

一种高效的业务流分类算法

杨建华¹, 谢高岗¹, 张广兴^{1,2}, 李忠诚¹

(1. 中国科学院计算技术研究所, 北京 100080; 2. 湖南大学计算机与通信学院, 湖南长沙 410082)

摘 要: 通过分析单链路监测的流量特征和业务流监测分析需求, 提出了一种高效业务流分类算法. 算法把业务流分类过程分为三个阶段: 第一阶段采用 Hash 方法, 尽量分散流记录的分布; 第二阶段可采用两种方式, 一种是线性链表, 另一种是查找树; 最后一个阶段是线性查找. 分别在两种不同型号的流量监测系统上实现了该算法, 实验结果显示, 当链表长度为 300 时, 查找不成功并创建新的流记录和查找成功并更新流记录的时间分别为 1.8 μ s 和 1.3 μ s, 相应的业务流分类能力为 55 万和 77 万 pps.

关键词: 业务流监测; 流分类; 算法

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2006) 03-0549-04

An Efficient Algorithm for Flow Classification

YANG Jian-hua¹, XIE Gao-gang¹, ZHANG Guang-xing^{2,1}, LI Zhong-cheng¹

(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China)

(2. College of Computer and Communication Hunan University, Changsha, Hunan 410080, China)

Abstract After analysis of requirements of traffic flow monitoring an efficient accurate flow classification algorithm is proposed. The algorithm divide the flow classification process into three phases and each phases reduces the dimensions of flow classification. The experience shows that when the list table length equals 300, the classification process time for successful searching and updating or unsuccessful searching and new record inserting is 1.8 μ s and 1.3 μ s. And the classification ability is 0.55 and 0.77 million packets per second.

Key words traffic flow measurement; flow classification; algorithm

1 引言

随着网络技术的不断发展, 网络结构越来越复杂, 所包含的网络节点越来越多, 网络通信速率不断提高, 并且各种新的网络应用, 包括数据、语音、多媒体等层出不穷, 因此详细了解网络流量特征, 包括每个数据包的传输特征、数据包交换特征以及业务流特征对于网络管理、运维以及科学管理、甚至网络技术的持续发展都非常重要, 特别是对于大型复杂的 Internet 骨干更为重要^[1].

业务流分类方法属于包分类方法的一种. 包分类技术被广泛应用于防火墙的接入控制、IP 路由、基于策略的路由、区分服务及网络计费等方面. 业务流分类和上面提到的包分类技术略有不同: 首先, 几乎所有文献中提到的包分类方法都需要一个初始的规则库, 而业务流分类不需要初始规则库, 其规则根据网络中实际网络流量进行创建、更新和老化; 其次, 包分类所使用的规则库需要通过手工

输入或者路由信息交换完成, 其更新频率相对较小, 而业务流分类的规则库 (即为网络中业务流记录信息库) 更新频率相对较大, 每增加一个新的业务流或者老化一个业务流都需要对规则库进行更新; 其次, 由于包分类用途的限制和对灵活性的要求, 其规则大都支持基于前缀、范围以及通配符的匹配, 而业务流分类则主要根据业务流的定义进行精确查找、匹配; 最后, 包分类技术的每条规则对应一个操作, 如转发、过滤等, 如果分类不成功, 即没有发现匹配的规则, 则或者丢弃或者按照缺省规则转发数据包; 而业务流分类如果查找不成功, 并且没有达到系统所支持的最大并发业务流数目, 则创建并插入新业务流记录, 如果达到了系统所支持最大并发业务流数目, 则丢弃或者只进行基本协议信息统计, 如果查找成功则更新业务流记录的大小以及记录的最后更新时间. 另外, 已有包分类技术比较侧重于查找效率和存储空间考虑, 而业务流分类则对查找效率和规则更新的支持程度同等重要. 综上, 现有的

