IP报文分类算法概述*)

谭明锋[1] 高 蕾[1] 龚正虎[1] 徐田荣[2]
(国防科学技术大学计算机学院 长沙 410073)[1] (中国电子设备系统工程公司计算站 北京 100089)[2]

摘 要 Internet的高速发展要求提供高性能的IP流分类算法以更好地为防火墙、QoS、流量工程、资源预留、网络地址转换等提供服务。由于IP报文分类算法的多域特征,因此其具有相当的难度。研究者提出了很多报文分类算法,本文将这些算法概括为5类:基于Trie树的算法、基于空间分割的算法、启发式算法、基于硬件实现的算法和其他算法,并对IP报文分类算法的思想、原理和过程进行了介绍和分析,说明了这些算法之间的联系,并对这些算法在搜索和更新的时间性能、空间性能、适用性范围和优缺点等进行了分析和比较。作为总结,本文还对IP报文分类算法研究的方法和趋势进行了分析和总结。

关键词 IP报文分类,路由器,流量工程,QoS,区分服务

Survey of IP Packet Classification Algorithms

TAN Ming-Feng[1] GAO Lei[1] GONG Zheng-Hu[1] XU Tian-Rong[2]
(School of Computer Science, National University of Defense Technology, Changsha 410073)[1]
(Computing Unit, China Electronic Equipment Systems Engineering Corporation, Beijing 100089)[2]

Abstract The fast incensement of Internet requires high performance IP packet classification algorithms to serve different services, such as fire wall, QoS, traffic engineering, resource reservation, net address translation, etc. IP packet classification is a hard problem due to its multiple dimension character, hence researchers have proposed a variety of algorithms which can be categorized as algorithms based on Tries, algorithms based on space splitting, heuristic algorithms, hardware-specific algorithms and other algorithms. This tutorial describes the main idea and process of these algorithms, and the relationship among them. Also this paper discusses the suitable application for each algorithm, and compares their performance, their strong points and weak points. As the conclusion, this paper summarizes the main research methods and the trends of the IP packet classification algorithms.

Keywords IP routing lookup, Router, Traffic engineering, QoS, DiffServ

1 引言

Internet的高速发展给网络的研究和使用带来了巨大的挑战和许多有待解决的复杂问题。首先是物理链路速度飞快增长,根据研究机构Dataquest调查[1],到2004年,14%核心路由器间的链路速度将达到OC-768(40 Gbps),21%的边缘路由器间链路速度将达到OC-192(10 Gbps)。要适应这种飞速增长,就要求作为连接链路结点的路由器性能的提高,以期在单位时间内能够处理更多数目的报文。

另一方面,传统的Internet只是提供尽力传输的转发机制,按照先到先服务的方式根据报文的目的IP地址决定下一跳。但是随着各种网络应用的发展,未来的网络必须为用户提供更多的服务类型和更好的服务质量。这要求路由器提供更多额外的处理机制来实现区分服务,如防火墙访问控制、虚拟专用网、基于策略的路由、QoS调度、网络入侵监测监控、网络地址转换、拥塞控制、流量记账、资源预留、负载平衡,以及基于内容的转发等。这些都是以报文分类为基础的,报文分类通常对报文中的多个字段进行处理,相对于报文转发需要更多的操作。而且这些不同的应用,其分类,它们的分类器中规则数目的多少和分布特征也不同,而且对于分类器的速度、实时性、分类维度、适用规则数目的多少、软件或是硬件实现方式等的要求都不同。

因此综合上面两个方面,就要求报文分类算法能够在分类复杂度增加的同时在单位时间内处理更多的报文才能做到线速转发。而从现有的研究来看,多维报文分类算法仍然存在一定不足,因此对高性能的报文分类算法的研究很有必要。

2 问题描述

IP报文分类技术为在路由器等设备中把到达的报文归类为不同的数据流提供了支持机制:它用不同的规则来标识各个数据流,每条规则根据对报文头部各字段的分析指出该数据流中的报文应当执行的相应操作,如拒绝某指定源地址发来的所有报文。通常一个报文可能与多条规则匹配,而所有匹配的规则中代价最小,或者也可称为优先级最高的那条匹配规则即为该报文的分类结果。为了描述方便,本文假设规则库中对则数目为N,最大字段宽度为W,分类的字段数目为d。

通常情况下,分类器数据库中的规则使用源和目的IP地址来确定网络路径,而使用传输层字段如源和目的端口号以及协议类型表示网络应用程序特征。目前比较常见的分类组合是:源目IP地址的二维分类和源目IP地址、协议类型、源目端口号的五维分类。因此分类器中的每条规则由d部分

*)国家重点基础研究发展计划(973计划),新一代互联网路由与交换理论(No. 2003CB314802)资助课题。谭明锋 博士生。

组成，$R[i]$表示规则 R 的第 i 部分，它表示的是该条规则对报文头第 i 个字段的约束条件。当满足下列条件时我们说一个报文 P 与一条规则 R 匹配：1. 对于任意的 i，P 的第 i 个字段满足 $R[i]$所表示的规则；2. 当有 1 条或多条规则满足条件 1 时，其中优先级最高的即为与该报文匹配的规则。与一条规则匹配的所有报文将被执行同样的操作，用标识符 classID 来唯一表示。

IP 报文分类最简单的一种方法是为到达的报文顺序匹配每条规则。显然，这种方法具有最好的空间性能而时间性能最差。而 d 维 IP 报文分类问题可以等价为 d 维欧氏空间中点的定位问题：分类器中的每条规则对应 d 维空间中的一个或者多个超立方体（当某个域中是用非连续前缀表示时，一条规则代表了空间中的多个超立方体）；而每个报文头代表了该 d 维空间中的一个坐标点，分类算法就是要得到包含了该坐标点的所有超立方体中代价最小或优先级最高的那一个。

