Hash 函数实现数据包分流算法研究*)

瞿 中[1,2] 邱玉辉[3]

(重庆邮电学院计算机科学与技术学院 重庆 400065)[1] (重庆大学计算机学院 重庆 400030)[2]

(西南师范大学人工智能研究所 重庆 400715)[3]

摘 要 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进行实时、快速的分类,对数据包分类的研究具有重要的现实意义。Hash 算法采用了散列算法的基本思想,并引入了流的局部性原理加速散列查找的过程。由于时间精确度较高和面向对象的特点,选用了 C++ 语言编程对该算法进行了仿真测试,最后对 Hash 算法分析表明,Hash 算法具有良好的时间复杂度和空间复杂度,可以实现快速的分流。

关键词 Hash 算法,数据流分类,流的局部性原理,数据包分类

Research on Hash Algorithm of Packet Classify Into Flow

QU Zhong[1,2] QIU Yu-Hui[4]

(College of Computer Science & Technology, Chongqing University of Post & Telecom, Chongqing 400065)[1]

(College of Computer Science, Chongqing University, Chongqing 400030)[2]

(AI Research Center of South West China Normal University, Chongqing 400715)[3]

Abstract With the development of Internet technology and improvement of application technology, real-time and fast packet classifications have been applied to more and more services. In this article, an algorithm with fast classify packet into flow was give. It takes Hash algorithm as main idea, and speeding hash search with the localness of flow. According to the characters of high time precision and object-oriented, C++ to emulate this algorithm was chosen. At last, time complexity and space complexity were analyzed. The analysis shows that this algorithm has nice time complexity and space complexity and can achieve fast shunt.

Keywords Hash algorithm, Flow classification, Localness of flow, Packet classification

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[1,2]分类算法分为两大类,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(静态的"类"所对应的规则和动态的"流"所对应的规则)进行匹配的过程[3,4]。但是两种算法有着本质的不同点。主要表现在以下两个方面:第一,规则库的表现形式不同;第二,规则库的规模不同,在相同网络环境中进行考察,一般来说规则库的规模较小,而流表的规模较大。

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,如图 1 所示。

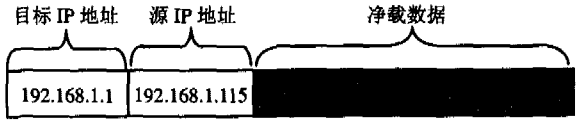

图 1 数据包的结构

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。

对于数据包处理,已经有大量的文献试图重新定义数据包处理的含义[5]。数据包处理包含七项基本功能:

(1)数据包拆卸和组装;(2)数据包分类;(3)数据包修改;(4)业务/流量管理;(5)队列和策略管理;(6)安全性处理;(7)控制和管理。

一些科研人员经常仅将前五项功能称为数据包处理,而把所有七项功能作为网络处理。网络处理与网络处理器(NP)是两个不同的概念,网络处理是指上面提供的功能,而网络处理器是完成网络处理的特殊半导体器件[5]。

2 "流"的局部性原理

在不同的情况下,可以对"流"进行不同的定义。比如"具有相同的源地址的所有分组";"具有相同的源/目的地址的所有分组";"属于同一个 TCP 连接的所有分组"等[6]。本文考虑了一个折衷的定义,把"具有相同的源/目的地址的所有分组"定义为"流"。因此动态流的分流就是要把具有相同的源/目的地址的分组区别为不同的流。但是需要指出在其他的应

*)项目基金:重庆邮电学院青年教师/社会科学基金项目(No. A2004-19),国家十五重大科技计划项目(No. 2002BA107B)和重庆市自然科学基金支持项目(No. 2004BB2182)。瞿 中 博士研究生,主要研究方向:计算机应用技术、数据包分类技术、计算机系统结构等。邱玉辉 教授,博士生导师。主要研究方向:计算机应用技术、人工智能等。

