

基于多维空间动态划分与 RFC 的包分类改进算法

陈小雨^{1,2}, 陆月明^{1,2}

(1. 北京邮电大学信息与通信工程学院, 北京 100876;
2. 可信分布式计算与服务教育部重点实验室, 北京 100876)

摘要: 针对 RFC 算法随着规则集规模的增加, 占用的内存空间以近似指数规模骤然增大的问题, 提出了一种改进型的包分类算法 HRFC (Hybrid-RFC)。该算法通过决策树完成规则集多维空间的动态划分, 借助多阶段缩减树完成对每个子集的映射, 从而实现包的快速高效分类。实验表明, 该算法能够在保障分类速度的同时, 有效地降低空间开销。

关键词: RFC; 包分类; 决策树; 划分

中图分类号: TP393.08

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018024

Improved packet classification algorithm based on multidimensional space dynamic division and RFC

CHEN Xiaoyu^{1,2}, LU Yueming^{1,2}

1. School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
2. Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing 100876, China

Abstract: According to the existing problem that memory usage grows exponentially with the size increase of rule set in RFC (recursive flow classification) algorithm, an improved packet classification algorithm, HRFC (Hybrid-RFC) was put forward. The new algorithm completes the dynamic division of a multidimensional space rule set by a decision tree, accomplishes the mapping of each subset with multiple phase reduction trees, so as to realize fast and efficient packet classification. The simulation results show that the new algorithm can reduce the space usage effectively while guaranteeing the performance of classification speed.

Key words: RFC, packet classification, decision tree, division

1 引言

随着互联网架构的不断演进、互联网应用的不
断涌现以及互联网用户的爆发式增长, 基于单
一地址的传统路由技术已经不能满足日益增长
的网络业务和网络安全需求。而数据分组分类
能够依据数据分组的包头信息对网络流量进行
划分,

进而提供差异式服务, 这使该技术在路由器、
安全网关、流量控制器等各类网络安全与管理
设备中得到了广泛应用^[1,2]。与此同时, 随着
数据中心网络、软件定义网络、天地一体化网
络等前沿网络技术的发展, 高性能的数据分组
分类技术也成为下一代网络发展和演进中的
研究热点^[3,4]。

简单来说, 包分类就是将数据分组的包头信

收稿日期: 2018-01-03; 修回日期: 2018-02-15

通信作者: 陈小雨, chenxiaoyu12345@126.com

基金项目: 国家重点研究计划基金资助项目 (No.2016YFB0800302)

Foundation Item: The National Key R&D Program of China (No.2016YFB0800302)

息与预设的规则集进行匹配,从而决定该数据分组所属的数据流的过程。依据规则表示方式的不同,匹配的方式分为精确匹配、范围匹配和前缀匹配3种。假设规则集总数为 N ,每条规则包含 d 个维度,对于一个到达的数据分组 P ,如果存在一条规则 T ,对于任意的 $i(1 \leq i \leq d)$,均满足包头信息 $P[i]$ 匹配规则信息 $T[i]$,则称数据分组 P 与规则 T 匹配。之后,便可根据与 T 相关联的方法对 P 进行处理。

