路由器中的包分类算法研究*)

甘利杰
(重庆工商大学计算机科学与信息工程学院　重庆 400067)

摘　要　在Internet路由器中将数据包分类成流采用了散列算法的基本思想，并引入了流的局部性原理来加速散列查找的过程，用软件对该算法进行了仿真测试，并在最后从时间复杂度和空间复杂度两个方面对其进行了性能分析。实验结果表明，该算法能够快速地实现分流。

关键词　包分类，数据包，流

The Study of an Algorithm for Packet Classification of Router

GAN Li-Jie
(Computer Science & Information Engineering College, Chongqing Technology & Business University, Chongqing 400067)

Abstract　The process of categorizing packets into "flows" in an Internet router is called packet classification. All packets belonging to the same flow obey a pre-defined rule and are processed in a similar manner by the router. The main idea is Hash algorithm. How to speed the hash search with the localness of flow has been introduced. At last, its performance of time complexity and space complexity is analyzed. The analysis shows that this algorithm has nice time complexity and space complexity and can achieve fast shunt.

Keywords　Packet classification, Data packet, Flow

1　引言

随着Internet规模的不断扩大与应用技术的不断进步，越来越多的业务需要对数据包(Packet)[1~4]进行实时、快速的分类。在对分类算法的应用环境进行分析后，发现分类算法所针对的规则库可以是静态的，也可以是动态的。静态的规则库比如路由表查找中的路由表，包过滤技术中的过滤器等；动态规则库比如基于流的应用中的流表。因此可以把数据包分类算法分为两个大的类别，它们分别是针对静态规则库而采用的数据包分类算法和针对动态规则库采用的数据包分流算法，它们的相同点在于这两种算法都是把一个数据包与已有的规则进行匹配的过程。

数据包的结构非常复杂，主要由"目的IP地址"、"源IP地址"、"净载数据"等部分构成。数据包处理包含数据包拆卸和组装；数据包分类；数据包修改；业务/流量管理；队列和策略管理；安全性处理；控制和管理等功能。数据包处理是指对通过数字通信和网络设备的数据包进行处理，是网络处理的关键功能。数据包处理既存在于网络核心，也存在于接入/边缘网络。设备在网络中的位置以及服务供应商的需求决定了所需要的数据包处理的类型和数量、线路速率和功能，也决定了数据包处理的正确方法，以及半导体器件的选择。

基于类的数据包分类算法和基于流的数据包分类算法在规则库和流表的表现形式和规模上的不同，针对它们各自的特点在算法的研究上有各自的侧重点。比如基于类的数据包分类算法主要解决对于类的优先级的处理，而基于流的数据包分类算法也主要解决流表的大规模和变化频繁所带来的问题。

应用于高速网络业务量测量系统中的基于"类"的数据包分类算法，用于针对网络管理员指定的规则库进行匹配分类，以作为统计分析的前提。

2　包分类算法

基于"类"的数据包分类算法由两个部分构成：预处理部分和查找部分[4]。算法[5]的示意图如图1所示。

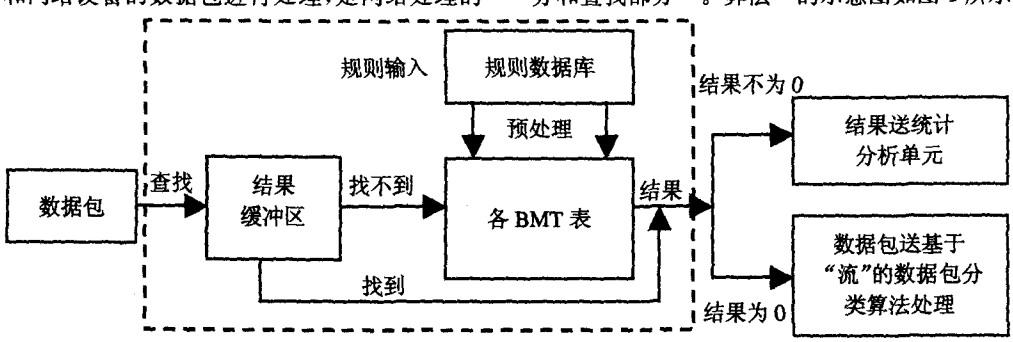

图1　数据包分类

*)本研究由重庆市自然科学基金支持项目(No.2004BB2182)资助。甘利杰　主要研究方向：软件工程、操作系统等。

2.1 预处理

预处理部分完成对规则的预处理和对规则变化时 BMT 表的维护。

第一步：根据各个地址字段建立相应的 BMT 表 BMT1～BMT K。每个 BMT 表的容量为 $2m \times n$（m 为该 BMT 表对应地址字段的长度，n 为规则数）。各个 BMT 表的存储单元初始值为 0。表示没有规则对应。