如果能够为空间中的每个点预先计算包含该点的所有规则，或者直接计算包含该点的规则中代价最小的规则，那么这条最小代价规则所代表的类别就是该点所代表的报文头所对应的类别，而在对报文进行分类时直接使用报文头二进制串作为地址去访问相应的类别。理论上这种算法进行分类时只需要一次访存，分类速度最快。但是，多维空间中的点的数目是非常巨大的，例如对于源 IP、目的 IP、协议类型、源端口号、目的端口号这样的一个五维分类问题，所对应的 5 维空间中点的数目为 $2^{32+32+8+16+16}=2^{104}$，因此直接计算每个坐标点对应的类别所需的存储空间和时间都是不可接受的。

因此通常 IP 报文分类算法选择的是上述两种极端方法的折衷，但是由于 IP 分类算法的“多域”特征导致了其时空复杂性较高。以二维规则分类器为例，一条 2 维规则可表示为二维欧氏空间中的一个矩形，图 1 显示了最坏情况下一个二维分类器中规则相互重叠的情况。由于这种重叠，使得问题的复杂性大大增加，要兼顾获得较好的时空性能具有相当的难度。

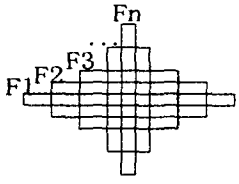

图 1　最坏情况下的分类器结构

我们从 IP 报文分类应用的实际出发可知道一个理想的 IP 路由查表算法要能够满足下列要求，这也是全面的评价一个报文分类算法好坏的标准：

1. 高速分类：由于物理链路速度不断提高，分类算法要能在单位时间内能够处理更多报文。
2. 内存需求小：算法使用的内存小，则可以使用更快的存储器来存储数据结构，从而得到更好的访存性能，降低访存延迟对性能带来的影响。
3. 更新时间短：对于网络地址转换以及防火墙访问控制等规则更新较频繁的应用，IP 报文分类算法必须能够及时处理这些更新。
4. 实现的灵活性强：一个好的 IP 报文分类算法应该能够根据速度以及价格的不同需求以软件或硬件方式实现。
5. 能够处理大容量规则集合：随着未来 Internet 提供更多的区分服务，分类器规则集合将不断增大。好的算法应能够处理大容量的规则集合，性能对规则数目不敏感。
6. 对分类维度可扩展性：理想的分类算法能够对任意数目的分类域进行匹配。对于所设计的分类算法应该具有相当的通用性，不应限定具体的分类域的数目和特定种类。
7. 规则的强表示能力：能使用各种方式描述规则，包括连续前缀表示、非连续前缀表示、通配符表示、范围表示，以及上述各种方式的组合表示。

3　IP 报文分类算法介绍与比较

我们将分类算法分为基于 Trie 树的算法、基于空间分割的算法、启发式算法、基于硬件的算法以及其他算法进行介绍。为了具体说明各类算法的查找过程，我们以一个包含有 8 条二维规则的规则集为例。

表 1　示例规则集

规则	R0	R1	R2	R3	R4	R5	R6	R7
域 1	00*	00*	0**	0**	10*	11*	1**	***
域 2	00*	01*	10*	***	1*	10*	0**	11*

3.1　基于 Trie 树的算法

基于 Trie 树的算法是建立层次式的 Radix Trie[2] 结构，依次在各个维度上进行查找，当在一个维度上得到匹配结点，便开始在与该结点连接的，对应另一个维度的独立 Trie 树上继续搜索。这类算法在最坏情况下需要的访存次数为用于分类字段的总位数，因此通常需要较多访存。此类算法包括 Hierarchical Tries，Set_pruning Trie[3]，Grid-of-Trie[4] 和 EGT-PC[5] 等。

3.1.1　Hierarchical Tries　Hierarchical Tries 对一维空间上的 Radix Trie 数据结构进行简单扩展，采用递归的方法构造而成。首先以规则集合中所有规则的第 1 个字段的前缀集合为基础构造一个一维 Trie 树 F1-Trie。对于 F1-Trie 中每一个表示某一前缀的结点 p，递归地构造一个 $(d-1)$ 维的 Hierarchical Tries 树 T_p 结点 p 和 T_p 之间用 next-pointer 指针连接。因此 Hierarchical Tries 的空间复杂度是线性的，但是在进行查询匹配的时候，如果在 T_p 上没有后续维空间上匹配的规则，则需要在上一维 Trie 树中进行回溯继续查找，所以它在最坏情况下的搜索时间复杂度较差。为了解决这个问题，提出了 set_pruning Trie。

3.1.2　Set_pruning Trie　该算法通过拷贝规则来避免 Hierarchical Tries 中的回溯。报文在 F1-Trie 中匹配某个节点 p 后，沿着它的 next-pointer 向下查询，所有在该维上属于此最长匹配的前缀的规则都会在后续 Trie 树的相应结点上复制，以保证每一个可能的匹配规则在这次遍历中可以遇到。显然，这样避免了回溯，但耗费空间大大增大。因此 Grid-of-Trie 引入了 switch pointer 以兼顾时空开销。

3.1.3　Grid-of-Trie　Grid-of-Tries 是在 Hierarchical Tries 和 Set-pruning Tries 的基础上加以改进提出的一种较为高效的适用于 2 维的报文分类算法。它以一个两层 Hierarchical Tries 树为基本的数据结构，对于收到的报文 (h_1, h_2)，在第一层 Trie 树中查找匹配 h_1 的最长前缀结点，然后跟踪它的 next-pointer，查找 F2-Tries 树中与 h_2 的最佳匹配。为了减少空间存储要求以使一条规则在其中只占据一个结点并且无需回溯查找，Grid-of-tires 算法利用了预计算并在某些结点上保存一个交换指针（switchpointer）使得查询时间为O