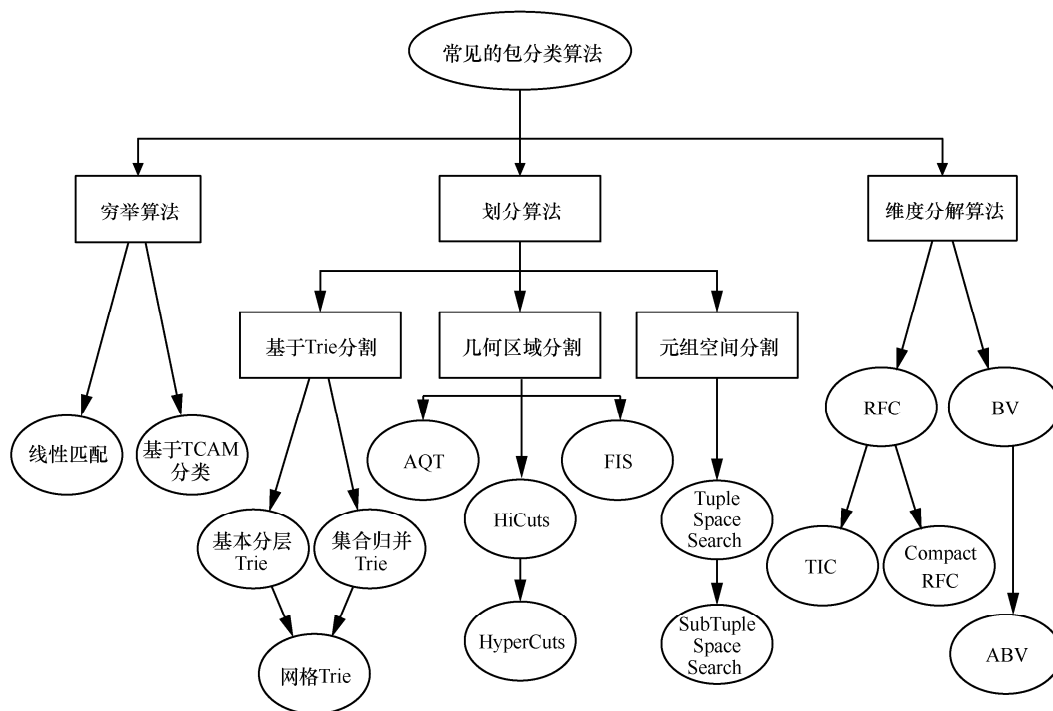
由于包分类算法的重要性和应用的广泛性,很多研究人员进行了研究,也出现了很多的优秀算法,如图1所示。传统的包分类一般使用线性查找,在规则数目比较少的情况下这是最佳的方法;但当规则集规模较大时,该算法时间性能会非常差。基于三态内容寻址存储器(TCAM, ternary content addressable memory)的大规模并行查找^[5]也是穷举法的一种,该方法是目前速度最快的包分类算法,但由于是基于硬件实现,成本较高,并不适合大范围的应用;基于Trie树分割^[6-8]的算法通过规则的前缀构建Trie树,算法实现简单,但对规则维数的扩展性差,且不支持范围匹配;基于元组空间的二元组空间查找(TSS, tuple space search)算法^[9]根据前缀长度的不同,通过

预计算对规则进行分组,这种方法空间占用率小、平均性能好,但是Hash函数的使用使其性能不稳定。基于维度分解的算法是软件实现的算法中速度较快的,但是随着规则集规模的增大,空间消耗在最差情况下会以指数的形式骤增,空间代价较高,此类别的代表算法有交叉乘积^[8]、RFC^[10]、ABV (aggregated bit vector) ^[11]等;基于几何分割的HiCuts^[12]、HyperCuts^[13]等算法把规则集映射为多维空间,使用决策树实现规则集的分割,通过几何搜索完成包的分类。

目前,RFC算法是除硬件方案之外较快的多维数据分组分类算法,但该算法也存在较多缺陷。Pak^[14]致力于完善RFC递归树的快速构建,Trivedi等^[15]解决了RFC的增量更新问题;本文的关注点为RFC随着规则集规模的增加而可能出现的内存爆炸问题^[10]。本文所提的算法结合多维空间的几何区域分割与RFC算法的优点,能够达到在牺牲较小分类速度的同时,大大改善空间消耗的效果。

2 HRFC 算法

对较大规模的规则集进行处理时,RFC空间消耗十分巨大,造成这一问题的主要原因是交叉



乘积表会随着规则数的增大而以近似指数的规模增长^[10]。本文从这一点出发进行改进，提出了 HRFC (Hybrid-RFC) 算法。该算法采用化大为小的思想对规则集进行预处理，会牺牲一定的分类速度，但能够实现对空间消耗的显著优化。

