流量自适应的多维度包分类方法研究

万云凯[1)] 嵩 天[1)] 刘苗苗[1)] 刘 翼[1),2)] 李 丹[3)]

[1)](北京理工大学计算机学院 北京 100081)

[2)](延安大学网络信息中心 陕西 延安 716000)

[3)](清华大学计算机科学与技术系 北京 100084)

摘 要 以软件定义网络为代表的新兴网络技术发展使得包分类不再局限于传统的五元组而是面向更多维度,以开放协议 OpenFlow 为例,网络包分类需要针对几十个维度,并且维度数量在持续增加,这种多维度包分类功能成为了软件定义网络实际应用中的性能瓶颈.该文分析了五元组包分类算法向更多维度扩展的局限性,利用网络流的局部性原理,提出了一种流量自适应的多维度包分类方法.该方法可以根据网络流量的实时分类结果动态调整多维度匹配顺序,优先匹配当前流量所需要的字段,通过忽略通配字段达到优化查找速度的目的.同时,该方法将多维度字段分组,结合具体字段类型选择最优匹配算法.在 Open vSwitch 中增加所提出方法,实验结果表明,该方法相比已有的包分类算法在用户态模式下性能提高约 2 倍,相比从五元组包分类算法扩展的方法性能提高 40%以上.

关键词 包分类;软件定义网络;流量自适应;位向量;OpenFlow

中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2017.01543

A Flow Adaptive Multi-Dimensional Packet Classification Algorithm

WAN Yun-Kai[1)] SONG Tian[1)] LIU Miao-Miao[1)] LIU Yi[1),2)] LI Dan[3)]

[1)](*School of Computer Science, Beijing Institute of Technology, Beijing* 100081)

[2)](*Network and Information Center, Yanan University, Yan'an, Shaanxi* 716000)

[3)](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract The most important function of the data plane in software defined network (SDN) is to classify packets by using tens of packet header fields, namely multi-dimensional packet classification, which is highly extended from the most commonly used five-tuple fields in the contemporary packet classification. The number of dimensions is still increasing with the development of SDN. In this paper, we analyzed the drawbacks of the classification algorithms directly extended from five-tuple packet classification and surveyed the existed algorithms used in practical systems, such as Open vSwitch. Then we presented a flow adaptive algorithm based on bit vector for multi-dimensional packet classification, especially designed for tens of header fields. This algorithm first classifies packet against each header field separately, correlates them and optimizes the search speed by dynamically re-order different fields, and then intentionally skips some wildcard fields according to the locality of traffic flow. The packet classification on different header fields may exploit specific design algorithm according to different matching methodologies of header fields. Experimental results on the Open vSwitch platform, which is an implementation of OpenFlow protocol in SDN, show that the proposed algorithm achieves about two times speedup

收稿日期:2015-08-25;在线出版日期:2016-07-18.本课题得到国家自然科学基金(61272510,61432002,61522205)和陕西省教育厅科学研究项目(14JK1825)资助.**万云凯**,男,1990 年生,硕士,主要研究方向为互联网体系结构和软件定义网络.**嵩 天**(通信作者),男,1980 年生,博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为网络信息安全、互联网体系结构和微处理器设计.E-mail:songtian@bit.edu.cn.**刘苗苗**,女,1992 年生,硕士研究生,主要研究方向为互联网体系结构和下一代互联网.**刘 翼**,男,1982 年生,博士研究生,工程师,中国计算机学会(CCF)会员,主要研究方向为互联网流量分析与建模、深度数据包检测和网络流量管理、网络安全.**李 丹**,男,1981 年生,博士,副教授,主要研究方向为互联网体系结构、数据中心网络、软件定义网络和云计算.

in user mode than the current algorithm in Open vSwitch, and over 40% speedup than other algorithms directly extended from five-tuple classifications.

Keywords packet classification; software defined network; flow adaptable; bit vector; OpenFlow

1 引 言

软件定义网络(Software Defined Network,SDN)已经成为互联网及数据中心网络的研究热点,其开放协议 OpenFlow[1]通过将网络设备控制平面和数据平面分离实现网络流量的灵活控制,为核心网络及应用创新提供了良好平台. OpenFlow 交换机作为 OpenFlow 网络的核心部件,其主要功能是通过预定义的规则对流量进行分类,并通过流表信息转发网络包. 这是一个网络包分类过程,即通过识别网络包包头一个或多个字段(Field)信息,将其匹配到预定义的规则集中,然后根据匹配规则对应的决策(Action)对网络包进行相应处理.