第二步：根据各规则所涉及的地址字段到各个相应的 BMT 表中进行处理。如果规则对应某个地址字段值，则到对应的 BMT 表的对应该规则的比特置“1”，表示规则和地址字段匹配。如果规则和某个地址范围匹配，则到对应的 BMT 表的对应区域把该规则对应的比特置“1”，表示规则和这些地址字段匹配。

第三步：当规则发生变化的时候，需要对 BMT 表进行维护：

当添加一条新规则的时候，根据该规则所对应的各个地址字段值，到各 BMT 表中的对应该规则的比特置“1”，表示规则和地址字段匹配。

当删除一条规则的时候，根据该规则所对应的各个地址字段值，把各 BMT 表中相应该规则的比特清“0”，表示任何一个数据包都不能和这条规则匹配。实际上就意味着这条规则不存在了。以后还可以用一条新的规则替代这条旧规则。

当对一条规则进行修改的时候，首先根据旧规则所对应的各个地址字段，把各 BMT 表的相应该规则的比特清“0”，表示规则不再和该地址相匹配。然后根据新规则所对应的各个地址字段，到各 BMT 表的相应新规则的比特置“1”，表示新规则和该地址单元匹配。

2.2 查找

查找算法完成对每一个输入数据包的查找。

(1)把数据包到结果缓冲区中进行查找，查找可以用线性查找、二分查找、散列(Hash)查找等一些常见的算法。如果能匹配，则对应的结果中“1”比特对应于和该数据包匹配的规则，否则，转(2)。

(2)根据数据包的各个地址字段的值到各个相应的 BMT 表中查找对应的表项。

(3)把各个表项的值进行“与”运算，得到结果为“1”的比特位对应于和该数据包匹配的规则。

3 包分类算法实现

Main()函数先从数据包数据文件中找到源地址行取出源地址，然后读取目的地址行取出目的地址，合并源地址和目的地址得字符串 sLinetemp，由它可以实例化一个 hash 表项，然后查找 hash 表，查找不成功则在 hash 表里插入一个新的项，如果插入失败表示 hash 表已满，这个时候要更新 hash 表，去掉其中一个项，然后将新项插入。

程序中主要函数的功能：

public class hashtablee()；hashtablee 类，含有关键字 key、标志 flag、标志 hash 表上这个位置是否为空。

public void init(ref hashtablee[]ht)；hash 表初始化函数，将各个节点的标志置为空。

public bool insert(ref hashtablee[] ht, hashtablee e)；往 hash 表插入一个元素，成功后返回 true；不成功返回 false。

public int search(hashtablee[]ht, string k)；查找 hash 表 ht，根据字符串即源地址和目的地址计算关键字 key，以 key 为地址从 hash 表中取出 data 与 k 比较，看是否相等，如果相等，则得到分流结果，如果不相等，采用线性探测再散列法继续向后查找，查找到则得分流结果，否则表示它是一个新的流，再看散列表是否满，若没有满，则直接插入，否则去掉散列表中一个值，将这个新的值 k 插入到散列表中。