2.1 主要思想

HRFC 算法通过决策树^[14]完成规则集的划分，之后对每个规则子集构建多阶段缩减树。对于决策树的中间节点，本算法根据规则集的特点不断对其所包含规则的某个维度进行动态切割，直到每个节点所包含的规则数目均小于阈值参数 th ；对于决策树的叶节点，构建多阶段缩减树，完成规则子集的快速匹配。算法的整体结构如图 2 所示。当参数 th 大于等于原始规则集的规则总数时，HRFC 会转化为标准的 RFC 算法。因此，RFC 可看作本算法的一种特例。

2.2 规则集的划分

依据功能差异，用于对 k 维规则集进行划分的决策树的节点分为中间节点与叶节点。中间节点不断对规则集进行划分，直至规则子集总数满足规模要求。对于叶节点，主要完成其所含规则子集的多阶段缩减树的构建。

对于任意的一个中间节点 v ，包含如下属性。

1) $R(v)$: 节点 v 包含的规则集；特别地，树的根节点 S 包含所有的规则。

2) D : 被选中用于划分规则子集的维度。规定每个维度仅能被选用一次。

3) $C(v)$: 针对维度 D 的划分点集合 $\{c_0, c_1, \dots, c_{m-1}\}$ ；假设 $R(v)$ 组成的多维空间在维度 D 的取值为 $[0, n]$ ，那么包含 m 个划分点的 $C(v)$ 完成了 $[0, n] \rightarrow [0, c_0][c_0, c_1] \dots [c_{m-1}, n]$ 的区间划分。

4) $U(v)$: 子节点集合 $\{u_0, u_1, \dots, u_{m-1}\}$ 。每一个子节点均与一个划分区间相对应，存储满足该划分区间的要求规则。

中间节点首先需要选取用于划分规则子集的维度 D ，之后确定划分点集合 $C(v)$ ，并依据 $C(v)$ 完成规则集的划分，下面具体阐述。

2.2.1 维度选取

对于规则集 S ，将其在维度 D 上大于等于 a 且小于 b 的规则所组成的集合记为 $S_D(a, b)$ ；特别地， $S_D(a, a+1)$ 表示在维度 D 上满足值为 a 的所有规则，将这样的规则集称为 S 在维度 D 上的最小规则集，记为 $S_D(a)$ ； $RN(S_D(a, b))$ 表示规则集 $S_D(a, b)$ 所包含的规则数目。