近年来,随着网络业务和网络安全需求的不断增长,软件定义网络及 OpenFlow 协议受到越来越多的关注,广泛部署在数据中心等实际环境中. 不同于应用在防火墙(Firewall)[2]、服务质量(Quality of Service)[3-4]、策略路由(Policy Routing)[5]、流量计费(Traffic Billing)[6]等领域基于五元组(源 IP 地址、目的 IP 地址、源端口号、目的端口号、协议类型)的传统包分类方法,"OpenFlow 包分类"(本文用这个名词代表 SDN 中的包分类)面临如下挑战:

(1) OpenFlow 包分类算法不再局限于五元组字段,而是扩展至十二甚至更多字段. OpenFlow 的这种包分类算法相较于五元组分类算法,在性能、存储和可扩展性等方面的需求更具挑战.

(2) 随着软件定义网络应用的进一步发展,包分类规则将从人工定义逐步过渡至由控制器自动生成,随之而来的规则数量将持续增加,从目前广泛使用的几千条可以发展至上万条甚至更多.

(3) OpenFlow 包分类通常需要处理 10 Gbps、40 Gbps 及 100 Gbps 等万兆流量,并要求达到线速的处理能力以应对网络突发流量情况,这需要核心包分类算法具备很高的处理性能.

针对 OpenFlow 包分类方法的分类字段多、规则数量多和高速处理等要求,本文提出了一种能够自适应流量的多维度包分类方法,主要贡献如下:

(1) 分析总结了 OpenFlow 包分类的特点,提出了适用于 Openflow 包分类算法的指导性原则.

(2) 提出了一种适用于 OpenFlow 多字段特点的流量自适应包分类算法. 该算法基于位向量设计,对每个字段进行单独匹配,生成相应的位向量,然后进行合并. 同时,该方法利用流量自适应思想,将分类字段根据局部性使用情况进行排序,优先匹配适应于局部性流量的字段从而避免匹配剩余通配字段,实现提高匹配速度的目的. 针对字段类型不同的特点,选取不同匹配算法,配置更加灵活.

(3) 在开源 Open vSwitch 系统中实现了本文提出的方法,并测试了该方法性能. 实验表明,该算法的包分类速度快于系统已有的包分类算法和由五元组扩展而来的包分类算法.

本文第 2 节介绍传统包分类算法和网络流的局部特性;第 3 节介绍 OpenFlow 环境下包分类方法的特点;第 4 节描述本文提出包分类算法的主要思想和详细设计;第 5 节对算法进行实验验证;第 6 节总结全文.

2 相关工作

2.1 包分类问题及方法

2.1.1 包分类问题的描述

在互联网模型[7]中,待传输数据被各层协议包头依次封装,每层包头都包含若干字段,这些字段分别携带该层协议特征. 包分类即根据网络包的包头字段与预定义规则集中各条规则的各个字段进行匹配,将网络包分配到相应规则集中.

包分类问题可以描述如下:规则集 C 中有 n 条分类规则,共涉及 k 个字段,每条规则 R 至多包含 k 个字段,每个字段表示为 R_i,其中 $i=1,\cdots,k$;网络包头 H 中对应字段表示为 H_i;R_i 和 H_i 之间存在如下几种匹配方式,分别为:

(1) 精确匹配. R_i 和 H_i 表示为具体数值或字符串,当 $H_i=R_i$ 时,R_i 与 H_i 精确匹配.

(2) 前缀匹配. R_i 和 H_i 表示为字符串,当 R_i 是 H_i 在字符串意义上的前缀时,H_i 与 R_i 前缀匹配,其中,最长前缀匹配是前缀匹配的特殊情况.

(3) 范围匹配. R_i 是一个数值范围区间,即 $R_i=$

$[val_1, val_2]$，H_i表示为具体数值，当$H_i=[val_1, val_2)$时，H_i与R_i范围匹配.

(4)通配符匹配. R_i是一个通配符，可以匹配任何 H_i，H_i与R_i通配符匹配.

