多决策树包分类算法

李振强[①] 张圣亮[②] 马 严[①②] 赵晓宇[③]
[①](北京邮电大学计算机科学与技术学院 北京 100876)
[②](北京邮电大学信息网络中心 北京 100876)
[③](法国电信北京研发中心 北京 100080)

摘 要：网络安全、网络测量、服务质量、流路由等都离不开多维包分类算法。设计一种在时间上和空间上都很好的包分类算法非常困难。该文在分析大规模规则集的特征的基础上，根据协议类型域只有有限的几种取值的特点，提出一种多决策树包分类算法。该算法既可用软件实现，也适宜硬件实现，并且在理论上适用于 IPv6 的包分类。当采用硬件实现时，多棵树可以并行查找，树内查找可以采用流水结构，算法的查找复杂度为 $O(1)$。该算法可用于改进广泛应用的 HiCuts 和 HyperCuts 算法，与之相比，多决策树算法在预处理时间、内存占用和查找时间上都有很大提高。

关键词：包分类；决策树；服务质量；流路由
中图分类号：TP393.03 **文献标识码**：A **文章编号**：1009-5896(2008)04-0975-04

Multiple Decision Tree Algorithm for Packet Classification

Li Zhen-qiang[①] Zhang Sheng-liang[②] Ma Yan[①②] Zhao Xiao-yu[③]
[①](*School of Computer Science and Technology*, *Beijing Univ. of Posts and Telecomm.*, *Beijing* 100876, *China*)
[②](*Network Information Center*, *Beijing Univ. of Posts and Telecomm.*, *Beijing* 100876, *China*)
[③](*France Telecom. R&D Center of Beijing*, *Beijing* 100080, *China*)

Abstract: Multiple dimension packet classification is an enabling function for many Internet applications, such as network security, network monitoring, quality of service, flow routing, etc. It is difficult to develop a packet classification algorithm that is efficient in both space and time. Based on the observation that there are only a few possible values for the protocol field in the classifier, a Multiple Decision Tree (MDT) algorithm is proposed. This algorithm is suitable for both software and hardware implementation, and is applicable for IPv6 in theory. When MDT is implemented with hardware, the multiple tree can be searched in parallel and pipeline can be used to search within a specific tree, thus the search complexity of MDT is $O(1)$. With advantages in preprocessing time, memory consumption, and search time, MDT can be used to improve the widely used HiCuts and HyperCuts algorithms.

Key words: Packet classification; Decision tree; Quality of Service (QoS); Flow routing

1 引言

在计算机网络中很多应用都离不开包分类(Packet Classification, PC)，例如网络安全，网络测量，服务质量(Quality of Service, QoS)，流量整形，负载均衡，流路由(Flow Routing, FR)[1]等。这些应用都是基于流(Flow)的，因此，网络设备(路由器、防火墙、整形器等)必须将收到的数据包匹配到相应的流中，然后再执行和该流相关联的操作。包分类就是用来完成数据包和规则集的匹配的，它从数据包中取出相关的域，在规则集中进行查找，最后返回与该数据包相匹配的优先级最高的规则或者与该规则相关联的操作。

流往往由数据包中的多个域决定，这些域通常包括源 IP 地址、目的 IP 地址、源端口、目的端口和协议类型(一般称为 5 元组)，还可能包括 TCP 标志位、服务类型等，在 IPv6 中还会使用流标签。因此，包分类是基于这些域的多维分类算法(与之相对应的基于目的 IP 地址的路由查找算法是一维分类算法)，并且规则集中 IP 地址和端口往往用前缀和范围表示，这增加了算法设计的难度，设计出一个在空间上和时间上都很好的包分类算法非常困难。新的多维包分类算法必须以尽量少的内存消耗和尽量快的查找速度来支持含有更多规则(包括 IPv6 规则)的规则集。