假设规则集 S 在维度 D 的取值范围为 $[0, n]$ ，将 $RN(S_D(i))$ 的归一化结果记为 P_i 。

$$P_i = \frac{RN(S_D(i))}{\sum_{j=0}^{n-1} RN(S_D(j))}, 0 \leq i < n \quad (1)$$

依据式(1)，定义规则集 S 在维度 D 上的熵 H_D 为

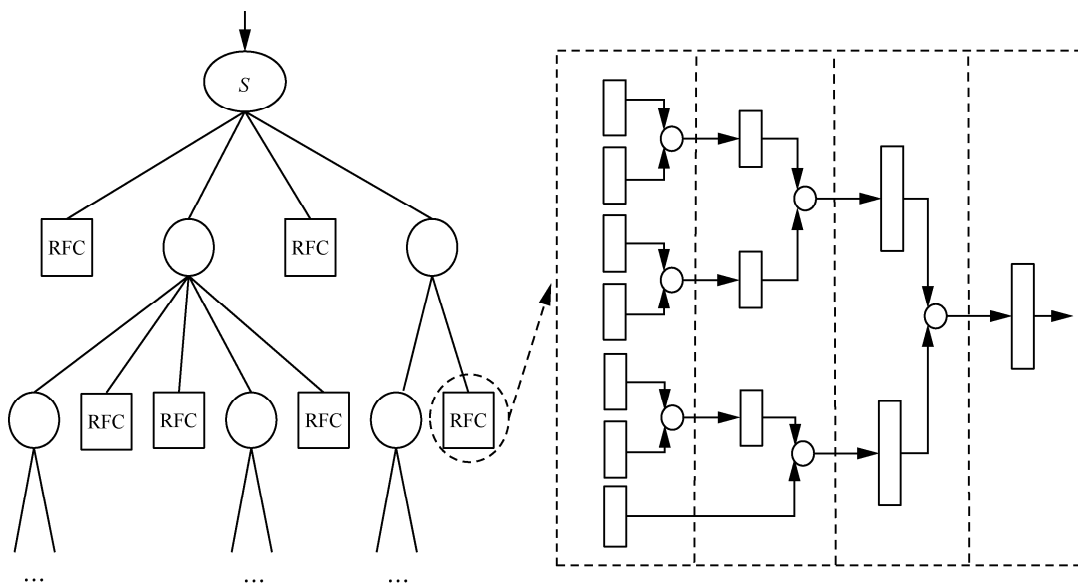


图2 HRFC 算法

$$H_D = E(-\log P) = -\sum_{i=0}^{n-1} P_i \log P_i \quad (2)$$

熵 H_D 能够在一定程度上反应规则子集在维度 D 的分布情况。 H_D 越大, 表明规则子集在该维度的分布越均匀; 反之, 表明规则子集在该维度的分布越集中。划分的目的是尽可能地将规则集分成满足指定规模要求的规则子集, 那么规则集在某个维度分布越均匀, 越容易完成划分, 从而降低决策树的高度, 减小分类的时间消耗。因此, 从这一角度考虑, 熵值 H 的维度越大, 越适用于规则集的划分。

2.2.2 规则集划分

对于在维度 D 上取值范围为 $[0, n]$ 的规则集 S , 划分策略从最小规则子集 $S_D(0)$ 开始, 逐步向右扫描, 更新当前子集 S'_D , 直至其不满足参数 th 的要求时, 生成划分点, 并以该划分点为新的起始点, 重复上述步骤。即

$$S'_D = \begin{cases} S'_D \cup S_D(i), RN(S'_D \cup S_D(i)) < th \\ S'_D, RN(S'_D \cup S_D(i)) \geq th \end{cases} \quad (3)$$

对于无法满足参数 th 划分要求的连续的最小规则子集, 通过合并的方式减小划分点个数, 即

$$S'_D = \begin{cases} S'_D \cup S_D(i), S'_D \geq th, S_D(i) \geq th \\ S'_D, S_D(i) < th \end{cases} \quad (4)$$

图3展示了以目的端口(简记为 DP) 为划分维度, DP 取值范围为 $[a, h]$ 、 th 取值为4时的划分结果。

2.3 多阶段缩减树的构建

假设规则集的维度数为 d , 用 L 表示由这 d 个维度的信息所形成的比特串的长度, 那么该比特串的取值为 $[0, 2^L - 1]$, 共 2^L 种情况。如果将每种

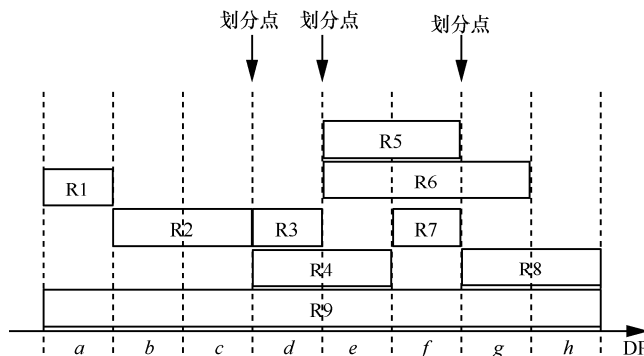


图3 针对维度 DP 进行划分

情况对应的匹配规则预先计算并存储在一个表项中(用 $eqID$ 表示), 那么通过一次内存访问便可得出查找结果。然而在实际的多维包分类问题中, S 通常很大, 以源地址(IPV4的情况下为32 bit)、目的地址(32 bit)、源端口(16 bit)、目的端口(16 bit)、协议号(8 bit)组成的五元组为例, 此时 d 为5, L 为104 bit, 需要存储 $2^{104} \approx 10^{31}$ 个 $eqID$ 表项; 存储这个数量级的表项显然是不现实的。