当且仅当网络包头 H 中的每个字段 H_i都能与规则 R 中对应字段R_i通过对应方式形成匹配时，则称网络包头 H 与规则 R 相匹配.

在分类规则中，除各字段的信息之外，还包括规则决策(Action)和规则优先级(Priority). 规则决策用来表示对匹配该条规则数据包的后续处理操作，如丢弃(Drop)、重置(Reset)连接、拒绝(Deny)、转发(Forward)等；规则优先级表示当一个包可能与多条规则相匹配时，选择匹配规则的优先关系.

2.1.2　经典包分类算法

传统网络应用依靠目的 IP 地址来决定数据包在路由器的下一跳地址[8-9]. 随着网络结构复杂化和网络服务多元化的趋势，网络包分类技术逐渐应用于更高级别的网络功能. 网络包分类技术能够依靠网络包头中的五元组对网络流量进行细粒度分类，从而提高网络管理水平. 包分类技术既可以在硬件上实现，也可以在软件上实现. 其中，最常使用的硬件算法是基于三态内容寻址存储器(TCAM)[10]的包分类算法，该算法通过保存字段掩码的方式存储任意长度的表项，既可用于精确匹配又可用于前缀匹配，而且速度快、实现相对简单，但 TCAM 造价昂贵、功耗大的缺点限制了它的使用范围. 基于软件的包分类算法相比硬件算法应用更加灵活.

包分类算法中的每一个字段又被称为一个维度(dimension)，软件包分类算法大体上可以分为两类：多维联合分类算法和单维组合分类算法[11].

多维联合分类算法主要应用在传统五元组包分类中，主要包括基于空间划分的算法和基于哈希(Hash)的算法等. 基于空间划分的算法将包分类问题转化为数学几何中多维空间的点定位问题，将一个 d 维规则 R 理解为 d 维空间的一个超立方体，而网络包头 H 则对应空间中的一个点，若 H 落入 R 所代表的超立方体中时，H 即与 R 匹配[12]，这类方法的典型代表是 HiCuts[13]算法、HyperCuts[14]算法、HIC[15]算法和 SAX_PAC[16]算法. HiCuts 算法首先将所有字段都转化为范围匹配，采用多级空间分解方式在每一个字段上把规则集划分为多个较小的规则子集，然后根据规则子集的划分建立一棵决策树，规则子集存放在决策树的叶子结点上，而在内部结点上则存储分类导向信息. 整个算法结合了决策树和线性查找两种分类方式，当数据包到来之后，首先沿着决策树遍历至某一叶子结点，然后与该叶子结点上的规则进行线性匹配. HyperCuts 算法是 HiCuts 算法的改进算法，通过把一些优先级较低的规则如通配规则或前缀较短的规则存储在根结点中来避免叶子结点含有较多的重复规则；另一方面，通过对多个字段同时切割来减少切割次数，从而降低决策树的深度，提高查找速度. HIC(Hybrid Intelligent Cuttings)算法进行了进一步优化，根据切分域的特征，分别在 IP 域和端口域采用比特位切分和精确投影点切分来划分规则集. SAX_PAC 算法提出了一种新奇的划分方式，根据顺序独立分类器的特性对规则集进行分解，将规则划分为多个顺序独立分类器的子集，每个子集中规则不重叠，每个子集按照包含最多域的顺序独立分类器进行划分，然后根据子集分类器相关域并行查询每个子集，对每个子集的输出查询剩余域的值，输出完全匹配的规则.

基于哈希的多维联合分类算法典型代表是元组空间查找(Tuple Space Search，TSS)[17]算法、改进的 TSS 算法[18]和 D^2BS(Dynamic Discrete Bit Selection)[19]算法. TSS 算法先将所有规则按照各字段前缀长度的组合划分成比规则数目小的元组集合，然后在这些元组里串联进行哈希查找. OpenFlow 的一个开源实现 Open vSwitch 采用了改进的 TSS 算法[18]. D^2BS[19]算法提出在多个域中通过启发式动态选择区分位，然后根据区分位划分规则集，该方法要求每个域的长度固定且位置不可变，在扩展性方面较弱. 多维联合分类算法的共同特点是不单独地考虑每个字段内部特点，而是简单地把包头的所有字段看作一个维度，进行联合查找. 因此，当包分类字段增多时，字段间的组合情况会显著增加，组合间线性查找使得这些算法的性能会严重下降.