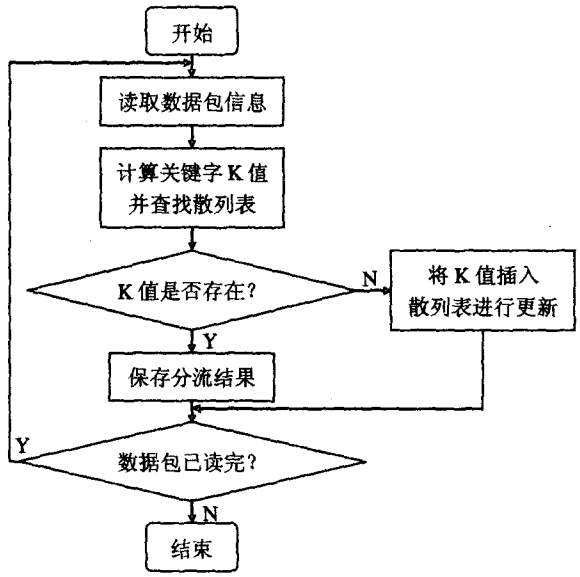

图 2 主程序流程图

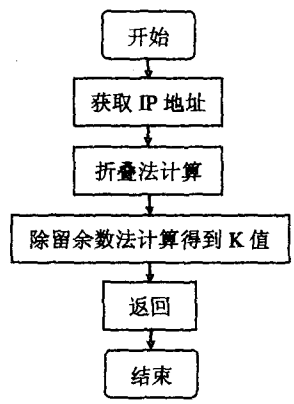

图 3 K 值计算流程图

public void updatetable(ref hashtablee[]ht, string k)；更新 hash 表，去掉一个不常用的关键字，用关键字 k 替代，查找不成功时将用到。

public int delete(ref hashtablee[]ht)；删除一个最近不常用的关键字。

public class packetline；存储数据包的一行。

public class buffer；缓冲区，可以存储数据包的若干行。

public static void getnext, public static int findpos；用于定位到某个数据包。

4 包分流算法实现

基于“流”的数据包分类算法的特点是流表的容量大，流表的更新速度较快。Jun Xu 等人提出了用散列(Hash)的方法来进行处理，在硬件实现方案中，Jun Xu 等人设计了一个面向第四层路由的高速缓存体系结构，它可以达到很高的命中率(92%)，并且命中率稳定(方差 0.6%)，这是传统的缓存管理算法无法达到的，Jun Xu 等人采用了近似 LRU 算法，该算法充分运用了 Internet 的第四层流量的本地性行为。Jun Xu

(下转第 63 页)

在运行这段代码时，后台进行了以下的工作。

1. 当初始化时，LoginContext 对象首先在 JAAS 配置文件中找到 Sample 文件，然后根据配置文件的内容决定该加载哪个 LoginModule 对象。

2. 在登录时，LoginContext 对象调用每个 LoginModule 对象的 login()方法。

3. 每个 login()方法进行验证操作或获得一个 CallbackHandle 对象。

4. CallbackHandle 对象通过使用一个或多个 CallBack 方法同用户进行交互，获得用户输入。

5. 向一个新的 Subject 对象中填入验证信息。

在鉴别实现中，我们在服务提供者 P 和可下载到本地的代理 proxy 上分别部署一个 Login 模块。在可下载到本地的代理 proxy 上，根据 ID_A 信息，以及随机数 n_A 和共享密钥 k，计算 $h(k, ID_A, n_A)$并于服务提供者 P 发送过来的值比较，实现对服务提供者 P 的鉴别。如果相同，则判断服务提供者 P 是合法用户，同时生成一个随机数 n_P，并把 ID_P，n_P，$h(k, ID_P, n_P)$发送给服务提供者 P。代码可以表示为：

if AuthCode= $h(k, ID_A, n_A)$ {……}

在服务提供者 P 的 Login 模块中，根据 ID_p，随机数 n_P 和共享密钥 k，计算 $h(k, ID_P, n_P)$并于代理 proxy 发送过来的值比较，实现对代理 proxy 的鉴别。如果相同，则判断代理 proxy 是合法代理 proxy。代码可以表示为：

if AuthCode=$h(k, ID_P, n_P)$ {……}

5 安全性分析

本文提出了一个 Jini 系统中服务提供者和服务请求者的鉴别协议，并通过双方交互建立安全连接，具有以下特点：

(1) 服务提供者和服务请求者共享一对称密钥，使用带密钥的 Hash 函数实现了双方的双向认证。

(2) 每次认证时，双方都选择不同的随机数，可以防止攻击者的重发攻击。

(下转第 75 页)

(上接第 55 页)

等人还提出一种称为动态集相关的模型，这是一种基于新型的统计独立的并行散列模型。该技术可以降低散列冲突不命中率达到 75%～90%，使用惰性回写技术，吞吐量可以进一步达到 37.5/75Mpps。