为了减小 $eqID$ 表项的存储, 可将上述过程中的一次映射转变为多阶段映射, 每一个阶段映射都将较大的集合映射成若干个较小的集合, 并将映射结果用交叉乘积表^[8]组合起来, 如此反复迭代, 最终完成所有的长为 L 的比特串到 $eqID$ 的映射。算法在执行过程中, 由多阶段映射所构建的数据结构称为多阶段缩减树^[10]。

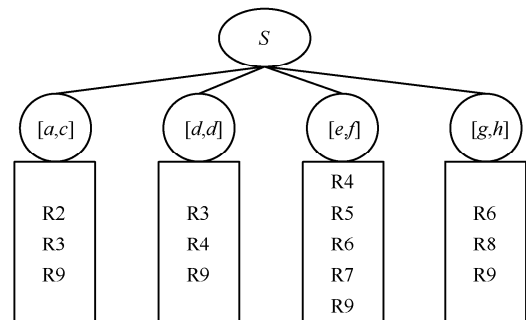
图4展示了针对表1所示的二维规则集构建的二阶段缩减树, 以及对含有二维信息 $\langle 001, 100 \rangle$ 的对象与表1所示的规则集的匹配过程。

表1	二维规则集示例	
规则	F1(3 bit)	F2(3 bit)
R1	001	001
R2	01*	0**
R3	001	100
R4	01*	100
R5	01*	10*
R6	***	0**

2.4 查找分类过程

查找分类过程可分为确定规则子集和通过缩减树匹配这2部分。

规则子集的确定是通过预处理部分所创建的决策树完成的。首先提取待匹配的数据分组的五



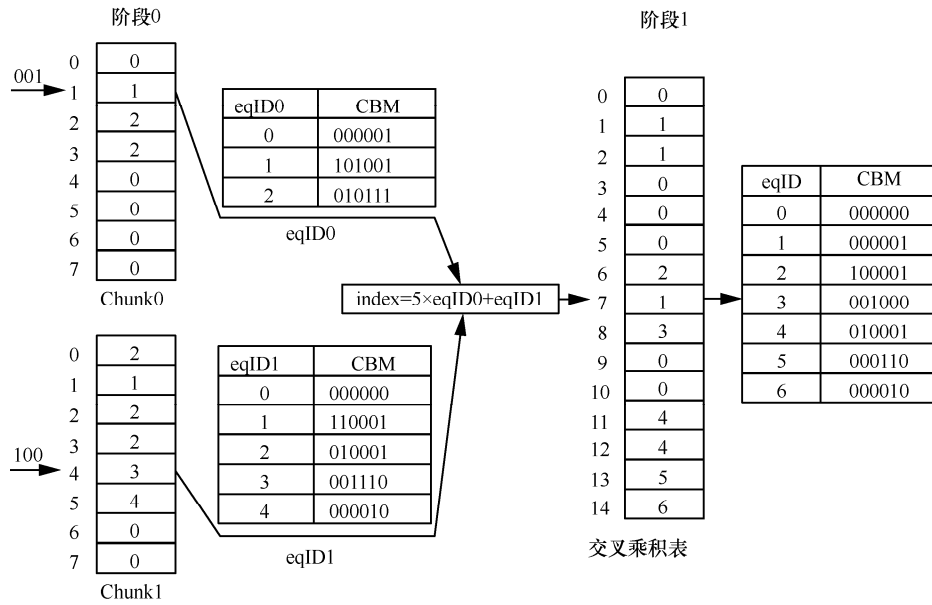


图4 构建多阶段缩减树

元组信息，然后从决策树的根节点开始，逐层做决策，直到叶子节点为止。在叶子节点处，通过与之相连的多阶段缩减树完成数据分组信息到规则的映射与匹配。

需要说明的是，当规则表中有不止一条规则匹配数据分组时，可能出现规则冲突。对于这一问题，本算法借助最优匹配加以解决，也可以使用最长前缀匹配或其他方法，该问题非本文讨论重点，不做过多讨论。查询算法的伪代码如算法1所示。