(W)。下面我们结合具体的查询实例来进一步说明算法原理及 switch pointer 的使用。

图 2 的左图部分显示了根据表 1 中的各条规则建立起二层的 Trie 树结构,并通过预计算在各个第二层 Trie 树的结点之间添加了 switch pointer。图 2 右图为报文(00 * , 10 *)到达时所经过的第一层 Trie 树 T 上的路径及两个第二层 Trie 树 $T(P)$和 $T(P')$,分别为 T 中两个表示前缀 P 和 P'(00 * 和 0 * *)的结点的 next pointer 指向。$T(P)$和 $T(P')$的结点之间存在一个标记为"1"的 switch pointer,它们满足下面的条件:P'是 P 的父前缀,$T(P)$中的某个结点 u(图例中为根结点)没有标记为 1 的子结点,则 $T(P)$中没有表示前缀$(u,1)$的结点,而 $T(P')$中存在表示前缀$(u,1)$的结点 v,这时,u 就可以用一条标记为"1"的 switch pointer 来指向 v。报文(00 * , 10 *)到达 u 时查找标记为 1 的路径失败,便可以沿着根结点的 switch pinter 跳到 v 继续查找而无需在第一层 Trie 树 T 中回溯。

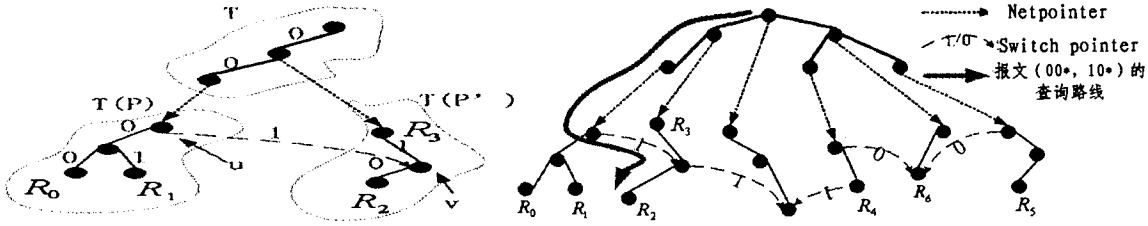

图 2 switch pointer 添加方法(左)和 Grid-of-Trie 数据结构以及查询实例(右)

switch pointer 的引入消除了单纯的层次 Trie 树中存在的回溯需求,只需要经历一条路径就能够找到最佳匹配规则,并且不用复制规则,在时间和空间上都有较好的性能。但由于其初始数据结构的建立需要大量的预计算,在插入或删除规则时都有可能需要建立或删除 next-pointer 和 switch pointer 以及 F2-Tries,不利于规则的快速更新。

Grid-of-Trie 适用于二维规则的报文分类,如目的地址-源地址对,并不具备处理分类维数扩展方面的灵活性。不过在处理 VPN 及多播转发的目的地址-源地址等二维分类问题时,Grid-of-Trie 不失为一个较好的分类算法。

3.1.4 EGT-PC 最近由 Florin Baboescu 等提出的 EGT-PC 算法基于对现实分类器进行观察所得的新发现:对于给定的一个报文,只考虑目的-源地址字段,和它匹配的规则数一般都为 5 个或更少。对优秀的 2 维分类算法 Grid-of-Trie 进行扩展,结合对最后匹配的几条规则的简单线性查找,产生一种新的多维分类算法。这就意味着任何有效的二维分类机制都可以用一个小规模的线性查找来进行扩展。

3.2 基于空间分割的算法

基于空间分割的算法通常是将整个搜索空间递归的按照某种规则切割成若子空间,然后预计算这些子空间与规则之间的关系,根据这些信息构建决策树以进行搜索。此类算法主要有 AQT[1],FIS tree[6],HiCuts[7] 和 HyperCuts[8] 算法等。

3.2.1 Area-based quadtree(AQT) 这种算法是递归空间分解思想在二维空间上的一个实例。具体的分解过程可以用 Quadtree 来表示,Quadtree 是一个用于表示对空间进行二元分解的 4 分支树。树中的每个节点 v 代表分解过程中的一个方形区域,它的四个子节点对应于将结点 v 所表示的二维空间划分成的大小相等的四个象限空间。如图 3 左图中的正方形被划分为的四个子正方形,分别用图 3 右图中的四个子结点 $v1,v2,v3,v4$ 来表示,如果一条规则至少跨越了了该空间的一个维范围,则将该规则分配给该结点,已经分配过的规则不再参与分配。对每个结点分别进行再分解,直到每个象限中只含有一条规则。如图 3 对于到达的报文(00 * ,10 *)根据每一维的第一位进入 01 子象限结点,该结点不存在分支节点,即可得到 R2 为最佳匹配规则。

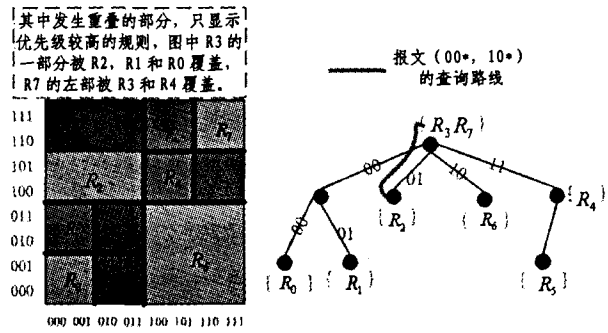

图 3 规则在二维空间的分布图以及 AQT 数据结构及查询实例

AQT 提供一种非常有效的更新机制,对于 N 条规则的二维分类器,只需要$(\alpha\sqrt[8]{N})$的更新时间,α 是一个可调的整数参数。而且可以看出,AQT 树的最大高度等于参与分类的字段的宽度,比起 grid-of-Trie 对每个字段都要建立一层 Trie 树的层次结构,前者的查询速度要高于后者。