用场合中，只需更改“流”的定义，算法也同样适用。

目前网络中的多数应用服务，比如 WEB 页面，FTP 文件，将被拆分为一个个数据进行传输，而这些数据包具有相同的地址字段：源 IP 地址、目的 IP 地址、源端口号、目的端口号、协议字段等，根据“流”的定义，这些数据包属于同一个流，因此这些 WEB 页面或者 FTP 文件属于同一个流。这些 WEB 页面或者 FTP 文件的第一个数据包到达之后，在一个较短时间内到达另一个属于同一个流的数据包的可能性就非常大。这个特性在本文中称之为“流”的局部性原理（Localness of flow）[6]。

3 散列(Hash)算法

散列(Hash)算法也称为杂凑法，是一种常见的检索方法，其基本思想是以关键字的值为自变量，通过一定的函数关系（散列函数），计算出对应的函数值来，把这个函数值解释为节点的存储地址（散列地址），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 散列查找中“流”的局部性原理的应用

在进行数据包分流的时候，如果把属于同一个流的第一个到达的数据包的分流结果进行缓存，则之后到达的同属一个流的数据包和这个保存的结果直接比较而得到结果的可能性就比较大。这样就可以省略对散列表的查找过程，在最好的情况下一次查找就可以得到分流的结果。如图 2 所示。

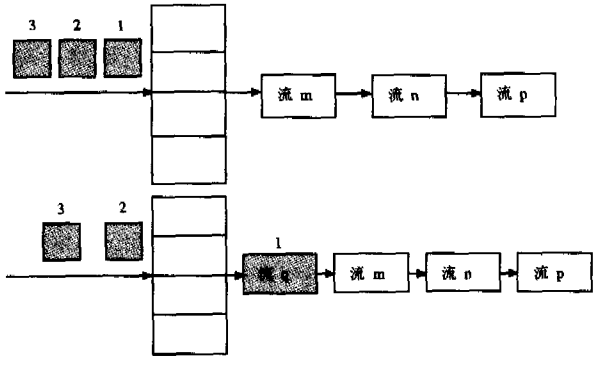

图 2 散列查找中“流”的局部性原理的应用

5 定时清除和“流”的局部性原理的使用

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。在本算法中，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。

另外，在更新流的时间标记的时候，将再次运用流的局部性原理。一个新流产生之后，随后在较短的时间内到达同一个流的可能性也较大。如果用抽样的数据包更新所属流的时间标记，而不是用每个数据包的时间戳来更新所属流的时间标记，则可以减少若干次对内存的写操作，这样可以进一步提高分类速度，如图 3 所示。

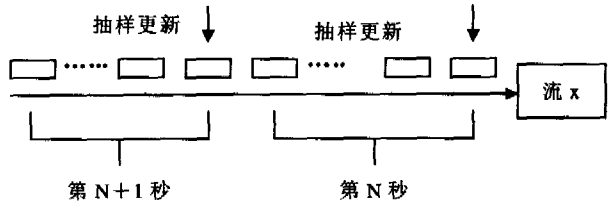

图 3 定时清除中“流”的局部性原理的应用

本算法分为两个部分：分流部分和定时清除部分。

定时清除部分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：

第一步：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；

第二步：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。如果找不到则转第三步；

第三步：如果找不到则说明这是一个新流，就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。

6 算法的仿真和性能分析

6.1 算法流程

本程序采用了 C++ 编程来实现算法，以达到仿真的目的，仿真用到的数据包文件是由 Windup 软件通过监听网络并对网络捕获包产生的。要运行 Windup 先要安装 Wincap 软件，然后在命令提示符下先进入 Windup 的安装目录，然后运行 windup -f -n>f:\weizhi\s. txt，这样将 windup 对网络监听得到的信息定向在 f:\weizhi\s. txt 文件里，对监听的信息进行整理，得到如下形式的数据包：

172. 19. 88. 25. 133 > 202. 202. 43. 121. 23

172. 19. 88. 59. 138 > 172. 19. 91. 255. 138

仿真过程的由 VC 实现，在 VC 中散列表的容量是 15，包在散列表中的活跃时间是处理 50 个数据包的时间，定时扫描的时间是处理 10 个数据包的时间。每次读入缓冲区的数据包数也是 10，总的数据包数是 200，程序的流程图如图 4 所示。