不失一般性，本文讨论基于 5 元组的包分类算法。通过分析大规模规则集的特征，发现协议类型域只有通配符和有限几种精确取值，据此提出多决策树 MDT(Multiple Decision Tree) 算法。 MDT 首先根据协议类型域的值将规

2006-09-20 收到，2007-02-15 改回
法国电信北京研发中心资助课题

则集分成若干子集，然后为每一个子集建立一棵决策树。MDT 既可用软件实现，也可用硬件实现。当采用硬件实现时，多棵树可以并行查找，树内查找可以采用流水结构，算法的查找复杂度为 $O(1)$。MDT 不依赖具体的 IP 地址结构，对 IP 地址的长度不敏感，因此在理论上也适用于 IPv6 的包分类。与广泛使用的 HiCuts 算法[2]相比，MDT 在预处理时间、内存占用和查找效率上都有很大提高。

本文其他部分的内容如下：第 2 节总结了规则集的特征；第 3 节详述了本文提出的多决策树算法；多决策树算法的性能分析及其与 HiCuts 算法的比较在第 4 节给出；第 5 节是结束语。

2 规则集的特征

网络中实际应用的规则集往往位于防火墙或者路由器中，由于安全等方面的原因这些规则集很难获得，一些研究机构在保密协议的约束下获得后也不能将其公开，这给包分类算法研究和仿真验证带来很多不便。此外，真实的规则集一般只有几百条规则，规模较小，不足以验证包分类算法的扩展性。通过分析和建模，华盛顿大学圣路易斯分校的 Taylor 和 Turner 开发了源码开放的 ClassBench[3]，用于产生符合真实规则集特征的不同规模的规则集，以供学术界和产业界研究和验证包分类算法使用。本文该部分对规则集特征的分析就是基于 ClassBench 产生的规则集，本文第 4 节对算法性能验证时使用的不同大小的规则集也都是由 ClassBench 产生的。

2.1 前缀、区间和精确匹配

规则集中的规则一般由数据包中的某些域(通常是上文中提到的 5 元组)及规则对应的行为(允许、拒绝、SLA 等)组成。ClassBench 产生的规则集中不含规则行为，因为规则行为往往与规则的具体应用场合相关，对于研究包分类算法，不含规则行为的规则集已经足够。ClassBench 产生的规则集中的每条规则由 6 个域组成：源 IP 地址、目的 IP 地址、源端口、目的端口、协议类型和标志域。标志域和协议类型相关，一般是 TCP 的标志位或者 ICMP 的类型，本文只研究除标志域外的 5 元组。

不同的域有不同的表示方法。IP 地址一般以前缀表示，端口一般用范围表示，而协议类型一般是某个精确取值。表 1 是一个示例规则集，其中包含 5 条规则。

表 1 一个包含 5 条规则的规则集

源 IP	目的 IP	源端口	目的端口	协议
21.25.0.0/16	21.25.13.5/32	*	=80	TCP
0.0.0.0/0	59.64.11.0/24	*	*	ICMP
0.0.0.0/0	152.68.0.0/16	<1024	>1023	TCP
21.25.32.0/24	21.25.37.0/24	700−800	800−1000	UDP
0.0.0.0/0	0.0.0.0/0	*	*	*

2.2 协议类型域的特征

协议类型域(在 IPv6 中称为 Next Header)为 8bit，共有 256 种取值，由 IANA 负责统一分配，目前已经分配了 140 多个[4]。但不是所有的取值都会出现在规则集中。规则集中的协议类型要么是通配符，要么是以下 9 种取值中的一个：6 (TCP，Transmission Control Protocol)， 17 (UDP，User Datagram Protocol)， 1 (ICMP，Internet Control Message Protocol)，47 (GRE， General Routing Encapsulation)，89 (OSPF，Open Shortest Path First)，88 (EIGRP，Enhanced Interior Gateway Routing Protocol)，50 (ESP for IPv6， Encapsulating Security Payload)， 51 (AH for IPv6，Authentication Header)， 4 (IP in IP Encapsulation)。各种协议类型的规则在规则集中所占的比重差别很大，如图 1 所示。TCP 占了将近一半(49%)，其次是 UDP(27%)，通配符(13%)，然后是 ICMP(10%)，最后，所有其他协议类型的规则加起来约占 1%，并且这些类型的规则不是在所有的规则集中都会出现。