3.2.2 Fat Inverted Segment tree(FIS-tree) FIS-tree 数据结构是根据规则在一个坐标轴上的投影所形成的分段集合建立的。其叶结点表示各个分段的端点将坐标轴划分成的各个基本间隔,如图 4 中的[000,001]、[010,011]等。FIS-tree 的任一内部结点表示其子结点表示范围的合集,并且每个结点 v 都保存有指向父结点 parent(v)的指针。每个结点 v 包含一个规则的规范集合(canonical set)S,设该结点表示的范围为 $I(v)$,若存在规则 r,使得 $I(v)\subseteq r$ 但 $I(\text{parent}(v))\not\subseteq r$,则 $r\in S$。具体在解决二维分类问题时,首先根据所有规则在 x 轴上的投影分段集合建立 FIS-tree(x-FIS tree),再根据每个结点所存储的各个规则规范集合在 y 轴上投影的分段集合建立相应的 FIS-tree(y-FIS tree)。

查询过程分两个阶段,如图 4 所示。首先从 x-FIS tree 的根节点出发到达表示包含报文的基本间隔的叶结点,然后沿指向父结点的指针倒退查询,在每个结点对其相应的 y-FIS tree 进行查询,找到与报文匹配的代价最小的规则。

FIS-tree 能够快速更新规则,并且可以通过调整 FIS-tree 的高度等参数来满足不同的时空需求。如前所述,基于 Trie 树的数据结构对于一维上的查询问题比较有效,一旦扩展到多层次的 Trie 树结构,数据结构的复杂度及查询时间会急剧

膨胀，上述的几种算法采用了预计算引入额外指针信息(switch point)(grid-of-Trie)或通过对二维空间进行分解的思想(AQT 和 FIS-tree)对维度引起的复杂性进行简化，使得在低维报文分类问题(典型的为前缀表示的目的地址-源地址报文分类)中可以使用这种数据结构。

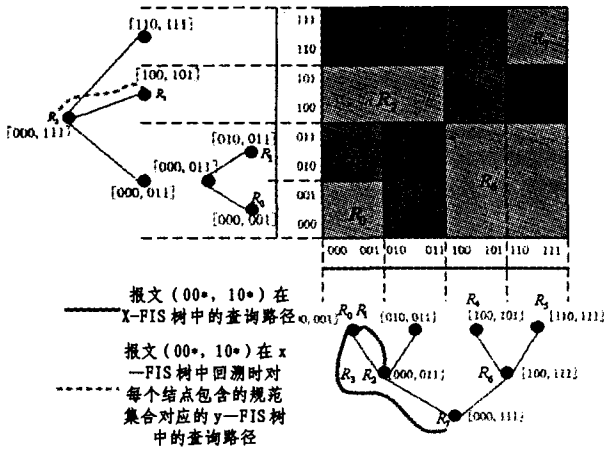

图 4　FIS-tree 数据结构查询实例

上面介绍的这些较为高效的二维查找算法中，AQT 和 Grid-of-Trie 在建立数据结构时每条规则只进行了一次存储，而由于结点对应的规则规范集合相交的可能性较大，FIS-tree 通常会对规则重复存储，从而对空间的利用相对较差。查询效率方面，Grid-of-Trie 在内部结点处判断分支走向时只需进行一次一维查询，AQT 则需要两次一维查询，但由于后者采用四叉树的结构，使得树的高度降低，因此二者在查询时间方面针对不同的数据库可能会有不同的优劣比较；而 FIS-tree 在判断分支走向时，需要将报文字段取值依次与结点所保存的字结点范围端点进行比较，增加了访存次数，并且相对于前缀表示的字段，更多的支持字段的范围表示。更新时间方面，grid-of-Trie 的更新操作过于复杂，适用于相对静态的分类器，而 AQT 和 FIS tree 则提供了高速有效的更新机制。由于只适用于低维报文分类，Grid-of-Trie，AQT 和 FIS tree 的使用受到很多限制，无法在某些需要对报文进行高维分类处理的情形下发挥起高效的优势。

3.2.3　HiCuts　HiCuts 算法首先将整个空间看成一个 d 维的超立方体，然后递归地使用启发式规则选择其中一个维度，将该维度等分为 n 份，得到 n 个子超立方体，然后在 n 个子超立方体上重复该过程，直到子超立方体中规则的数目少于特定值 binth。

图 5 是根据表 1 所示的分类器建立的 HiCuts 数据结构。HiCuts 算法使用了空间分割的思想，在分割过程中可以根据参数平衡权衡在查询时间和内存耗费。但对于多维分类问题，当规则数目较大时，需要建立的决策树将非常庞大，内存消耗及查询时间也会急剧膨胀。