4.1 散列算法

散列(Hash)算法的基本思想是以关键字的值为自变量，通过一定的函数关系，计算出对应的函数值来，把这个值解释为节点的存储地址，将节点存入到这个存储单元里去。查找时再根据要查找的关键字用同样的散列函数计算地址，然后到相应的地址单元里去取要找的节点。所以这种方法也被称为关键字-地址转换法。用散列法存储的线性表叫做散列表。在散列表里可以实现对节点进行快速的查找。

散列算法由两个部分构成：散列计算和散列查找，散列计算的计算过程应该做到两个方面：一是计算简单，二是计算应该尽量减小散列冲突。散列计算比较常用的算法是平方法、折叠法、除留余数法等，在本算法中，采用了折叠法加除留余数法。即首先把流的地址字段分段，然后进行顺叠相加，最后把相加的结果和散列表的大小进行模运算，得到散列地址。这样处理计算简单，得到散列地址也有较好的随机性。

散列函数的选取可以减少冲突但是不能避免冲突，因此如何解决冲突就是散列法中一个必须要解决的问题。处理散列冲突的方法基本上有两类，一类叫做拉链法，另一类叫做开地址法。在本算法中，采用了拉链法进行解决散列冲突。

4.2 定时清除

随着时间的推移，冲突次数的增加，存储空间如果不做处理的话将会被耗尽。另一方面，某些流已经退出使用，比如某个 IP 地址已经不在进行网络使用，则在内存中保存的记录也应该被清除，把数据记录保存到外存中，比如数据库中。释放的存储空间可以为后面到达的新流继续使用。在数据包分类算法中，采用了定时清除的方法，每个数据包到达之后用该数据包的时间戳更新所属流的时间标记，以反映该流的活跃时间，然后定时清除已经不再活跃的流。定时清除部分完成定时对散列表中的记录进行扫描，清除已经不再活跃的流，释放存储空间。

分流部分的流程如下：

(1)根据源 IP 地址、目的 IP 地址进行散列计算，计算出关键字 K，散列函数可选取折叠法加除留余数法；

(2)根据散列计算的结果关键字 K 的值，到散列表中进行散列查找。如果找到则得到分流的结果。如果找不到则转(3)；

(3)如果找不到则说明这是一个新流，则为这个流建立一个新的散列表项，插入到关键字 K 对应的同义词子表，作为第一个节点。

5 算法复杂度分析

算法的执行时间由散列计算的时间(记为 T 计算)和散列查找的时间(记为 T 查找)组成，执行时间为 T 计算＋ T 查找 。通常情况下，计算的时间要远小于查找的时间，所以算法的时间约等于 T 查找，而散列查找的时间 TS 要取决于冲突的次数。当查找的关键字是 N，散列表的基本区的大小是 M 的时候，散列查找的平均次数是 $N/2M$，算法的时间复杂度是 $O(N/M)$。同时，算法中使用的流的局部性原理可以加速查找过程，最好的情况下一次查找就能得到结果。

算法的占用存储空间主要是散列表的存储空间 UHash。而散列表主要用于对系统中的流进行存储。假设系统中流的数目是 M，且每个流的记录要占用 N 个字节，则 UHash 要占用的存储空间是 $M \times N$，算法所占用的存储空间大约是 $O(N)$。

结束语 程序的运行结果根据不同的环境，结果有所不同。影响结果的因素有，硬件环境(即仿真计算机的配置)、所确定的散列表的容量、处理的数据包数、处理数据包的相关性、包在散列表中的活跃时间、定时扫描的时间等。算法分析表明算法具有良好的时间复杂度和空间复杂度，算法可以实现快速的分流。

参 考 文 献

1 付歌，杨明福.一个快速的二维数据包分类算法. 计算机工程. 2004,30(6):76～78

2 Telikepalli A. 数据包处理方法和解决方案[J]. 今日电子，2002,7:21～25

3 小高知宏. TCP/IP 数据包分析程序篇[M]. 北京:科学出版社，2003

4 Comer D E,等著.用 TCP/IP 进行网际互联(第二卷:设计、实现与内核)[M]. 张娟,等译. 北京:电子工业出版社，2003

5 徐士良. 计算机常用算法. 北京:清华大学出版社，1995