6.2 程序中用到的基本数据结构

```
#define size 15 //散列表的桶数
#define packetsize 20 /* 一次读入缓冲区的数据包数，同时也是定时扫描的时间 */
#define number 20 /* number 存放源 IP 目的 IP 源端口，目的端口的数组长度 */
//各桶中同义词表结构
```

```
struct ListNode
{ char *Key1; //源 IP 地址
  char *key2; //目的 IP 地址
  char *key3; //源端口
  char *key4; //目的端口
  ListNode *link; //链指针
  int time;} //记录何时对此流进行最后一次操作
  HashTable(); //散列表的构造函数
```

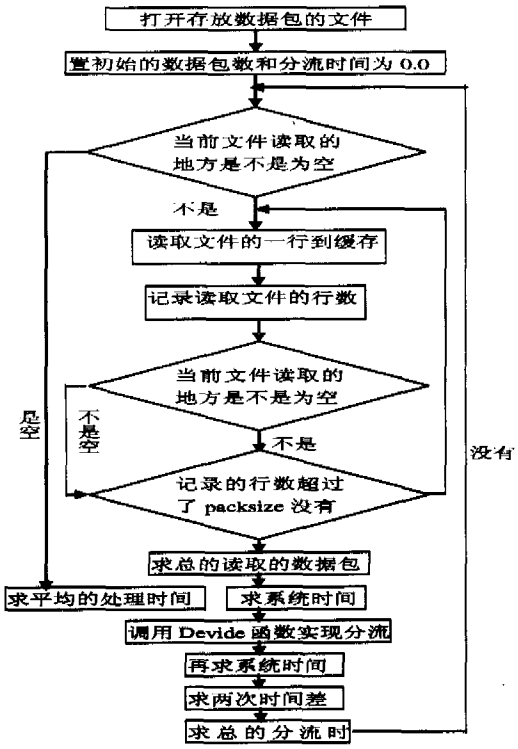

图 4　程序流程图

6.3　程序中主要自定义函数及其实现

(1)HashTable()

函数功能:对 HashTable 类的对象进行初始化。

实现方法:将全局变量中的 size 的值赋给散列表的容量,将源 IP,目的 IP,源端口,目的端口赋值为空,将处理的数据包数置为 0。

(2)char * * HashTable::Find(char *x,char *y,char *z,char *w)

函数功能:搜索整个散列表,观察散列表中是否有某个节点的源 IP,目的 IP,源端口,目的端口号分别与 x,y,z,w 相等,如果有分别相等的节点,则返回此节点的源 IP 的首地址,如果没有,则返回 0。

参数:x 是目前流的源 IP,y 是目前流的目的 IP,z 是目前流的源端口,w 是目前流的目的端口。

实现方法:先利用流的局部性原理将 x,y,z,w 和与上次操作节点的源 IP,目的 IP,源端口,目的端口分别进行比较,如果分别相等,则搜索到结果,分流成功,更改此节点的操作时间标志,同时修正上次操作节点的信息,最后返回源 IP 的首地址。如不相同,则用散列函数求出桶号,在这一桶的同义词链表中逐个比较,如找到源 IP,目的 IP,源端口,目的端口分别与 x,y,z,w 相等,则查找成功,更改此节点的操作时间标志,同时修正上次操作节点的信息,最后返回源 IP 的首地址,若将整个同义词链搜索完毕,还没有找到,则返回 0。

(3)int HashTable::HashFunc(char *x,char *y,char *z,char *w)

函数功能:求出桶号。

参数:x 是数据包的源 IP,y 是数据包的目的 IP,z 是数据包的源端口,w 是数据包的目的端口。

实现方法:函数首先将源 IP 字符串的用点号分隔的各个字段析取成整数,然后将其相加。同理可以求出目的 IP 地址各字段之和。然后将源 IP,目的 IP 相加得到的和,和源端口,目的端口字符串析取出来的整数相加,得到的和除以桶数,就得到了桶号,最后将桶号返回。

(4) void HashTable::HashInsert(char *x, char *y, char *z,char *w)