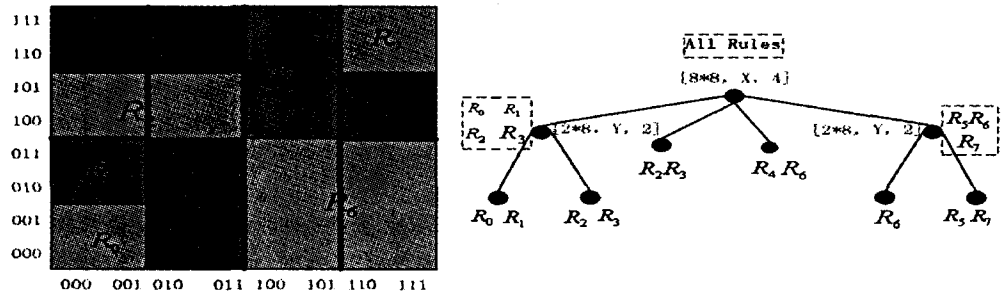

图 5　HiCuts 树结构实例，其中三元组表示：[空间大小，要分割的维，分割的份数]

3.2.4　HyperCuts　类似于 HiCuts 建立决策树，HyperCut 也通过分割搜索空间并在每个结点上使用启发式规则，用本地最优化决策建立决策树。所不同的是，HiperCuts 每次在多个维上进行分割。下面是根据分类器实例建立 HyperCuts 决策树的例子。

可以看出 HyperCuts 只用一步就可到达包含指定数目规则的各个叶结点，其决策树的高度要小于 HiCuts 的决策树。因此在某些情况下，HyperCuts 的空间和时间性能都要好于 HiCuts 算法。HyperCuts 在使用硬件实现时能够完全流水化，允许快速的更新。

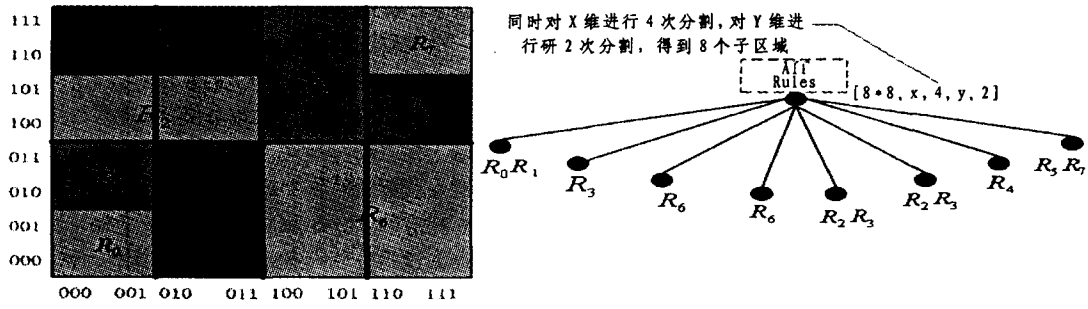

图 6　HyperCuts 决策树实例

3.3　启发式算法

实际上前述的 HiCuts 算法和 HyperCuts 算法以及本节要介绍的 RFC[9] 算法也都属于启发式算法。本节主要介绍 RFC 算法。

RFC 算法的主要思想是将 IP 分类问题看成一个从 S 位报文头向 T 位的 classID 的多步映射过程，其中 $T=\log_2 N$ 且 T 远小于 S，其中 N 是则的总数。算法对报文的处理过程为：1、将报文头部各字段被划分为多个部分；2、用这些字段作为

索引并行访存，获得预先计算得到的匹配信息 eqID，其中 eqID 长度小于索引长度；3、将得到的数个 eqID 组合，得到新的索引；4、重复 2 和 3，直到获得最后的规则索引号 ClassID。

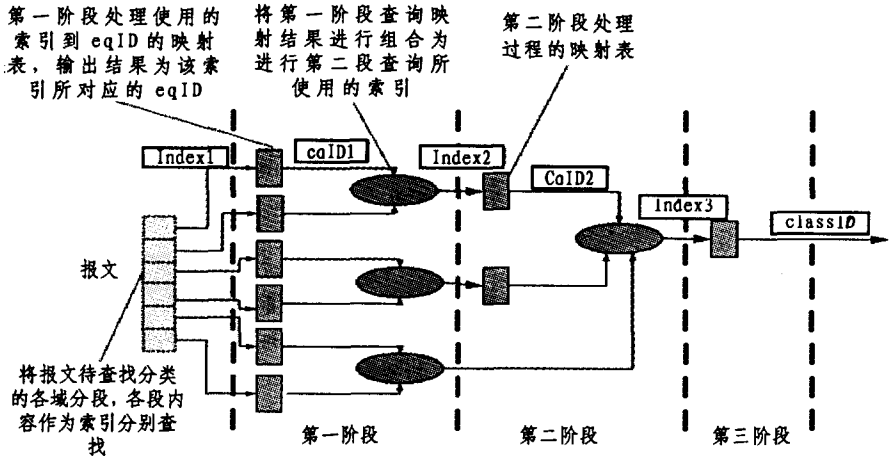

图 7　RFC 中的搜索过程

RFC 算法通过将最具相关性的部分组合在一起以及在内存消耗允许的情况下，尽可能多地进行合并获得更好的搜索性能。不过为了构建由索引获得 eqID 的映射表，需要为每次更新进行大量的预处理工作，这些工作是由软件完成的，需要巨大的内存和预处理时间，故只适用于规则更新不频繁的分类器。RFC 的查询可由硬件或软件实现，但由软件实现时其性能较低。RFC 在查询时间上优于 HiCuts，但对存储空间要求非常大。

3.4　基于硬件实现的方法

由于查找算法的低效始终是高性能路由器的瓶颈，因此一些算法利用硬件的并行性来加速处理。除了上述如 RFC 算法等一些算法能够通过 ASIC 设计用硬件实现外，目前较多的是使用 TCAM 来分类。所有的规则以优先级降序存储在 TCAM 存储器数组中，报文到达时会并行的与所有存储单元进行比较并且选择存储地址最小的一个作为最终匹配的规则。

