一种适用于多维的快速IP分类算法*

喻中超，徐 恪，吴建平

(清华大学 计算机科学与技术系，北京 100084)

E-mail: yzc@csnet1.cs.tsinghua.edu.cn

http://netlab.cs.tsinghua.edu.cn

摘要：网络应用的发展要求路由器必须有能力支持防火墙、提供QoS、流量计费等一系列功能，这些功能都要求路由器对IP包进行分类以完成对数据包的不同处理.在Grid of Tries算法的基础上，提出了一种新的IP分类算法.该算法不仅克服了Grid of Tries算法在多维IP分类方面的局限性，而且在时间和空间性能上都优于Grid of Tries，是目前综合性能比较好的分类算法.

关键词：IP分类；路由查找；Trie树；IPSec

中图法分类号：TP393 **文献标识码**：A

从Internet发展趋势来看，未来的IP网络必须为用户提供更多的服务类型和更好的服务质量.这些服务包括区分服务(differentiated qualities of service)[1]、防火墙(firewalls)[2]、基于策略的路由(policy-based routing)[3]、虚拟专用网络(virtual private network)、流量计费(traffic billing)[4]等等.所有这些服务都需要将到达的数据包归类为不同的数据流，是以包分类(packet classification)技术为基础的.由于在Internet上传送的绝大部分都是IP包，因此，相应的技术也称为IP分类(IP classification)技术.

IP分类必须得到路由器的支持.随着各种网络应用的发展，路由器不仅担负着传统的路由转发的任务，还必须具备IP分类的功能，两者都需要高效率的查找算法.光纤通信技术的成功使得通信链路能够以吉比特乃至更高的速度进行数据传输，而路由器正在成为网络的瓶颈.近年来，路由器技术在体系结构和内部的交换结构方面都取得了较大的突破[5].目前路由器的瓶颈主要表现在查找算法上.

近年来还出现了一些新的协议，如MPOA(multi-protocol over ATM)和MPLS(multiprotocol label switching)等，它们将路由器的查找任务转移到网络边缘，以缓解路由器瓶颈的压力.但是，一方面向已基本成型的TCP/IP协议栈中引入新的协议会使得本来已经脆弱的网络体系结构更加脆弱，另一方面，这些协议的引入并没有完全避免对路由数据库的查找，所以，研究更高效的查找算法仍然十分有必要.

本文先为IP分类问题从数学上建模并简单地介绍IP分类算法研究的现状，然后从二维IP分类出发提出了一种适用于多维的高效IP分类查找算法，并和其他算法进行了比较.

---

* **收稿日期**：2000-04-13；**修改日期**：2000-06-27

基金项目：国家自然科学基金资助项目(90104002)；国家863高科技发展计划资助项目(863-306-ZD-07-01)

作者简介：喻中超(1977－)，男，湖北武汉人，硕士，主要研究领域为分组分类，调度算法；徐恪(1974－)，男，江苏洪泽人，博士，讲师，主要研究领域为计算机网络体系结构，计算机系统性能评价；吴建平(1953－)，男，山西太原人，博士，教授，博士生导师，主要研究领域为计算机网络体系结构，计算机网络协议测试，形式化技术.

1 IP 分类数学模型

为了下文叙述方便,我们先定义一些术语.

1.1 术语和符号说明

定义 1. 二值比特只能取 0 和 1 两个值,三值比特可以取 0、1 和 * 这 3 个值,其中 * 为通配符,可以与 0 和 1 中的任意一个进行匹配.

定义 2. 比特串是具有一定长度(称为比特串宽度)的比特序列.比特串可视为一个数值(二值比特串)或者一个数值范围(三值比特串).

定义 3. 对比特串宽度为 W 的二值比特串 D,它的一个前缀 P 是长度介于 0 到 W 之间的一个二值比特串,$length(P)$表示前缀 P 的以比特数为单位的长度.

定义 4. 包头 H 是有 K 个域的实体,包头的各个域分别表示成 $H[1],H[2],\ldots,H[K]$,其中每个域都是二值比特串.

定义 5. 一条过滤规则 F 也具有 K 个域.与过滤规则的一个域 $F[i]$相关联的有一个匹配方式,它可以是精确匹配(exact match)、前缀匹配(prefix match)或范围匹配(range match)这 3 种匹配方式中的任何一个.