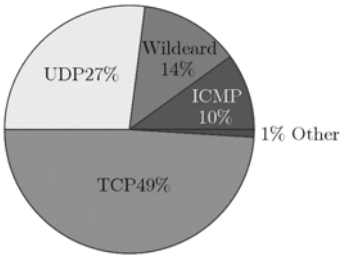

图 1 各种协议类型的规则在规则集中所占的比重

3 多决策树算法

多决策树算法的总体思想如图 2 所示。根据协议类型域的取值对规则集进行划分；协议类型为 TCP，UDP，ICMP 和任意的规则因为数量较多，每种协议类型的规则单独作为一个子集，并用决策树进行组织；协议类型为其他的规则因为数量较少，不再对每种协议进行单独组织，而直接将它们组织成线性表结构。决策树的构造可以使用已有的算法，例如 Modular[5]，HiCuts[2]，HyperCuts[6]等。这里需要注意的是，在本文的算法中，协议类型为通配符的规则被当作一种

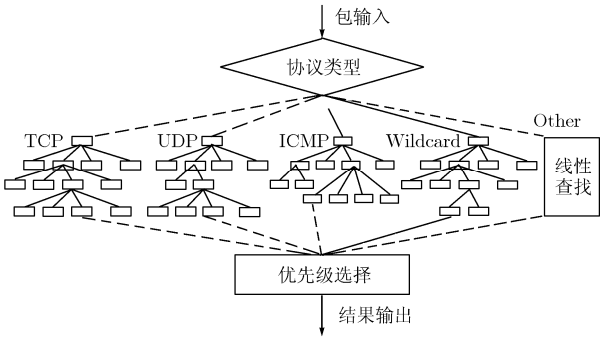

图 2 多决策树算法

特殊类型的规则也用一棵单独的决策树来组织，而没有将它们复制到其它决策树和线性表中。这一方面降低了决策树的高度，提高了算法的性能，另一方面也降低了算法的内存消耗。多决策树构造好之后，查找时不管数据包的协议类型是什么都在协议类型为任意的决策树上查找(在图 2 中用实线表示)，同时在相应协议类型的决策树上或者线性表中查找，最后比较两者的查找结果，返回优先级最高的规则作为最后的查找结果。

3.1 构造决策树

决策树的构造在根据协议类型划分得到的子集中进行。因为参与决策树构造的规则已经由 5 维降到了 4 维，并且规则的数量也大大减少，即使是比例最大的 TCP 规则也不足原有规则数量的一半，所以决策树的构造过程大大加快，决策树的高度也随之降低。图 3 是构造多决策树的伪代码。

```
BuildMDT(classifier)
{
  Partition classifier into subset based on protocol;
  For each subset do
  if (subset.protocol == TCP, UDP, ICMP or * )
      BuildDecisionTree(subset);
else
     BuildLinearList(subset);
}

BuildDecisionTree(subset)
{
  if (|subset| < leafSize){
    BuildLeafNode(subset);
  return;
}
  BuildInternalNode(subset);
N=0;
  for each dimension in subset do{
      N^i = number of distinct values of dimension i;
      N +=N^i;
}
  N /= 4;
  Cut = 1;
  for each N^i > N do{
  Cut^i = optimal number of cuts on dimension i;
  Cut *= Cut^i;
}
  for (i = 0; i < Cut; i++){
  subset^i = do the i^th Cutting;
    BuildDecisionTree(subset^i);
}
  return;
}
```