TCAM 由于它的简单和高速很快得到广泛应用，但也存在一些缺点：1、所有规则必须采用或者转换为前缀方式表示，因此一跳原始规则可能会生成数十条甚至更多新的规则，同时需要一定的预处理时间；2、由于需要按照优先级顺序存储，因此插入或删除规则时需要复杂的操作。3、存储密度小，单位 Bit 和总体价格昂贵；4、由于需要并行对所有单元进行比较，功耗很大。

3.5　其他算法

3.5.1　Tuple Space Search　二元组空间查询算法（Tuple Space Search）[10]，其平均查询时间短，更新速度快，但每次查询和更新的时间不确定。对于高维规则分类该算法需要较多的访存次数。

基本的元组查询算法是将一次分类查询分解为多次精确的匹配查询。该算法对规则数据库进行分析，发现尽管其中的规则包含了很多不同的前缀和范围，然而不同字段前缀长度的数目通常很少，因此这些长度组合起来的数目也不会很多。

算法首先将每个 d 维的规则映射到一个 d 元组上，元组的第 i 部分存储的是规则的第 i 维的前缀长度（对于范围表示的字段，它的长度定义为嵌套层次）。然后该算法从规则中提取一定的比特位作为 hash 关键字，并将每个元组所对应的规则存储到该元组对应的 hash 表中，这样，当报文到达时，可以线性地对每个元组对应的 hash 表进行查找（hash 查找通常只需要一次访存操作），由于元组的数目通常大大小于分类器中规则数目，因此相对于原始的线性查找，这种元组空间查询方法的性能要好得多，并且增加规则的更新操作只需要一次 hash 访存。具体的各元组对应的 hash 表构造示例如表 2。

表 2　Tuple Space Search 算法主要数据结构示例

规则->元组表		元组->规则表	
规则	元组	Hash 表项中的元组	Hash 表项对应元组的规则集合
R_0(00*,00*)	(2,2)	(0,1)	{R_5}
R_1(00*,01*)	(2,2)	(1,1)	{R_6}
R_2(0**,10*)	(1,2)	(1,2)	{R_2}
R_3(0***,****)	(1,0)	(2,1)	{R_4}
R_4(10*,1**)	(2,1)	(2,2)	{R_0,R_1}
R_5(11*,10*)	(0,1)	(1,0)	{R_3}
R_6(1**,0**)	(1,1)	(0,2)	{R_7}
R_7(***,11*)	(0,2)		

总之，该算法在元组数目很小的时候性能良好。然而，hash 表的使用使得查询和更新耗费时间不可估计，因此并不能保证在任意的规则数据库上都高效运行。而且元组的数目随着不同的规则集合而不同，也可能非常巨大，最坏情况可能是 $O(W^d)$。

3.5.2　ABV 算法　ABV[11] 算法由于采用并行查找集中地进行访存操作，使得查询时间大大减少。然而最坏情况下的空间复杂度非常高，并且由于硬件方面的限制使得可实现的规则数目也受到限制。

这种位向量方法将 d 维报文分类问题划分为 d 个子问题，然后再将结果相交。首先为规则中的每一个字段建立一个单维的 Trie 树。Trie 树中每个相对应于一个有效前缀的结点用一个 N 比特的向量来表示（N 为规则的数目），某个结点表示的前缀和哪几条规则匹配，则该向量的相应比特位置 1。对于到达的报文，在各维 Trie 树上进行搜索后得到最长匹配前缀结点，其对应的向量即为各个字段的结果向量，然后将所得向量做交集得到代价最小的匹配规则位。由于在实际中，各维匹配得到的向量一般都是稀疏向量，可以采用聚合过程，很快的排除掉没有匹配到的规则对应的比特位，从而减少

访存次数。

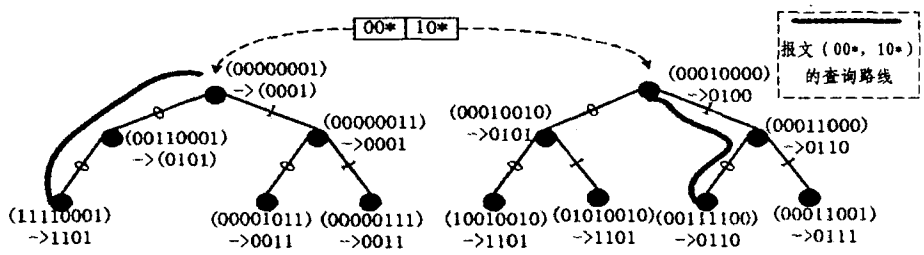

图 8　ABV 结构示例

ABV 可以使用用软件实现，也可以使用硬件的宽总线来实现。但是显然由于位图操作的限制使规则数目也受到限制。不过上面介绍过的各算法在维度增加时，数据结构及查询的复杂性可能成指数级增长，而 ABV 算法由于将多维分类问题分解为多个一维查找，因此在高维分类处理中仍然能够保持较高的效率。ABV 算法需要大量的预处理，其更新代价非常大，因为每一次的插入都可能要改变大量与相关结点相联系的比特向量。

3.6　IP 报文分类算法简要比较

以上介绍各典型算法时对各算法之间的性能优劣及改进关系已经进行了较为具体的分析，表 3 综合给出了主要 IP 报文分类算法最坏情况下的时空复杂度。

表 3　报文分类算法时空复杂度比较

算法	最坏情况时间复杂度	最坏情况时间复杂度
线性搜索	$O(N)$	$O(N)$
Hierarchical tries	W^d	NdW
Set-pruning tring	dW	N^d
Grid-of-tries	W^{d-1}	NdW
FIS-tree	$(l+1)W$	$L\times N^{1+1/2}$
Area-based quadtree(AQT)	$O(\alpha W)$	$O(NW)$
RFC	d	N^d
HiCuts	d	N^d
ABV	$O(dN)$	$O(dN^2)$