定义 6. 过滤规则 F 的域 $F[i]$通过一个数值来指定,如果包头 H 的域 $H[i]$与 $F[i]$满足 $H[i]=F[i]$(数值相等),就称 $H[i]$与 $F[i]$精确匹配.

定义 7. 过滤规则 F 的域 $F[i]$通过一个前缀来指定,如果包头 H 的域 $H[i]$与 $F[i]$开始的 $length(F[i])$个二值比特相同,那么就称 $H[i]$与 $F[i]$前缀匹配.

定义 8. 过滤规则 F 的域 $F[i]$通过一个数值范围指定,$F[i]=val1\ldots val2$,如果包头 H 的域 $H[i]$满足 $val1\leqslant H[i]\leqslant val2$,那么就称 $H[i]$与 $F[i]$范围匹配.

定义 9. 称过滤规则 F 与包头 H 匹配,当且仅当对 H 的每个域 $H[i]$都与 F 相应的域 $F[i]$匹配.$H[i]$与 $F[i]$的匹配方式由 $F[i]$的形式决定,可为上述 3 种匹配方式中的任何一种.

定义 10. N 条过滤规则的集合 FS 称为过滤规则数据库.

定义 11. 每条过滤规则 F 具有一个代价属性,记为 $cost(F)$.约定对$\forall F_1,F_2\in FS$,若 $cost(F_1)=cost(F_2)$,则必有 $F_1=F_2$,我们用规则的代价属性来保证 IP 分类查找的唯一性.

1.2 最佳匹配过滤规则和 IP 分类问题

定义 12. 定义以下问题为最佳匹配过滤规则问题:

给定过滤规则数据库 $FS\neq\varnothing$,对包头 H,求最佳匹配的过滤规则 f_{best},满足:

(1) $f_{best}\in FS$;

(2) f_{best}与 H 匹配;

(3) $\forall f\in FS,f\neq f_{best}$,若 f 与 H 匹配,则必有 $cost(f_{best})<cost(f)$.

IP 分类是最佳匹配过滤规则问题的一个实例.原则上讲,共有 7 个域可被选择为过滤规则的域:源/目的网络层地址(各 32 位)、源/目的传输层端口(各 16 位)、TOS(type of service,8 位)、协议域(8 位)和传输层协议标志(8 位),共计 120 位(为叙述方便,称这 7 个域是 IP 包头中的域,尽管有些域实际上是传输层中的域).实际上,过滤规则并不是对所有的域都感兴趣.通过对几个正在运行的 ISP 过滤规则数据库的统计发现,17%的过滤规则指定了 1 个域,23%的过滤规则指定了 3 个域,60%的规则指定了 4 个域[6].

路由查找(也称为 IP 查找)是 IP 分类的一个特例.路由查找是根据到来的 IP 包头查找路由表(一般是最长前缀匹配)得到下一跳(next hop)的地址.如果过滤规则数据库 FS 只有一个域——目的 IP,并且每条过滤规则的代价与前缀长度成反比,则此时的路由查找问题就成为一个域的 IP 分类问题.

通常与 IP 分类的每个不同类别相联系的有一个 ACTION,表示对于属于相同类别的 IP 包的处理动作.本文只关心从包头得到该包所属类别的 ID(下文称为 ClassID)的过程,并不考虑 IP 分类后对包的处理行为.

1.3 性能评价准则

IP 分类问题的核心是高效的查找算法.优秀的查找算法必须满足以下 3 个条件:

(1) 速度快.这是评价 IP 算法的最重要的标准.IP 分类总是以包为单位进行处理的,因此多以单位时间内处理的数据包的数量来衡量.算法的时间复杂性有 3 种评价指标:最坏情况、平均情况和统计情况.平均情况是指在完全随机的情况下,IP 分类的平均时间.统计情况是指在某种包分布和过滤规则匹配率分布的假设下,IP 分类时间的期望值.制约速度主要是指内存访问的次数,因此在 CPU 计算量可接受的范围内,理论分析也常用访存次数来衡量.

(2) 占用内存少.占用的内存不仅包括存储过滤规则数据库本身所占用的内存空间,还包括为实现快速查找所建立的静态的数据结构以及在查找计算过程中动态分配的空间.

(3) 易于更新.共有 3 种可能的更新.① 完全更新:从初始过滤规则数据库建立查找数据结构或者运行过程中的全部数据结构重建;② 增量更新:在运行过程中,向过滤规则数据库中插入或删除一条规则;③ 结构重组:随着过滤规则的不断插入或删除,可能会造成查找数据结构效率变低,需要对数据结构进行适当的调整.