包分类方法不太适合业务流监测分析,因此根据业务流监测的特殊性和网络业务流的特征,研究高效的业务流分类方法对实现高速业务流监测分析系统非常重要。

本文通过分析现有包分类方法的特点,结合网络业务流监测分析的特点和需求,提出了一种高效的业务流分类方法,方法把业务流分类过程分为三个阶段:第一阶段采用 Hash 方法,尽量分散流记录的分布;第二阶段可采用两种方式,一种是线性链表,另一种是查找树;最后一个阶段是线性查找。算法性能取决于业务流分类过程中的链表长度,当第二阶段采用线性链表时,链表长度和分类所用时间为线性关系,即平均链表长度越长可能的分类时间越长,性能越低。我们分别在两种不同型号的流量监测系统(NePro100和 NePro3000)上实现了该算法,实验结果显示,当链表长度为 300 时,查找不成功并创建新的流记录和查找成功并更新流记录的时间分别为 $1.8\mu\text{s}$ 和 $1.3\mu\text{s}$,相应的业务流分类能力分别为 55 万和 77 万 pps

2 业务流分类方法需求

业务流分类方法是实现基于业务流的流量监测的关键技术,早在 20 世纪 90 年代就开始了基于业务流的流量监测研究,如 IETF 的 RTFM (Realtime Traffic Flow Measurement) 工作组和 CAIDA 组织^[2]等。业务流分类方法和现有的包分类算法中的多维包分类非常类似。但是现有包分类算法大都针对策略路由、防火墙、入侵监测、4 层交换、QoS 等应用设计,而业务流分类算法主要需要完成基于业务流的流量监测分析。

通过对业务流监测流程的分析,根据业务流的特征,要实现基于业务流的流量监测,业务流分类方法必须满足如下要求:

- (1) 查找速度快,随着网络链路速度的提高,业务流分类必须具有较高的匹配速度。
- (2) 不需要初始业务流记录,即没有初始规则库,避免过多的预计算。
- (3) 动态性好,能够支持快速的业务流记录插入和删除。
- (4) 易于实现,方法应便于采用软件和硬件的方式进行实现,要便于采用流水线结构和并行逻辑进行实现。
- (5) 支持良好的精确匹配,业务流监测中的分类只需要精确匹配,实现不同主机之间的不同应用业务流的精确统计。
- (6) 支持大容量并发业务流,如百万条以上。

当前常用的几个典型的多维包分类算法有 RFC (Recursive Flow Classification)^[3]、GOT (Grid of Tries)^[4]、HICu (Hierarchical Intelligent Cuttings)^[5]、ABV (Aggregate Bit Vector)^[6]、TSS (Tuple Space Search)^[7] 等。RFC 算法必须事先计算 ClassID 并创建缩减树,对规则增加或者删除,必须重新计算 ClassID 并创建缩减树,因此不适合

进行业务流监测分析中的业务流分类。GOT 为了提高查找效率必须通过预计算的方式存储中间转向指针信息,但是过多的预处理和转向指针,使得该方法动态性差,增加或者减少规则需要对整个树进行重建。另外 GOT 比较适合二维包分类,不太适合多维包分类(如 5 维业务流分类)情况。对于 HICu 算法,在规则空间均匀分布的情况下有很好的性能,但由于构造 HICuts 树时是循环依次对每一维进行空间划分,如果一个 d 维分类器中的大部分规则只通过某一维来划分,而其他维的值相似或相同, HICuts 树的深度和结点会大大增加,预处理时间和占用的内存空间都会成倍增加,大大影响算法的性能。同样的,ABV 的预处理阶段需要对分类器中的规则按字段进行排序,但这会大大增加预处理时间。由于设计的最初目的, TSS 只支持前缀匹配,并且 Hash 算法的使用使得匹配和更新的时间具有不确定性。

综上,已有的包分类方法或者不能直接应用于业务流分类,或者存在更新性能较低的缺陷,因此研究专门的业务流分类算法对实现高效的业务流监测非常重要。

3 业务流分类算法设计

3.1 基本数据结构

本文设计的业务流分类算法采用三阶段查找方法,第一阶段采用 Hash 方法,尽量分散流记录的分布;第二阶段提供两种方式,一种是线性链表,另一种是查找树;最后一个阶段是线性查找。第一阶段 Hash 表的大小为 255×255 算法关键在于 Hash 函数的选择,即 Hash 索引值的计算方法。