算法1 HRFC 的查找分类算法伪代码

输入 root, packet

输出 rule

```

1) SA0, SA1, DA0, DA1, DP, PROT = getInfo(packet)
2) curNode = root
3) while ! curNode.isLeaf then
4)   for i ← 0 to root.conditionLen-1
5)     if curNode.getCondition(i). satisfied(packet) then
6)       curNode = curNode.getChild(i)
7)       Break
8)     End if
9)   End for
10) End while
11) index00 = curNode.rfc.phase0(SA0, SA1)

```

12) index01 = curNode.rfc.phase0(DA0, DA1)

13) index02 = curNode.rfc.phase0(SP, DP)

14) index10 = curNode.rfc.phase1 (index00, index01)

15) index11 = curNode.rfc.phase1 (index02, PROT)

16) index2 = curNode.rfc.phase2 (index10, index11)

17) rule = curNode.rfc.phase3(index2)

18) return rule

3 实验与分析

实验中所使用的规则集及对应分类测试数据分组均由 ClassBench^[16]产生。ClassBench 是由华盛顿大学路易斯分校的 Taylor 团队所开发的开源工具。该工具被数据分组分类研究广泛使用，是目前较为权威的规则集产生与测试工具。实验中使用的规则集如表2所示。

3.1 规则总数对空间消耗的影响

使用 ClassBench 生成 1 000~10 000 条规则，测试 HRFC 的预处理过程，即决策树与多阶段缩减树构建的空间消耗，并与 RFC 算法进行对比。实验中，HRFC 用于划分规则集的阈值 th 设为规则总数的 $\frac{1}{4}$ ，如对于规则总数为 4 000 条的测试集，规则划分的阈值 th 为 1 000。

表 2

实验规则集示例

规则	SA	DA	SP	DP	PROT
R0	239.3.196.91/32	13.77.63.217/32	161:161	80:65535	0x11/0xFF
R1	135.140.20.248/32	188.244.7.189/32	53:53	42:42	0x06/0xFF
R2	122.210.30.184/29	188.244.7.190/31	1024:65535	514:514	0x06/0xFF
R3	238.0.0.0/7	128.0.0.0/1	0:65535	1024:65535	0x11/0xFF
R4	239.235.252.28/32	201.217.36.173/32	0:65535	5190:5190	0x06/0xFF
R5	88.8.166.232/29	81.111.231.128/27	123:123	0:65535	0x06/0xFF
R6	128.0.0.0/1	56.13.197.53/32	69:69	80:80	0x2f/0xFF
R7	0.0.0.0/0	187.72.88.0/22	0:65535	80:80	0x01/0xFF
R8	0.0.0.0/1	128.71.176.0/21	0:65535	0:65535	0x00/0x00
...	...	...	...	...	...

分析图 5 可知,在规则数较小时(如 1 000、2 000、3 000),HRFC 的空间优化效果并不明显,空间消耗量与 RFC 算法相差无几。而在规则数较大时(如 7 000、8 000),HRFC 的空间开销要明显优于 RFC。随着规则总数的增加,RFC 的空间消耗会发生骤增,而 HRFC 通过规则集的划分,有效地改善了这一问题,且规则总数越大,改善效果越明显。

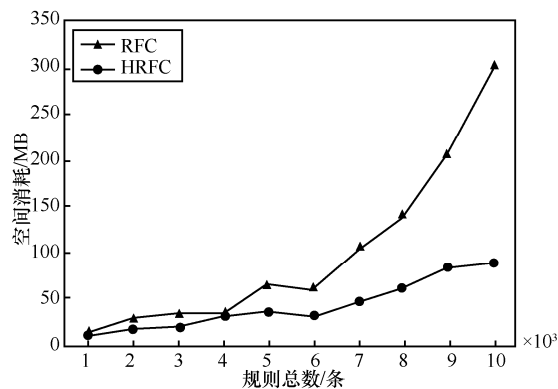


图 5 HRFC 与 RFC 的空间消耗比较

3.2 规则总数对分类时间的影响

在 3.1 节实验所构建结构的基础之上,本节进一步测试了 2 种算法的分类时间。每种规则集各进行 100 次分类测试,结果如图 6 所示。

HRFC 的分类速度要略慢于 RFC,这主要是由决策树的决策判断所造成的。此外,HRFC 的分类速度波动性稍大,这是由于分类速度与被匹配规则在决策树中的位置有关;被匹配规则所在节