功能:插入一个节点。

参数:x 是作为插入节点的源 IP,y 是作为插入节点的目的 IP,z 是作为插入节点的源端口,w 是作为插入节点的目的端口。

实现方法:插入一个结点,如果根据求桶号的函数求出的桶号所对应的头指针还没有结点,就直接将头指针指向它该节点,如果头指针指向的已有结点,就将其插入到该同义词结点链的最后面。同时要使 Guard 指针指向刚插入的节点。

(5) void HashTable::HashDelete(ListNode *p, ListNode *q,int j)

功能:将节点删除,并将删除节点的前后节点链接起来。

参数:p 是要删除节点的前一个节点,q 是要删除的节点,j 是删除的节点所在的桶。

实现方法:如果扫描到的节点是链首节点,则将这个桶的首指针 $ht[j]$ 指向它,然后将其删除。否则则将它的前驱节点的链接指针指向它的链接指针指向的节点,然后将其删除。

(6)void HashTable::HashSearchDelete()

函数功能:从第一个桶开始扫描散列表,直到将所有桶扫描完,将扫描到的不活跃的节点删除。

实现方法:从第一个桶开始扫描,当扫描到某一个散列表的节点的记录最后一次操作的时间标志与当前的时间标志之差大于 50(50 是指当散列表已经连续处理了 50 个数据包及其以上,但中间没有处理过于该数据包属于同一流的包,则该数据包已经为已经不活跃的包)时,就调用函数 HashDelete() 将其删除,然后继续扫描,直到将散列表的所有表项全部扫描完为止。

(7)void Devide(int k,HashTable &s)

函数功能:从缓冲取中析取源 IP,目的 IP,源端口,目的端口。并调用 HashTable 类的函数以实现其分分流。并调用函数以对已经不在活跃的流删除。

参数:k 是从主函数中得到的本次读入缓存的数据包数,s 是一个 HashTable 类对象的别名。

实现方法:从第一个缓冲取开始,首先检查第一字符是否为空格,直到检查到不是空格为止,然后将第四个点号之前的字符串并在其后加上一个'/0'的字符赋值给 x,然后将第四个点号之后到空格或<符号之前的字符串加上'/0'字符赋值给 z,然后将后面在一次遇到非空格的字符开始,将其到后面第八个点号之前字符串并子后面加'/0'符号赋值给 y,并将第八个点号后面的字符直到回车符之前的字符赋值给 w,并调用 HashTable 类的 Find 函数,如未查到该,则将该流插入散列表。当处理完所有的缓冲区后,就调用函数 HashSearchDelete(),删除不活跃的流。

6.4　程序中调用的库函数及其相关说明

(1)函数 strcmp

函数原型为 int strcmp(const char *$str1$,const char *$str2$),其功能按照字典的顺序比较 $str1$ 和 $str2$ 两个字符串,

(下转第 86 页)

由此，我们认为，本节给出的基于应用区域边界安全体系结构的描述规则是合理的、安全的，因此由这些规则组成的基于应用区域边界安全体系结构模型是安全的。

结束语 本文利用 BLP 模型、Biba 模型、RBAC 模型和信息流模型的基本性质，通过对文[5]提出的应用区域边界的安全体系结构进行分析，给出了该体系结构的描述规则。经对这些规则的分析，我们认为它们是合理的、安全的，相应的模型是安全的。通过对应用区域边界的安全体系结构的模型描述和验证，将有助于推动信息安全体系结构的理论研究。

参 考 文 献

1 GB/T 9287.2-1995. 信息处理系统．开放互连基本参考模型第 2 部分：安全体系结构

2 Kent S. Security Architecture for the Internet Protocol. RFC 2401, 1998.11

3 Information Assurance Technical Framework 3.1, 2002.9. http://www.iatf.net/framework_docs/version-3_1/index.cfm

4 沈昌祥. 构造积极防御的安全保障框架[J]. 计算机安全，2003，10：1～2

5 陈兴蜀. 应用区域边界的安全体系结构及实用模型研究[M]：[学位论文]. 成都：四川大学，2004

6 Bell D E, Lapadula LJ. Secure compter system [R]: mathematical foundation. MTR-2527, Mitrecorp, Bedford, MA, 1973 (NTIS AD771543)