图 3　构造多决策树的伪代码

3.2 查找

多决策树构造完成后，查找过程可以用软件实现也可以用硬件实现。当用硬件实现时，根据匹配包的协议类型可以同时在相应协议类型的决策树和通配符决策树中并行查找，查找结果送优先级选择器进行比较，优先级选择器将优先级较高的规则输出作为最后的匹配结果，如图 2 所示。在某棵决策树内部，如果将决策树同一层次的节点存放在独立的内存中，决策树的一个层次可以作为流水结构的一个阶段。这样，在决策树上查找一次只相当于一次访存，整个多决策树的查找过程只有并行的两个流水查找加上一次优先级选择，算法的查找复杂度为 $O(1)$。

如果查找过程用软件实现，每棵树的查找可以使用一个独立的线程，优先级选择使用一个线程。收到数据包后，根据数据包的协议类型将数据包交相应的线程进行查找，查找结果作为优先级选择线程的输入，最后由优先级选择线程将优先级较高的规则作为整个查找的结果输出。这种结构特别适合在多线程多核(Hyper threading multi-core) CPU[7]上实现。为了和 HiCuts 算法进行公平比较，第 4 节中使用的多决策树算法只用单线程实现，它的查找过程的伪代码如图 4 所示。

```
SearchMDT(pkt)
{
  tree = wildcard tree;
  if (tree.ruleNum < leafSize)
    rule1 = LinearSearch(tree, pkt);
  else
    rule1 = DecisionTreeSearch(tree, pkt);

  tree = pkt-specific protocol tree;
  if (tree.ruleNum < leafSize)
    rule2 = LinearSearch(tree, pkt);
      else
    rule 2 = DecisionTreeSearch(tree, pkt);
  return rule1.priority > rule2.priority ? rule1 : rule2;
}
```

图 4　多决策树查找的伪代码

4　性能测试与比较

为了验证多决策树包分类算法的性能，在 Linux 操作系统中开发了一个软件原形系统，原形系统中决策树的构造和查找都使用 HiCuts 算法，为了和 HiCuts 算法进行公平比较，本文的多决策树算法只使用单线程实现。当使用多线程或者硬件(并行加流水)实现时，多决策树算法在查找速度上会有更大提高。测试在一台普通 PC 机上进行，PC 机的配置为一个 AMD1.1GHz CPU，512M 内存。测试中使用了不同大小规则集，从 1000 条到 10000 条(注意本节中所有图的横坐标都以 k 为单位)，这些规则集都由 ClassBench[3]产生。

图 5 是多决策树算法与 HiCuts 算法在预处理时间(决策树构造时间)上的比较。对于多决策树算法，构造时间是所有决策树构造时间的总和。从图 5 可以看出，规则集中的规则越多，多决策树算法在构造时间上的优势越明显。

多决策树算法与 HiCuts 算法在内存消耗上的比较如图 6 所示。对于多决策树算法，内存消耗是所有决策树占用内存的总和。从图 6 可以看出，多决策树算法并没有因为建立多棵决策树而占用更多的内存，它的内存消耗和 HiCuts 算法相当，且略有减少。

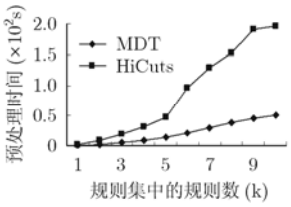

图 5 预处理时间

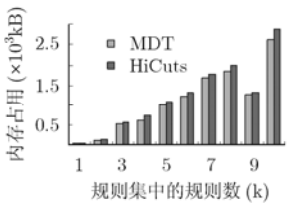

图 6 内存占用

基于决策树的包分类算法，其查找性能主要由决策树的高度决定，平均性能由决策树的平均高度决定，最差性能由决策树的最大高度决定。图 7 是多决策树算法和 HiCuts 算法在决策树高度上的对比，直方图代表最大高度(多决策树的最大高度是高度最大的决策树的高度)，曲线图代表平均高度(多决策树的平均高度是所有树的高度的平均值)。由图 7 可以看出，多决策树算法的最大高度虽然和 HiCuts 算法的相当，但树的平均高度大大降低，规则集中的规则越多两者的差距越大，10000 条规则时，多决策树算法的平均高度(5.75)只有 HiCuts 算法(9)的 64%。