点的深度越低,分类速度越快,而被匹配规则的深度是不定的,这造成了 HRFC 分类速度的波动,但总体来说,HRFC 分类速度曲线的趋势与 RFC 类似,并未随着规则总数的增加而骤增。

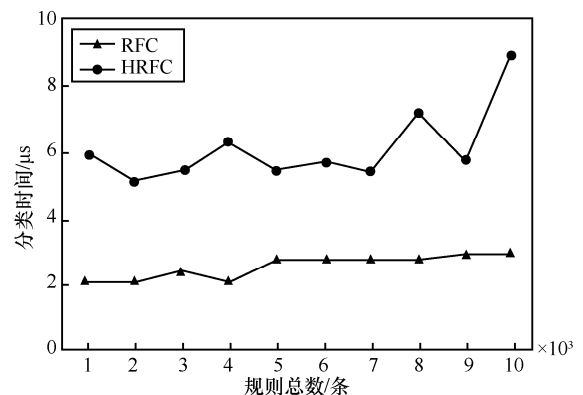


图 6 HRFC 与 RFC 的分类时间比较

3.3 阈值 th 对空间消耗及分类时间的影响

上述实验的 HRFC 算法中,用于划分规则集的阈值 th 均取的是规则总数的 $\frac{1}{4}$,但其实这一数

值并非最佳选择。如果 th 的取值过大,甚至接近规则总数,那么 HRFC 算法在性能上近似于 RFC 算法;特别地,当 th 大于等于规则总数时,HRFC 会转化为 RFC,两者无区别。如果 th 取值过小,会造成决策树深度过大,规则子集过多。不仅可能造成空间消耗增加,深度过高的决策树还会影响分类速度。因此,HRFC 算法的预处理过程空间消耗的最小值会在曲线中部的某一位置出现,

而分类时间曲线会随着阈值的增加而呈现缓慢减小的趋势, 仿真实验也进一步证实了这一点, 如图 7、图 8 所示。

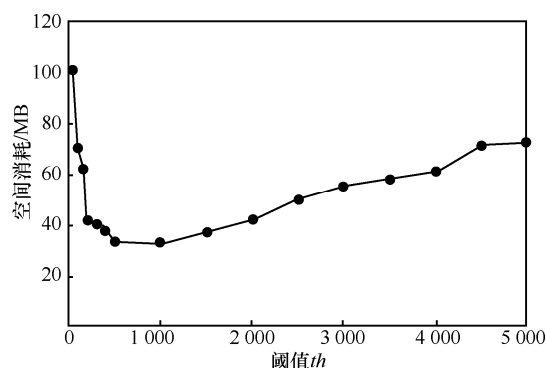


图7 阈值 th 对 HRFC 空间消耗的影响 (规则总数为 5 000 条)

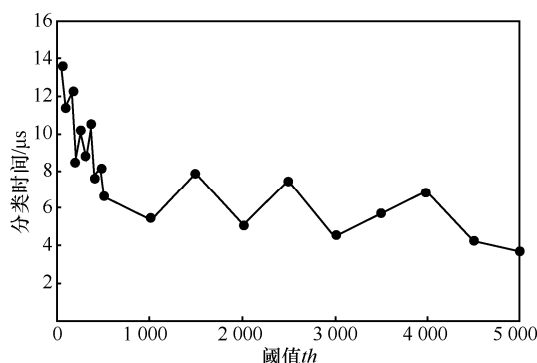


图8 阈值 th 对 HRFC 分类时间的影响 (规则总数为 5 000 条)

4 结束语

针对 RFC 算法在结构构建时空间开销大的缺陷, 本文提出了一种改进型算法 HRFC。该算法将规则集看作是多维空间, 借助于决策树对规则集进行切分, 而后对每个子集进行缩减树的构建。该算法牺牲了一定的包分类速度, 却能够有效地降低结构构建阶段的空间开销。此外, 本文也就规则集划分参数对性能的影响进行了探讨。