现存的大多数解决方案都比较重视查找的时间和空间效率.对于数据结构的更新,尤其是增量更新,还没有在保证时间和空间效率不受损的前提下的有效解决方案.IP 分类与路由表的更新不同,过滤规则基本上是人工配置的,而且更新的频率很低,所以问题也不如路由表的更新紧迫.本文所提出算法的更新策略的优化尚在进一步研究之中.下面,我们简单介绍目前 IP 分类研究的现状.

2 相关工作

基于 Pattern 的包分类[7].基于 Pattern 的算法最早是在操作系统中用于从输入队列中把数据包向不同进程空间分发.它是目前所知的第一个避免线性查找的包分类方法.它的查找速度仅与域的数目成正比,而与过滤规则的数量无关.但是,该算法对过滤规则有着十分苛刻的限制,不适合用在路由器上担任 IP 分类的任务.

Crossproducting[8].该算法基于缓存策略.通常 IP 包的到达并不具备良好的局部性,因此,缓存包头的效率不是很高.但该算法不是对 IP 包头进行缓存,而是对不同过滤规则的各域进行交叉组合(即 crossproducting),预先计算出不同 crossproduct 对应的 ClassID 放在缓存中.对到来的包头 H,分别根据各域进行查找,将各域的查找结果合并,得到的 crossproduct 所对应的 ClassID 即是 H 所对应的 ClassID.Crosspruducting 实际上是对一类包头的缓存,因此,命中率比缓存包头的情况要好得多.但是,crossproducting 很容易产生空间爆炸的现象,而且其查找的时间具有不确定性.

Modular 算法[9].Modular 算法是一种基于统计的包分类的算法,根据对过滤规则的匹配率的

分布和 IP 包的分布优化查找数据结构.但是,由于目前尚不能建立有效的统计参数,这种算法还难以用到实际的路由器中.

RFC 算法[6]. RFC(recursive flow classification)算法是一种通过将 S 比特包头比特串分步映射到 $T(T \ll S)$位 ClassID 的分类算法.它是已知算法中查找速度最快的,但它需要大量的预计算工作(通常是几十秒),而且很容易发生空间爆炸.

Grid of Tries 算法[8]. 该算法扩展了 Trie 树数据结构,用二级 Trie 树实现目的-源 IP 对的分类问题.它是一种优秀的二维 IP 分类的解决方案,但是由于它对过滤规则有很多限制,因此不适用于多维 IP 分类的情况.

CAM 查找. 这是一种通过硬件并行处理加快查找速度的解决方案.CAM(content addressable memory)在查找过程中,内存每个单元都和输入的键值并行比较,而且可以进行三值比特串的匹配,即 * 可同时与 0 和 1 进行匹配.由于工艺方面的原因,CAM 能够处理的过滤规则宽度十分有限,更重要的是,随着处理器和内存性价比的迅速提高,硬件解决方案将存在过时的风险.

在 Grid of Tries 算法的基础上,本文提出了一种新的算法——无冲突哈希 Trie 树查找算法.该算法的时间和空间平均性能都比 Grid of Tries 多维分类优秀,并且消除了 Grid of Tries 对过滤规则严格的限制.

3 无冲突哈希 Trie 树查找算法

3.1 基本思想

前面说过,可能成为过滤规则域的有 7 个域,但是实际的过滤规则数据库基本上只限于 5 个域:源/目的 IP、源/目的端口和协议域.协议域宽度为 8 比特,为了便于进一步扩展为 6 个域,同时也为了后面方便处理,本文约定,协议域的宽度为 16 比特.

目的端口、源端口和协议域可取的值为 0～65535,但是,实际上过滤规则的取值往往只是 0～65535 中极有限的一部分.目前,协议域实际上能够取的值只能是 TCP,UDP,ICMP,IGMP,(E)IGRP,GRE 和 IPINIP,或者是通配符“ * ”(即与 0～65535 中的任何值匹配).在绝大多数 Client-Server 结构中,所有端口大致可以分为两类[10],一类是 Reserved ports,端口号为 1～1023,另一类是 Ephemeral posts,端口号大于 1023.Ephemeral ports 通常用在客户端,大多是由 kernel 指定的,它除了标识一个连接的端点之外并没有其他意义.过滤规则几乎不可能对单独某个大于 1023 的端口感兴趣,比较常见的是如 gt 1023 的一个范围,表示大于 1023(小于 65535)的所有端口.对 Reserved ports 端口的限定主要集中在 20,21(用于 FTP)和 80(www),其他端口在过滤规则中较少出现.另外,Client-Server 结构本身的特性决定了在绝大部分情况下,通信双方的端口一个为 Reserved port,另一个为 Ephemeral port.