如前所述，现在对于各类算法的比较仅限于空间和时间性能的理论复杂度分析以及在少数几类规则数据库上的模拟测试，并没有系统的全面考虑规则数据库、算法及报文流这几类因素来分析比较每个因素的变化对于各方面性能的影响。

4　IP 报文分类问题研究方法分析

从对现有各较为实用的高性能算法分析结果来看，除了最新提出的 HyperCuts 在空间时间性能方面都表现出色并具有灵活的可扩展性、可硬件流水实现及快速更新等优势外，其他的算法在时间、空间性能、处理维度扩展或更新效率方面的优劣都各有侧重。将规则首先转化为前缀，基于多维前缀匹配的算法难以充分利用多维报文匹配的特点及规则库在结构和冗余方面的特征，在多维报文分类算法理论复杂度下限的限制下，无法进一步扩展或提高性能。而由于报文中字段特点的多样性，使得单一的处理思路发展也受到局限。结合各种思想的处理优势，提出综合性的算法是报文分类算法发展的一个重要趋势。我们在本节对现在主要的高性能报文分类算法的设计思路和研究方法进行小结，并简要分析其发展趋势。

4.1　多维空间分解

W 比特的前缀可以看作数轴上 0 到 2^W 上的一个地址范围。如果前缀对应于一条线段，则二维规则对应于一个矩形，三维规则对应于立方体，等等。从几何的观点看报文分类问题便是要找到包含给定点的最低代价的超立方体。使用递归空间分解的报文分类算法的主要思路就是通过递归地对搜索进行分割，逐渐缩小子空间包含的规则数目，以期达到包含少量规则的叶结点。AQT，FIS tree，HiCuts 和 HyperCuts 等都是根据递归分解空间的思想发展而来的，一般使用树结构对算法实现。由于此类算法可以一次对多个维度进行分割和计算，因而在搜索时存在潜在的并行性，通常可获得较好的性能。

总结几种使用递归分解空间思想的算法特点，可以发现对空间进行分割的方法由于具有较强的适应性（可以根据规则库的启发式特点决定选择分割的维度和分割的次数）、灵活性（将多维问题分解在规则投影的单个维上进行考虑和处理，可以提供高效的更新机制）以及可进一步改进（寻找减少分割冗余的分解空间方法）的特点，还有较大的发展空间，是报文分类算法研究思路的主流趋势之一。

4.2　对快速二维报文分类算法进行扩展

在 d 维的包含 N 个互不重叠的矩形区域的空间中定位点的算法复杂度的下限为时间复杂度 $O(\log_2 N)$ 和空间复杂度 $O(N^d)$，或 $O(\log_2 N^{d-1})$ 的时间复杂度和 $O(N)$ 的空间复杂度[6]。由于报文分类中的规则区域允许相互重叠，因此该下限也是报文分类算法的复杂度下限。而对于一些只应用于两个域规则的分类算法，上述的最低限度便不起作用了。有的算法可以达到对数时间和线性存储量[1,4,6,12]。但是这些算法只适用于一些特殊问题（只涉及源目的地址），并不能够解决一般性的 d 维报文分类问题。对现实规则数据库研究后发现，对于给定的一个报文，只考虑源目地址域，和它匹配的规则数一般都为 5 个甚至更少。即使对规模巨大的分类器来说，如果能够找到所有的源-目的地址对与报文匹配的规则，剩下的工作也只剩下最多需要对 20 个规则进行线性查找的任务了。这就意味着任何有效的两维分类机制都可以用一个小规模的线性查找来进行扩展，EGT-PC 为这种思想的代表算法。

4.3　结合规则数据库的结构和特点

由于规则库中规则之间的相互重叠，使得报文分类问题的复杂性大大增加，然而通过对实际分类器的结构特点研究，其规则间发生重叠的数目远远小于理论的上限，并且许多源目 IP 地址对具有相同的传输层字段[13]。因此利用规则数据库的这些特点进行算法设计，能够实现特定要求下的高效分类算法。并且在算法数据结构建立过程中通过对规则库特点的观察利用启发式进一步进行处理，这也是设计高效分类算法的有效途径之一。

（下转第 19 页）

首先找到由本地路由器生成的路由器 LSA 或网络 LSA。如果本地路由器是网络指定路由器，检查在这些 LSA 中每个条目所描述的对象，如果这个对象不在树上，则生成一条新的路由到这个对象的路由条目，否则就将其与之进行比较，保留代价较小的除去旧的代价更大的路由条目，并将以前树中的那一项删除掉。同时，如果对象描述的是一个路由器，就将其放入一个队列中。处理完本地路由器生成的域内 LSA 的所有条目后，算法再从队列中取出一个路由器，同样也在本地的数据库中找到它生成的域内 LSA，重新上面的步骤。在处理域内路由器及其生成的域内 LSA 后，此时就得到了全部的域内路由表。

然后依次计算本地路由器数据库中的网络汇总 LSA、ASBR 汇总 LSA 和 AS 外部 LSA 得到整个的路由表。

4 OSPFV3 路由协议软件及实验环境

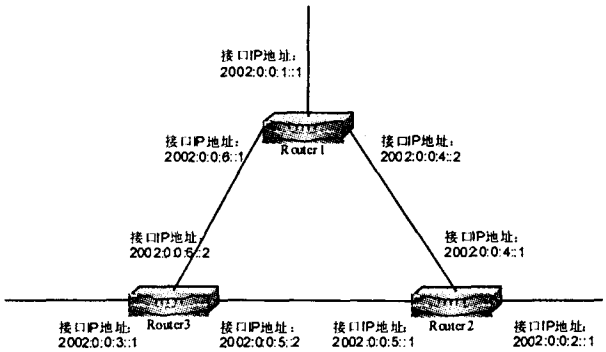

图 6 OSPFv3 路由协议软件实验环境拓扑图

基于 RFC2740 和 RFC2328 编写的 OSPFv3 路由协议软件运行于 FreeBSD 操作系统之上，所使用的验证系统实验平台拓扑图如图 6 所示。其中，实验环境的路由器系统由三台 PC 机组成，它们的硬件环境和软件环境分别如表 1 所示，各路由器之间以及路由器与终端主机之间通过 100M 以太网连接。