7 Biba K. Integrity Considerations for Secure Computing Systems [R]: [Mitre Report MTR-3153]. Mitre Corporation, Bedford, MA, 1975

8 Sandhu RS, Samarati P. Access control: principles and practice [J]. IEEE communications. 1994,32(9): 40～48

9 刘益和. B/S 模式信息安全系统的一种形式化描述[J]. 计算机科学，2004,31(9A)：217～219

(上接第 69 页)

当 *str*1 小于 *str*2 时，返回负值。当 *str*1 等于 *str*2 时，返回 0。当 *str*1 大于 *str*2 时，返回正值。在使用时应该包含文件 string.h 库函数。

(2)函数-ftime

函数原型 void -ftime(strrct -timeb * timeptr) timeptr 指向 SYS\TIME.H 中定义的结构指针。

函数-ftime 读取当前时间并将其存放到由指针 timeptr 指向的结构中。-timeb 结构在 SYS\TIMEB.H 中定义。函数-ftime 的四个域及其取值如表 1 所示。

表 1 函数-ftime 的四个域及其取值

域	取值
dsfflag	如果当地正采用夏令时，此域值是非零(参见-tzset 中对定义夏令时的解释)。
millitm	不到一秒部分的毫秒数。最后一位数字总是 0，因为 millitm 每次增加近 1%秒。
time	从 1899 年 12 月 31 日午夜(00:00:00)计算起的秒数。
timezone	通用协调时间向西与当地时间的差，以秒为单位 timezone 的值是根据全局变量-timezone(参见-tzset)的值设定的。

-ftime 函数为 timeptr 所指向的结构域赋值。它不返回值。使用它时应该包含库函数 #include〈sys\type.h〉或 #include〈sys\timeb.h〉。

(3)getline()函数

函数原型为 istream&::getline(char * pszTarget, int nCount, char delim='\n')。

实现的功能：从文件中读取一整行字文本包括空白字符。参数 pszTarget 是用于存放读取的文本的字符数组，参数 nCount 是最多读取的字符个数，delim 是作为读取结束标准的分隔符。默认的分隔符是'\n'，但是不将分隔符存入缓冲区使用时应该包含库函数 #include〈fstream.h〉。

(4)函数 strlen()

函数原型 unsigned int strlen(char 8*str*)。

函数的功能统计 *str* 中字符的个数(不包括终结符'\0'，返回字符的个数。应包括 #include〈string.h〉。

6.5 时间和空间复杂度分析

算法的执行时间分为两个部分，散列计算的时间(记为 $T_{计算}$)和散列查找的时间(记为 $T_{查找}$)两个部分，所以算法的执行时间为 $T_{计算}+T_{查找}$。通常计算的时间要远小于访存的时间，也就是查找的时间。所以算法的时间约等于 $T_{查找}$，而散列查找的时间 T_s 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 $N/2M$，所以假设流的数目是 N，散列表的基本区大小是 M，散列查找的平均次数是 $N/2M$，所以算法的时间复杂度是 $O(N/M)$。同时我们也指出，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 U_{Hash}。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 N，且每个流的记录要占用 K 个字节，则 U_{Hash} 要占用的存储空间是 $K\times N$，即 $O(N)$。所以算法所占用的存储空间大约是 $O(N)$。

结论 程序的运行结果根据不同的仿真环境，结果有所不同。影响结果的因素有，仿真的硬件环境(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。本算法中，得到的结果是 0.0003 秒。即每 0.0003 秒处理一个数据包，每秒处理 3333 个数据包，每个数据包的平均长度是 36bit，则数据流量是 120kbit/s。

参 考 文 献

1 Algorithms for Packet Classification. http://itpapers.zdnet.com

2 Packet Classification Repository. http://www.ial.ucsd.edu/classification/

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

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

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

6 Flow classification. http://www.hifn.com/technology/Classification.html