从上面的分析可以看出,实际上过滤规则端口和协议域的取值(或取值范围)在不同情况下的组合数目是非常有限的.基于此,我们建立了一个二阶段的查找表,可以进行无冲突的哈希查找.下面以表 1 为例进行说明.

表 1 是一个具有 6 条过滤规则的数据库.该表假定 ClassID 与每条过滤规则的代价相同.以目的端口为例,我们为 0～65535 中所有可能值对应一个位图,该位图表示一个端口值与哪些过滤规则匹配.例如,端口 21 和 22 的位图都是 100111,表示它们都和过滤规则 0、3、4 和 5 匹配.根据不同的位图,将 0～65535 中所有可能的目的端口值划分为不相交的等价类,等价类 a 的位图记为

$bmp(a)$. 目的端口的等价类总数记为 D. 所有目的端口等价类的集合记为 D_Set. 根据表 1 构造的 D_Set 为{{80},[20,21],{0～65535 中除 20,21,80 之外的其他值}}. 用同样的方法构造源端口等价类的集合 S_Set 和协议号等价类集合 P_Set，其等价类数目分别记为 S 和 P，由上面的分析可知，实际应用中应有 $D,S,P \ll 65536$.

Table 1 An example database of filters

表 1 一个过滤规则数据库示例

ClassID①	Dest-IP②	Src IP③	Dest Port④	Src Port⑤	Protocol⑥
0	10.1.*.*	10.2.*.*	*	*	*
1	10.3.*.*	10.4.*.*	80	*	17
2	10.5.*.*	10.6.*.*	80	*	17
3	10.5.*.*	10.6.*.*	[20,21]	*	6
4	10.7.*.*	10.7.*.*	*	gt 1023	6
5	*	*	*	*	*

①分类 ID，②目的 IP，③源 IP，④目的端口，⑤源端口，⑥协议号.

定义 13. 若 $a \in D\_Set, b \in S\_Set, c \in P\_Set$，则三元组 (a,b,c) 称为一个交叉组合.

将所有交叉组合的集合进一步划分成不同的等价类（该等价类的集合记为 DSP_Set），划分方法如下：两个交叉组合 (a,b,c)，(d,e,f)，如果 $bmp(a)\&bmp(b)\&bmp(c)$ 和 $bmp(d)\&bmp(e)\&bmp(f)$ 相同，则 (a,b,c) 和 (d,e,f) 属于同一个交叉组合等价类，否则属于不同的等价类.

与 DSP_Set 中每个元素对应的有一个目的-源 IP 前缀对集合，其元素为与 DSP_Set 中某个元素的端口和协议号匹配的过滤规则的目的 IP 前缀和源 IP 前缀对. 无冲突哈希 Tric 树算法先根据包头 H 中的目的端口、源端口和协议号，通过无冲突的哈希查找方法找到与上述交叉组合元素匹配的目的-源 IP 前缀集合的指针. 然后在目的-源 IP 前缀对中对 H 进行二维的 IP 分类，得到最终分类结果. 属于同一个 DSP_Set 元素的交叉组合共享一个目的-源 IP 前缀对集合指针.

3.2 无冲突哈希查找

对一个包头 $H(dport, sprot, proto)$（分别表示目的端口、源端口和协议号，此处暂不考虑两个 IP 域）. 分别以 $dport$，$sport$ 和 $proto$ 为索引查表，得到 $fd(dport)$，$fs(sport)$ 和 $fp(proto)$，然后选取以它们的某个函数 $g(fd(dport), fs(sport), fp(proto))$ 为索引再进行查表，得到 $h(g(fd(dport), fs(sport), fp(proto)))$，此值即为目的-源 IP 集合的指针. 对 D_Set 中所有的 D 个等价类以 0,1,2,...,$D-1$ 依次编号，$fd(dport)$ 定义为端口 $dport$ 对应的等价类号. fs 和 fp 以同一方法定义.