表 1 实验环境路由器软硬件配置

路由器编号	CPU	内存	操作系统
Router1	Pentium4 2.4G	512M DDR	FreeBSD 4.7-RELEASE
Router2	Pentium3 800M	512M SDRAM	FreeBSD 4.7-RELEASE
Router3	Pentium4 1.4G	512M SDRAM	FreeBSD 4.7-RELEASE

经过多次实验，当路由软件启动后，可以有效地建立路由器内部的路由转发规则。当一条转发通路上的某条链路不可用时，平均经过 47.1 秒的时间，路由器可以重建路由表，选择通过其它路由器转发数据分组，实现了路由选择协议的主要功能。

小结 随着 IPv6 协议日益广泛的应用，路由器作为网络互连的核心设备，它对于 IPv6 的支持程度成为人们所关心的问题。本文在对比分析 OSPFv3 路由协议的基础上，给出了一种 OSPFv3 路由协议软件的实现方案，并且通过实验验证取得了良好的效果，这对于进一步研发具有自主知识产权的 IPv6 路由器有着重要的意义。

参 考 文 献

1 Moy J T. OSPF Version 2. Request for Comments 2328, April 1998
2 Coltun R. The OSPF Opaque LSA Option. RFC2370, July 1998
3 Coltun R, Ferguson D, Moy J. OSPF for IPv6. Request for Comments 2740, Dec. 1999
4 Shaikh A, Goyal M, Greenberg A, Rajan R, RamakrishnanK K. An OPSF Topology Server: Design and Evaluation. IEEE J. Selected Areas in Communications, 2002,20(4)

(上接第 10 页)

4.4 其他方法

通过在算法更新或是重建算法数据结构的预处理过程中进行预计算和添加相应指针，可以在一定程度上避免报文查找过程中的回溯；算法的数据结构建立后，通过结点合并、路径压缩、提取公共子集消除冗余措施等预处理，可以对算法进一步改善，也有助于提高算法处理的空间和时间效率。但是这种方法通常会使更新复杂化，因此较适用于规则更新不频繁的环境中，或是用于优化原本就已需要在更新时重建整个数据结构的算法。

总结 Internet 的带宽和速度的增长要求网络单元要能在单位时间内能够处理更多数目的报文。而随着各种网络应用的发展，未来的网络必须为用户提供更多的服务类型和更好的服务质量，因此高性能 IP 报文分类算法的研究十分必要。最近几年以来，各种不同的 IP 路报文分类算法被提出，本文对 IP 报文分类算法进行了概述和总结，并在分类速度、更新速度、存储耗费、适用范围、对分类维度和规则数目的可扩展性、规则表示的灵活性等方面进行比较，并对 IP 报文分类算法研究的方法和趋势进行了分析和总结。IP 报文分类算法是众多网络上层服务和功能的基石，对它的回顾和总结将帮助我们加深 IP 报文分类问题的理解，也有助于对此问题的进一步研究。

参 考 文 献

1 Buddhikot M M, Suri S, Waldvogel M. Space decomposition techniques for fast layer-4 switching. In: Proc. of Conf. on Protocols for High Speed Networks, August 1999. 25～41
2 Knuth D E. The art of computer programming. vol3: sorting and searching, Addison-wesley, 3rd editon. 1998
3 Tsuchiya P. A search algorithm for table enTries with non-contiguous wildcarding: [unpublished report]. Bellcore
4 Srinivasan V, et al. Fast and scalable layer 4 switching. In: Proc. of ACM Sigcomm'98, september 1998
5 Baboescu F, Singh S, Varghese G. Packet Classification for Core Routers: Is there an alternative to CAMs? in INFOCOM, 2003
6 Feldman A, Muthukrishnan S. Tradeoffs for packet classification, In: Proc. of Infocom, March 2000,3:1193～202
7 Gupta P, McKeown N. Packet Classification using Hierarchical Intelligent Cuttings. In: Proc. Hot Interconnects VII, August 99, Stanford. This paper is also available in IEEE Micro, January/ February 2000,20(1):34～41
8 Singh S, Baboescu F, Varghese G, Wang J. Packet Classification Using Multidimensional Cutting. In: Proc. of ACM SIGCOMM, Karlsruhe, Germany, August 2003
9 Gupta P, McKeown N. Packet Classification on Multiple Fields. In: Proc. Sigcomm, Computer Communication Review, Sept. 1999,29(4):147～60
10 Srinivasan V, Suri S, Varghese G. Packet classification using tuple space search. The ACM Sigcomm'99, 1999
11 Baboescu F, Varghese G. Scalable packet classification in Proc of ACM Sigcomm'01, september 2001
12 Lakshman T V, Stidialis D. High speed policy-based packet forwarding using efficient multi-dimensional range matching. In: Proc. of ACM Sigcomm '98, sept. 1998
13 Kounavis M E, Kumar A, Vin H, Yavatkar R, Campbell A T. Directions in Packet Classification for Network Processors. Second Workshop on Network Processors (NP2), Anaheim, California, February, 2003. 8～9