN N. Packet classification using hierarchical Intelligent cuttings[J]. IEEE Micro, 2000, 20(1): 34 - 41.

[4]. BABOESCU F, VARGHESE G. Scalable packet classification[C]//Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. San Diego: [s n], 2001: 199 - 210.

[5] SRINIVASAN V, VARGHESE G, SURI S, *et al*. Fast and scalable layer four switching[C]//Proc SIGCOMM. Vancouver: [s n], 1998: 203 - 214.

[6] SRINIVASAN V, SURI S, VARGHESE G. Packet classification using tuple space search[C]//Proc SIGCOMM. Cambridge: [s n], 1999: 135 - 146.

[7] GUPTA P, MCKEOWN N. Algorithms for packet classification[J]. IEEE Network Special Issue, 2001, 15(2): 24 - 32.

[8] 王韬,龚俭,陆晟. 高速 IDS中的高维报文分类裁减策略[J]. 东南大学学报(自然科学版增刊), 2002, 32: 338 - 341.

[9] BABOESCU F, SINGH S, VARGHESE G, *et al*. Packet classification for core routers: Is there an alternative to CAMs? [C]//IEEE INFOCOM. San Francisco: [s n], 2003.

[10] 程光,龚俭,丁伟. 基于抽样检测的高速网络实时异常检测模型[J]. 软件学报, 2003, 14(3): 594 - 599.

(编辑 杨 波)