上述过程如图 1 所示. 方框表示查表，g 是哈希函数. 选择 $g(d,s,p)=PSd+Ps+p$，根据 fd，fs 和 fp 的定义，我们有 $0 \leqslant d \leqslant D-1$，$0 \leqslant s \leqslant S-1$，$0 \leqslant p \leqslant P-1$. 下面证明函数 g 是无冲突的.

定理. 定义 $g(d,s,p)=PSd+Ps+p$ $(d,s,p,D,S,P \in Z)$，其中 $0 \leqslant d \leqslant D-1$，$0 \leqslant s \leqslant S-1$，$0 \leqslant p \leqslant P-1$，若 $g(d_1,s_1,p_1)=g(d_2,s_2,p_2)$，则 $d_1=d_2$，$s_1=s_2$，$p_1=p_2$.

证明：由 $PSd_1+Ps_1+p_1=PSd_2+Ps_2+p_2$，得到 $p_2-p_1-PSd_1+Ps_1-PSd_2-Ps_2=P[S(d_1-d_2)+s_1-s_2]$，两边取绝对值得 $|p_2-p_1|=P|S(d_1-d_2)+s_1-s_2|$，因为 $p_1,p_2 \in [0,P-1]$，所以 $|p_2-p_1|<P$. 又由于 $|S(d_1-d_2)+s_1-s_2|$ 是整数，所以只能有 $p_2-p_1=0$，即 $p_2=p_1$. 所以 $S(d_1-d_2)+s_1-s_2=0$，$|s_2-s_1|=S|d_1-d_2|$，同理，

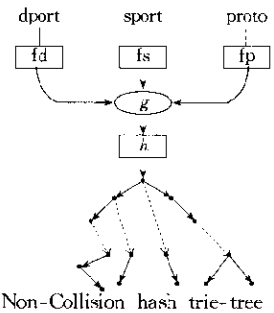


Fig. 1 Non-Collision hash trie-tree algorithm

图1 无冲突哈希Trie树算法

有 $s_2=s_1, d_1=d_2$. □

上面证明了 g 是一个无冲突哈希函数. 各表项以及二维 Trie 树的建立是在 pre-compute 过程中,通过两次扫描过滤规则数据库来完成的. 这样,根据包头中的目的端口、源端口和协议号,通过 4 次访存(或两次并行访存),必能得到目的-源 IP 前缀集合的指针,接下来将根据包头的目的-源 IP 和进行二维 Trie 树的查找. 上述各表的建立的算法描述见算法 1 和算法 2.

算法 1. *fd* 表的构建(*fs* 和 *fp* 表可同法构建)

(1) 初始化,为 *fd* 表分配 65 536 个表项空间,等价类计数器 *cc* 初始化为 0;

(2) 对 n 从 0~65535,进行步骤(3)和步骤(4);

(3) 扫描过滤规则的 *dport* 域对应的列,求 n 的 *bmp*;

(4) 如 n 的 *bmp* 出现过,则 n 属于 *bmp* 对应的 *D_Set* 等价类,将 n 属于的等价类的 id 填入表 *fd* 的第 n 个表项;否则 n 属于新的等价类,该等价类的 id 值为 *cc* 当前值,将其填入表 *fd* 的第 n 个表项,同时 *cc* 加 1.

算法 2. *h* 表的构建

(1) 为 *h* 表分配 $D\times S\times P$ 个表项空间,同时初始化等价类计数器 *cc* 为 0,n 初始化为 0;

(2) 对 *D_Set*,*S_Set* 和 *P_Set* 的等价类元素依照 id 递增顺序进行交叉组合. 对每个 $eqd\in D\_Set$,$eqs\in S\_Set$,$eqp\in P\_Set$,进行步骤(3);

(3) 求 $bmp=bmp(eqd)\&bmp(eqs)\&bmp(eqp)$(逐位与). 如果 *bmp* 已经出现过,则 n 属于 *bmp* 对应 *DSP_Set* 的等价类,将 n 属于的等价类的 id 填入 *h* 的第 n 个表项;否则 n 属于新的等价类,该等价类的 id 值为 *cc* 当前值,将其填入表 *h* 的第 n 个表项,同时 *cc* 加 1,n 加 1.

3.3 目的-源 IP 对的查找

在这一部分中,我们在 Grid of Tries 二维分类算法的基础上提出一种简化的二维 Tries 树查找算法. 我们将数据结构中的 Trie 树从一维扩展到二维,形成二维 Trie 树. 以表 2 中的数据库为例来说明这个扩展过程(假定表中 IP 地址的宽度为 2).