对于一条被监测链路,链路两端分别对应一组用户群或者服务器群,根据 IPv4 地址分配原则, IPv4 地址的最后一个字节分布最广泛也最分散,根据此特点,设计 Hash 函数如下。对于一条被监测链路,选择一端为内部,另一端为外部,因此流量可以被划分为出境和入境。如果捕获的数据包为出境流量,则选择数据包的源 IP 地址的最后 1 个字节作为低 8 位,目的 IP 地址的最后一个字节为高 8 位组合而成的一个 16 位数 hash 索引值,如式 1 所示。

$$\text{Hash_index} = (\text{src_ip4}) \ll 8 + (\text{dst_ip4}) \quad (1)$$

其中 src_ip4 dst_ip4 分别表示源、目的 IP 地址的最后一个字节。

反之如果为入境流量则把数据表的目的 IP 地址的最后一个字节作为低 8 位,源 IP 地址的最后一个字节作为高 8 位组成 hash 索引值。如式 2

$$\text{Hash_index} = (\text{dst_ip4}) \ll 8 + (\text{src_ip4}) \quad (2)$$

第二阶段可选用线性链表方式,每个 Hash 表项指向一个 IP 地址对结点链表,即所有 Hash 索引值相同的 IP 地址对结点形成一个线性链表。每个 IP 地址对结点则包含一个 Flow 记录链表,这些 Flow 记录具有相同的 IP 地址对。流记录的查找和存储采用的数据结构如图 1 所示。

第二阶段也可使用查找树, 则 *Hash* 表项指向的是一个由 *IP* 地址对结点组成的树. 树结构为有序二叉树, 树中结点进行排序的关键值 T_key 通过如下方法构造. 对于出境数据包, 用源 *IP* 地址的倒数第二个字节构造的 1~4 字节和 9~12 字节, 目的 *P* 地址的倒数第二个字节作为后 8 位构造 T_key 的 5~8 字节和 13~16 字节, 如式 3 所示.

$$T_key = (src_ip \& 0x0f) \ll 12 + (dst_ip \& 0x0f) \ll 8 + (src_ip \& 0xf0) + (dst_ip \& 0xf0) \gg 4 \quad (3)$$

如果为入境数据包, 则用目的 *IP* 地址的倒数第二个字节构造 T_key 的 1~4 字节和 9~12 字节, 源 *IP* 地址的倒数第二个字节作为后 8 位构造 T_key 的 5~8 字节和 13~16 字节, 如公式 4

$$T_key = (dst_ip \& 0x0f) \ll 12 + (src_ip \& 0x0f) \ll 8 + (dst_ip \& 0xf0) + (src_ip \& 0xf0) \gg 4 \quad (4)$$

T_Key 较大的结点作为右子结点, 较小的为左结点, 相等的作为中间结点. 任何一个 *IPPair* 结点的左右子结点是一个子树, 而中间结点则串成一个线性链表. 具体采用的数据结构如图 2 所示.

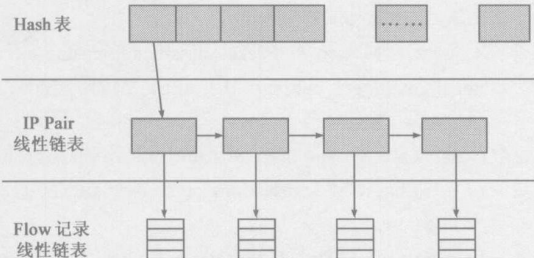


图 1 分阶段流分类算法数据结构示意图, 第二阶段采用线性表结构

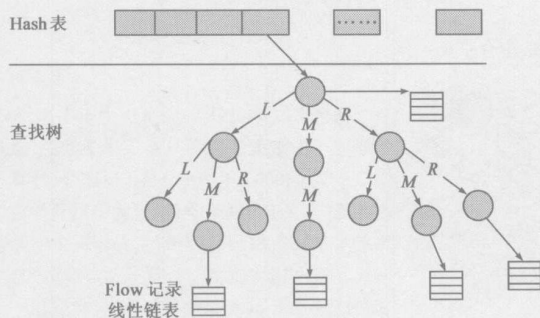


图 2 分阶段流分类算法数据结构示意图, 第二阶段采用 Trie 结构

3.3 查找与更新

根据 3.1 小节提出的业务流分类数据结构, 业务流分类算法包括如下四个部分: 查找流记录、插入/更新 *IPPair* 结点, 插入/更新流记录结点, 流记录老化与信息统计. 由于算法针对基于业务流的流量监测设计, 查找与更新同时并存, 即如果查找成功则更新流记录信息, 否则插入新的