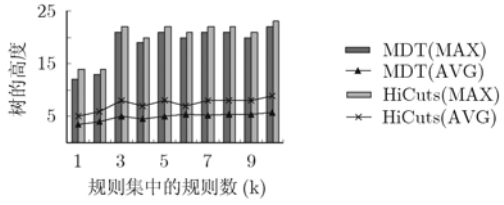

图 7 决策树的最大高度和平均高度

为了验证和比较算法的查找性能，对于每个大小的规则集用随机生成的 100 万个包进行测试。测试和比较结果见图 8。由于测试使用的多决策树算法是我们使用软件实现的一个单线程原型系统，查找时不能并行进行，需要在某种协议类型的决策树和协议为任意的决策树上进行顺序查找，然后比较两棵树的查找结果选出优先级较高的作为最后的查找结果，因此，我们选择多决策树算法查找过程中查找时间较长的那棵树的查找时间来与 HiCuts 算法进行性能比较。从图 8 可以看出多决策树算法的查找性能要优于 HiCuts 算法，当算法用硬件实现时，多决策树算法的这种优势将更加明显。

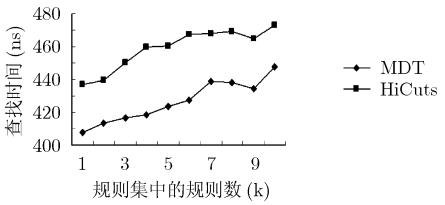

图 8 查找时间

5 结束语

通过分析大规模规则集的特征，发现规则集中的协议类型只有有限的几种取值，据此提出多决策树包分类算法。多决策树算法根据规则集中协议类型域的取值对规则集进行划分，并为划分得到的每一个四维子集建立一棵决策树。查找时，在协议类型为任意的决策树上与相应协议类型的决策树上同时进行，最后比较两者的查找结果，以优先级最高的规则作为最后的查找结果。多决策树算法对 IP 地址的长度不敏感，因此在理论上也可用于 IPv6 的包分类。多决策树算法可以用软件实现，也可以采用并行加流水的结构用硬件实现。与得到广泛应用的 HiCuts 算法相比，多决策树算法在预处理时间、内存占用和查找时间上都有很大提高。

参 考 文 献

[1] Dreibholz T. Flow routing project. 22 April 2006. http: // tdrwww. exp-math.uni-essen.de/dreibholz/ flowrouting/ index. html.

[2] Gupta P and McKeown N. Classification using hierarchical intelligent cuttings. *IEEE Micro*, 2000, 20(1): 34–41.

[3] Taylor D E and Turner J S. ClassBench: A packet classification benchmark. IEEE Infocom 2005, Miami, USA, 2005: 2068−2079.

[4] IANA. Protocol numbers. 28 March 2006. http://www. iana. org/assignments/protocol-numbers.

[5] Woo T Y C. A modular approach to packet classification: algorithms and results. IEEE Infocom 2000, Tel-Aviv, Israel, 2000: 1213–222.

[6] Singh S, Baboescu F, and Varghese G, *et al*.. Packet classification using multidimensional cutting. ACM SIGCOMM 2003, Karlsruhe, Germany, 2003: 213–24.

[7] Carver B. Multi-core technology and solutions. 21 April, 2006. http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/ xeon/238663.htm?page=1.

李振强： 男，1976 年生，博士生，研究领域为网络安全、路由查找和包分类算法.

张圣亮： 男，1982 年生，硕士生，研究领域为包分类算法.

马 严： 男，1955 年生，教授，博士生导师，研究方向为下一代网络协议及应用.

赵晓宇： 男，1977 年生，硕士，工程师，研究方向为网络安全、IP 多媒体子系统.