Table 2 An example of destination-source IP prefix pairs

表 2 目的-源 IP 前缀对集合示例

Class ID①	Dest-IP②	Src-IP③
0	0*	10*
1	0*	01*
2	0*	1*
3	00*	1*
4	00*	11*
5	10*	1*
6	*	00*

①分类 id,②目的 IP,③源 IP.

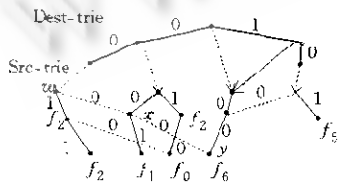


Fig. 2 Improved data structure of 2D trie tree

图2 改进的二维Trie数据结构

先不考虑过滤规则数据库中源 IP 地址,根据过滤规则数据库中目的地址前缀构造一棵 Trie 树(称为 Dest-Trie 树). 对于这个目的前缀 Trie 树中的每个节点,如果在数据库中存在其对应的目的地址前缀,则会指向一棵源地址 Trie 树(称为 Src-Trie 树),否则相应的指针为空(如图 2 中的空心圆圈所示). 任意一个 Dest-Trie 树中的节点不但包含过滤规则数据库中与该节点对应的源地址前缀,还应当包含该目的前缀的"前缀"(即它在 Dest-Trie 树中的祖先)所有的源地址前缀,其时间复杂度为 $O(W)$. 但每个 Dest-Trie 节点不但存储它自身的源地址前缀,还存

储其祖先的源地址前缀,存在大量的冗余拷贝.在最坏情况下,其空间复杂度可达到 $\Theta(N^2)$.

可以去掉这些多余的拷贝.每个目的地址前缀只包含在过滤规则数据库中与该目的地址配对的源地址前缀.当去掉冗余拷贝之后,查找一个目的-源 IP 对应的最小代价的 classID,不仅要从目的地址的最长匹配前缀所指向的 Src-Tric 树中进行搜索,还必须对该目的地址的最长匹配前缀的祖先所指向的 Src-Trie 树中进行搜索,从搜索所经过的路径中找出代价最小的过滤规则的 classID.由于查找过程中需要进行回溯,虽然节省了空间,但是时间复杂度却上升到 $O(W^2)$.

上述问题的解决方法是在上面的基础上引入转向指针,即通过预处理将 Src-Trie 树中的空指针指向其 Dst-Trie 树的某个祖先对应的 Src-Trie 中的节点,使得在查找过程中沿着最长匹配路径能够尽可能地前进.此外,为了确保算法的正确性,还必须保证目的-源 IP 前缀对中前缀长的节点,对应的代价也小.以表 2 中的规则 2、3 和 4 为例,它们的目的-源 IP 前缀 2 比 3 短,3 比 4 短.但是,2 的代价小于 3,3 的代价小于 4,这不符合上述前缀越长,代价越小的准则.实际上,将规则 3 和 4 从表 2 中去掉并不影响查找的结果,因为与 3、4 匹配的规则必然与 2 匹配,而且 2 的代价小,所以,实际上,3 和 4 永远没有匹配的机会.针对这种情况,有两种解决方法,一种办法是通过预处理一开始就保证过滤规则数据库 FS 中不存在冗余规则;另一种办法是在数据结构预计算(pre-compute)中进行处理,比如在数据结构中将规则 3 和 4 中存储的 ClassID 改为 2,即规则 2 的 ClassID,这样可以保证算法的正确性.

最后的二维 Trie 树如图 2 所示.其中"f"旁边的数字表示 ClassID(也是规则序号和代价).根据图 2,对一个目的-源 IP 对,查找最小代价过滤规则的过程如下:先对目的地址做最长前缀匹配,然后沿着目的地址最长匹配前缀所指向的 Src-Trie 根据包头的源地址沿着 0、1 指针(或者转向指针)尽可能地前进,直到无法继续前进为止,沿途经过的结点中存储的过滤规则代价最小的过滤规则所对应的 classID,即是其分类的 classID.

限于篇幅,二维 Trie 树的建立、预处理和查找过程的算法描述略.