参考文献:

- [1] 林闯, 单志广, 任丰原. 计算机网络的服务质量[M]. 北京: 清华大学出版社, 2004.
LIN C, SHAN Z G, REN F Y. Service quality of computer network[M]. Beijing: Tsinghua University Press, 2004.
- [2] JOSEPH D A, TAVAKOLI A, STOICA I. A policy-aware switching layer for data centers[C]//ACM SIGCOMM 2008 Conference on Data Communication. 2008: 51-62.
- [3] 万云凯, 嵩天, 刘苗苗, 等. 流量自适应的多维度包分类方法研究[J]. 计算机学报, 2017, 40(7): 543-1555.
WANG Y K, SONG T, LIU M M, et al. Study on multidimensional

- packet classification method of Flow self-adaptation[J]. Chinese Journal of Computers, 2017, 40(7): 1543-1555.
- [4] 肖玮, 陈性元, 包义保, 等. 基于多级关联信号树的高效可重构网包分类方法研究[J]. 高技术通讯, 2014, 24(9): 928-934.
XIAO W, CHEN X Y, BAO Y B, et al. The efficient econfigurable network packet classification method based on multilevel correlation signal tree[J]. Chinese High Technology Letters, 2014, 24(9): 928-934.
- [5] ZANE F, NARLIKAR G, BASU A. Coolcams: power-efficient TCAMs for forwarding engines[C]//Joint Conference of the IEEE Computer and Communications. 2003: 42-52.
- [6] KNUTH D. Fundamental algorithms: sorting and searching[DB/OL]. https://www.researchgate.net/publication/243573812_Fundamental_algorithms_vol_3_sorting_and_searching.
- [7] TSUCHIYA P F. A search algorithm for table entries with non-contiguous wildcarding[DB/OL]. https://www.researchgate.net/publication/2804328_A_Search_Algorithm_for_Table_Entries_with_Non-contiguous_Wildcarding.
- [8] SRINIVASAN V, VARGHESE G, SURI S, et al. Fast and scalable layer four switching[C]//ACM Sigcomm'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. 1998: 191-202.
- [9] SRINIVASAN V, SURI S, VARGHESE G. Packet classification using tuple space search[J]. Sigcomm Computer Communication Review, 1999, 29(4): 135-146.
- [10] GUPTA P, MCKEOWN N. Packet classification on multiple fields[J]. Sigcomm Computer Communication Review, 1999, 29(4): 147-160.
- [11] LAKSHMAN T V, STILIADIS D. High-speed policy-based packet forwarding using efficient multi-dimensional range matching[DB/OL]. https://www.researchgate.net/publication/2505564_High-Speed_Policy-based_Packet_Forwarding_Using_Efficient_Multi-dimensional_Range_Matching.
- [12] GUPTA P, MCKEOWN N. Classifying packets with hierarchical intelligent cuttings[EB/OL]. https://www.researchgate.net/publication/3215123_Classifying_packets_with_hierarchical_intelligent_cuttings.
- [13] SINGH S, BABOESCU F, VARGHESE G, et al. Packet classification using multidimensional cutting[C]//Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. 2003: 213-224.
- [14] PAK W, BAHK S. FRFC: fast table building algorithm for recursive flow classification[J]. IEEE Communications Letters, 2010, 14(12): 1182-1184.
- [15] TRIVEDI U, JANGIR M L. An optimized RFC algorithm with incremental update[C]//The International Conference on Advances in Computing, Communications and Informatics. 2014: 120-127.
- [16] TAYLOR D E. ClassBench: a packet classification benchmark[J]. IEEE/ACM Transactions on Networking, 2007, 15(3): 499-511.

[作者简介]



陈小雨 (1993-), 男, 北京人, 北京邮电大学硕士生, 主要研究方向为分布式计算与信息安全。

陆月明 (1969-), 男, 江苏苏州人, 北京邮电大学教授、博士生导师, 主要研究方向为分布式计算、信息安全。