流记录结点. 算法如图 3 所示.

```
Search(pkt, h, pkt)
{
    //extract src_ip, dst_ip, sport, dstoport, protocol
    p = extract(pkt);
    Hash_index = hash(p);
    // if search ippair node successful
    if((ippairmode = SearchIPpair(Hash_index, p)) != NULL){
        if search flow record successful
        if(flow record = Searchflow(p, ippairmode)) != NULL
            Updateflowinfo(p, flow record)
        else
            Insertnewflow(p)
    }
    else{
        Insertnewippair(p);
        Insertnewflow(p)
    }
}
```

图 3 分阶段业务流分类算法-查找与更新算法伪码

如果第二阶段采用的是线性表, 则查找链表是顺序查找, 每次插入新的 *IPPair* 结点均插入在相应 *Hash* 表指向链表的第一个结点, 新的流记录信息也插入在相应 *IPPair* 结点指向的流记录链表的第一个结点; 若第二阶段采用的是 *Trie* 结构, 则在查找到一个 *IPPair* 结点后首先计算 T_Key , 并根据 T_Key 大小决定在 *IPPair* 结点的左子树、右子树或者中间子树中继续查找知道查到 T_Key 相同结点.

4 测试实验与分析

将该流监测算法在网络流量监测系统探针 *NetPro* 中实现. 测试实验环境如图 4 所示.

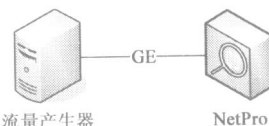


图 4 测试实验环境

流量产生器与 *NetPro* 通过 *GE* 链路直接相连, 流量产生器生成探测数据包, *NetPro* 对捕获的数据包进行 *MAC-PTCP/UDP Application* 逐层协议解析, 更新并输出流指标. 对每个数据包, *NetPro* 在处理捕获的数据包之前记录时刻 t_1 , 完成协议解析并进入流指标分析记录时刻 t_2 , 完成流指标分析记录时刻 t_3 , 则数据包在 *NetPro* 中分析总共花费时间为

$$T_1 = t_3 - t_1 \quad (5)$$

流指标分析时间为:

$$T_2 = t_3 - t_2 \quad (6)$$

则 *NetPro* 流量监测分析的吞吐量为:

$$p = \frac{1}{T_1} = \frac{1}{t_3 - t_1} \quad (7)$$

其中 \bar{T}_i 表示数据包在 *NetPro* 处理时间 T_i 平均值。

测试 1 由流量产生器产生 IP 地址对完全不同的 350 组数据包, 用以测试查找链表不成功并创建新记录节点时的处理时间; 测试 2 是在测试 1 的基础上产生同样的数据包序列, 用于测试查找链表成功并更新记录信息的处理时间。测试结果如表 1 所示。

假设 T_i 和 T'_i 表示链表查找成功和不成功时的业务流分析时间, 则链表长度和 T_i 、 T'_i 的关系如图 5 所示, 由图可知:

(1) 流分析过程, 特别是流链表查找, 是流量监测分析主要耗时过程, 提高流分析速度将提高流量监测分析的吞吐量;

(2) 插入流链表记录的流分析过程, 数据包处理耗时大于已建立链表流分析过程, 数据包平均处理时间较已建链表大 0.5 μ s

为了验证链表长度和流分析时间的关系, 分别对链表长度为 100 200 300 进行了多次测量, 表 2 为实验结果。

实验结果分析可知, 当链表长度为 100 时, 进行数据包分类所需的平均处理时间分别为 0.78 μ s (查找不成功, 需要插入新的业务流记录) 和 0.2 μ s (查找成功, 更新业务流记录所需时间)。此时, 系统可以达到的理论包处理速度分别约为 128 万 pps 和 500 万 pps