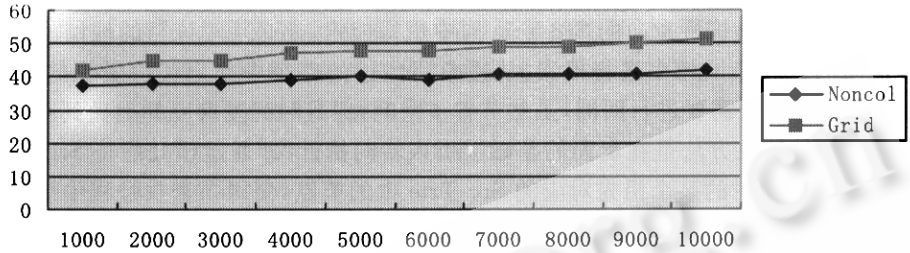
3.4 性能分析

在最坏情况下,得到二维 Tric 树的指针需要经过 4 次查表(串行查找),即查找 fd,fs,fp 和 h.沿二维 Trie 树查找,最坏情况下需要查找的节点数目是 $2W$,因此最坏情况下需要访存 $2W+4$ 次,查找时间基本与过滤规则的数目无关.对 Grid of Tries,即使假定其 Hash 函数选得足够好,最坏情况下的访存次数为 $4(1+2W)$.

无冲突哈希 Trie 树查找的空间性能稍微复杂一些.fs,fd 和 fp 的表项数目都是 65536,h 的表项数量为 $D\times S\times P$.从理论上讲,h 的表项可达到 $65536\times65536\times65536$.但实际情况是 D,S 和 P 是极小的,所以 h 的实际表项也是极小的.再考虑二维 Trie 树,由于一条过滤规则最多需要 $2W$ 个 Trie 树节点,一共有 N 条过滤规则,所以二维 Trie 树的空间最坏情况为 $2NW$.因此,空间复杂度可粗略地估算为 $Table\_Size+2NW$.多维 Grid of Tries 的空间差不多也为 $Hash\_Size+2NW$,其中 $Hash\_Size$ 为哈希表所占的空间,为了保证时间效率,通常 $Hash\_Size$ 要占很大的空间.从测试结果来看,平均的空间效率也是无冲突哈希 Trie 树的效率要好一些.

从理论上分析平均空间性能和平均时间性能是比较困难的,而且目前在 IP 分类研究领域,对过滤规则和数据包的采样工作还很不完善.我们设计了一个虚拟环境测量其平均性能,由于只比较它与 Grid of Tries 的相对性能,因此这个虚拟环境是满足要求的.对 IP 包流和过滤规则中 5 个域,依据能够得到的原始数据作出合乎情理的假设,然后通过随机发生器产生 IP 包流和过滤规则数据库.为了更具可比性,对过滤规则中端口和协议号的限制遵从 Grid of Tries 算法的约定.即使加入

对 Grid of Tries 有利的这样一个假设,无论从时间性能还是空间性能上来讲,也是无哈希冲突 Trie 树查找的性能要好.因此,在实际应用中,二者在性能上的差距会更明显.测量结果如图 3 和图 4 所示.

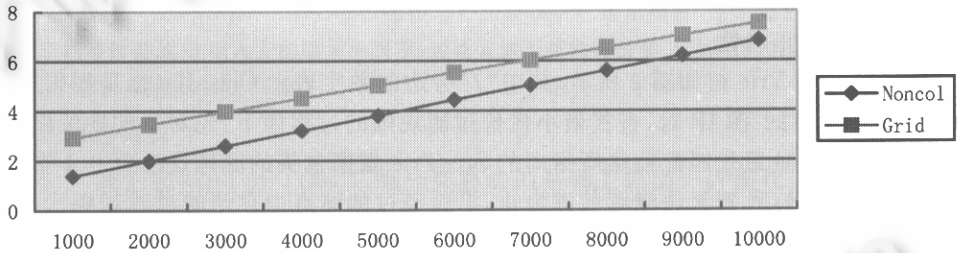


x-Axis plots the number of filters and y-axis plots the total seconds consumed while processing 10^7 packets①

① Noncol 代表无冲突哈希 Trie 树算法,Grid 代表 Grid of Tries,横坐标表示过滤规则数目,纵坐标表示对 10^7 个包头进行分类所需总的时间.

Fig. 3 Comparison of time performance between non-colision trie-tree (denoted by Noncol) and Grid of Tries (denoted by Grid)

图 3 无冲突哈希 Trie 树和 Grid of Tries 树时间性能比较图



x-Axis plots the number of filters and y-axis plots the maximum memory (MB) consumed①

① Noncol 代表无冲突哈希 Trie 树算法,Grid 代表 Grid of Tries,横坐标表示过滤规则数目,纵坐标表示消耗的最大内存(MB).

Fig. 4 Comparison of space performance between non-colision trie-tree (denoted by Noncol) and Grid of Tries (denoted by Grid)

图 4 无冲突哈希 Trie 树和 Grid of Tries 树空间性能比较图

4 结束语

本文提出的无冲突哈希 Trie 分类算法已经成功地应用于国家 863 高科技发展计划重点攻关项目"高性能安全路由器"的研制之中.在"高性能安全路由器"中集成了分组过滤和 IPSec 功能,具有 4 个千兆以太接口,4 个 155M SDH 接口和 8 个快速以太接口.在使用千兆以太这种高速接口的情况下,传统的分类算法不能满足其线速分类的要求.而本文中提出的无冲突哈希 Trie 分类算法完全可以达到线速分类分组的性能要求.

无冲突哈希 Trie 树算法还可以进一步改进.在二维 Trie 树的查找过程中,查找每次是根据 1 个比特前进一步,如果每次查找多个比特,那么 Trie 树的深度会大大降低,时间性能会进一步提高,但这需要对过滤规则进行扩展,增加了额外的内存.下一阶段的工作,我们希望通过研究 IPMA (Internet performance measurement and analysis project)[11]的 IP 前缀长度的分布,根据前缀长度的分布有效地解决如何选择 Trie 树的最大深度以及每次通过查找哪些比特前进等问题.

References:

[1] Weiss, W. QoS with differentiated services. Technical Journal, 1998,3(4):48~62.

[2] Bellovin, S., Cheswick, W. Network firewalls. IEEE Communications Magazine, 1994,32(9):50~57.

[3] Jyh-haw, Y., Randy, C., Richard, N. W. Interdomain access control with policy routing. In: Chang, C., Fabre, J. C., eds. Proceedings of the IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems. Los Almitos, CA: IEEE Computer Society, 1997. 46~52.

[4] Edell, R., McKeown, N., Varaiya, P. Billing users and pricing for TCP. IEEE Journal on Selected Areas in Communications, 1995,13(7):1162~1175.

[5] Xu, Ke, Xiong, Yong qiang, Wu, Jian-ping. Analysis of broadband IP router architecture. Journal of Software, 2000,11(2):179~186 (in Chinese).

[6] Gupta, P., McKeown, N. Packet classification on multiple fields. ACM Computer Communication Review, 1999,29(4):146~160.

[7] Bailey, M. L., Gopal, B., Pagels, M. A., *et al*. PATHFINDER: A pattern-based packet classifier. In: Lepreau, J., Bershad, B., eds. Proceedings of the 1st Symposium on Operating System Design and Implementation. Monterey, CA: Usenix Association, 1994. 95~104.

[8] Srinivasn, V., Varghese, G., Suri, S., *et al*. Fast scalable level four switching. ACM Computer Communication Review, 1998,28(4):191~205.

[9] Woo, T. Y. C. A modular approach to packet classification: algorithms and results. In: Grueir, R., ed. Proceedings of the IEEE Infocom2000. San Francisco, CA: IEEE Computer Society Press, 2000. 1210~1217.

[10] Stevens, W. R. UNIX Networking Programming vol 1. 2nd ed. Englewood Cliffs, NJ: Prentice Hall Inc., 1998.

[11] Merit Inc. IPMA statistics. http://nic.merit.edu/ipma.

附中文参考文献:

[5] 徐恪,熊勇强,吴建平.宽带 IP 路由器的体系结构分析.软件学报,2000,11(2):179~186.

A Fast IP Classification Algorithm Applying to Multiple Fields*

YU Zhong-chao, XU Ke, WU Jian-ping

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

E-mail: yzc@csnet1.cs.tsinghua.edu.cn

http://netlab.cs.tsinghua.edu.cn

Abstract: As the network applications develop, routers must support those functions such as firewalls, provision of QoS and traffic billing etc. All these functions need classification of IP packets, according to which it is determined how different packets are processed subsequently. In this paper, a new IP classification algorithm is proposed based on the Grid of Tries algorithm. The new algorithm not only eliminates original limitations in the case of multiple fields but also shows better performance in regard to both time and space. It has better overall performance than many other algorithms.

Key words: IP-classification; route-lookup; Trie-tree; IPSec

* Received April 13, 2000; accepted June 27, 2000

Supported by the National Natural Science Foundation of China under Grant No. 90104002; the National High Technology Development 863 Program of China under Grant No. 863-306-ZD-07-01