5 结论

通过分析业务流监测需求, 结合单链路监测的流量特征, 提出了一种高效、精确无源业务流分类算法。算法中采用的业务流定义为经典 5 元组方式, 业务流结束判断方法为固定超时时间方法, 超时时间为 60 秒。采用该算法的业务流分类过程分为三个阶段, 第一阶段采用 Hash 方法, 尽量分散流记录的分布; 第二阶段提供两种方式, 一种是线性链表, 另一种是查找树; 最后一个阶段是线性查找。算法实现简单, 更新方便。算法性能取决于业务流分类过程中的链表长度, 当第二阶段采用线性链表时, 链表长度和分类所用时间为线性关系, 即平均链表长度越长可能的分类时间越长, 性能越低。

该算法应用于 *NetPro* 3000 以及 *NetPro* 100 网络监测探针中, 实验结果显示, 当链表长度为 300 时, 单个数据包查找不成功后插入新流记录和查找成功并更新流记录信息的平均处理时间分别为 1.8 μ s 和 1.3 μ s, 对应的理论数据包处理能力可以分别达到 555555 6pps 和 769230 8pps

参考文献:

[1] A Fekmann, A Greenberg, C Lund, N Reingold, J Rex-

表 1 查找成功和不成功时的流分析时间比较表, 单位: μ s

测量序号	描述	样本数量	最小处理时间	最大处理时间	平均处理时间	标准差
测试 1	查找不成功并插入链表结点	350	0.773333	7.210667	1.82684952	0.626975207
测试 2	查找成功并更新流记录	350	0.261333	7.205333	1.29273905	0.661057906

表 2 链表长度与流分析时间表, 单位: μ s

链表长度	样本数量	最小处理时间	最大处理时间	平均处理时间	标准差
100	2000	0.768000	2.640000	0.77932000	0.116005242
200	2000	1.301333	3.248000	1.31158667	0.108805490
300	2000	1.834667	7.626667	1.89532000	0.322129976

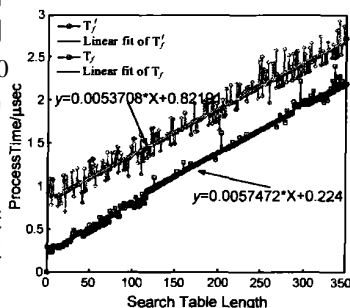


图 5 流分析时间与链表长度关系

for F-Tune Deriving traffic demands for operational IP networks Methodology and experience [J]. IEEE/ACM Transactions on Networking 2001, 9(3): 265-279.

- [2] CAIDA. <http://www.caida.org/Z/OL>.
- [3] P Gupta, N McKeown. Packet classification on multiple fields [A]. Proceedings of ACM SIGCOMM [C]. Massachusetts USA, 1999. 146-160.
- [4] V Srinivasan. Fast and Efficient Internet Lookups [D]. USA: Washington University, 1999.
- [5] P Gupta, N McKeown. Packet classification using hierarchical intelligent cuttings [J]. IEEE/ACM Transactions on Networking 2000, 20(1): 34-41.
- [6] F Baboescu, G Varghese. Scalable packet classification [J]. IEEE/ACM Transactions on Networking 2005, 13(1): 2-14.
- [7] V Srinivasan, S Suri, G Varghese. Packet classification using tuple space search [A]. Proceedings of ACM SIGCOMM [C]. Massachusetts USA, 1999. 135-146.

作者简介:



杨建华 女, 1978年2月出生于山东, 2000年毕业于山东大学数学与系统科学学院, 获理学学士学位, 2005年毕业于中国科学院计算技术研究所, 获工学博士学位, 现为中国科学院计算技术研究所助理研究员, 主要研究方向为新一代互联网网络测量与流量分析、网络流量管理和控制等。E-mail: jhyang@ict.ac.cn



谢高岗 男, 1974年7月出生于浙江, 现任中国科学院计算技术研究所副研究员, 硕士生导师, 主要研究方向为新一代互联网可信技术的研究和开发等。