单维组合分类算法的主要设计思想是单独地对数据包每个字段进行匹配，并对每个字段的匹配结果进行合并从而找到最终匹配的规则. 其中典型的算法有递归流分类(Recursive Flow Classification，RFC)[20]算法和位向量(Bit Vector，BV)算法[21-22]. RFC 算法是一个多阶段的分解-合并过程，通过对规则集多步映射并优化来达到快速分类的目的，其中每一阶段映射称为一次缩减(Reduction)，由阶段映射构建的数据结构称为缩减树，但它对包头的划分和字段并没有直接关系，一个字段可能被划分成多个块(Chunk)，一个块可能包含多个字段，另外，该算法的性能受到阶段

《计算机学报》

数目和阶段选取方式的制约,而且算法消耗的内存空间会随规则集的扩大和包头位数的增多急剧增加,容易产生内存爆炸.

位向量(Bit Vector,BV)算法[21-22],对于规则集中的 n 条规则,首先将这 n 条规则按照优先级进行排序,然后给每个字段定义一个 n 位的向量,每位对应一条规则,按照优先级从低到高地址依次递增.向量初始值为"0",当每个字段匹配后,被匹配规则对应的向量位置改为"1",然后,将各字段的向量之间进行"与操作"获得最后的匹配结果.其中最高地址"1"位对应的规则具有最高优先级,即为最佳匹配规则.例如,表 1 按优先级从高到低将二维规则集进行排列,两个维度都采用前缀匹配,图 1 根据表 1 建立了位向量图.假设此时一条包头字段为(110,011)的数据包到来,它在 F_1 字段的匹配结果是"001101"(6 个规则的匹配结果向量),在 F_2 字段的匹配结果是"011100",两个向量进行与运算,结果为"001100",即 R_2 与 R_3 为匹配结果,其中 R_2 优先级高于 R_3,最终匹配结果为 R_2.

表 1 位向量包分类算法规则举例

规则\维度	F_1	F_2
R_0	00*	00*
R_1	0*	01*
R_2	1*	0*
R_3	11*	0*
R_4	0*	1*
R_5	*	1*

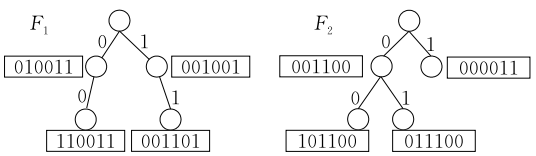


图 1 位向量包分类算法举例

单维 BV 算法有两个主要优点:(1)它具有良好的可扩展性和可并行性,因为匹配工作在每个字段上单独进行,所以字段增加并不会显著影响性能;(2)它可以充分利用各个字段特点,针对不同字段的需求采用不同且高效匹配算法,从而提高整体包分类算法的速度.由于 OpenFlow 包分类字段数量大于普通的包分类问题,字段匹配方式更加多样,因此使用 BV 算法[23-24]进行 OpenFlow 包分类相比其他算法有着明显优势.但 BV 算法乃至大部分单维组合包分类算法的一个缺点是:包分类过程需要对所有字段进行匹配,而规则集中大部分流表规则一般只包括少数有效字段,其他字段均为通配符,相比直接匹配较少的有效字段,单维组合包分类算法在时间上具有较大"浪费".

2.2 网络流量的局部性特点

一段时间内网络中具有相同包头字段(例如五元组)的数据包被称为一个流(Flow).研究表明,在现实网络中少部分流占用着绝大部分的网络流量,例如,网络中 9% 的流所含网络包数占总网络包数的 86.5%,字节数占总字节数的 90.5%[25].这些占绝大部分网络流量的少数流被称作大象流(elephant flows).在一些大型流量应用中,如 P2P 文件共享、网络视频等,大象流所携带的网络包数量极大且持续时间很长;相应地,剩下的携带很少网络包且持续时间很短的网络流被称作小鼠流(mice flows),例如,访问一个 HTTP 页面会产生访问图片、视频和文本等不同元素的并发访问流.

大象流和小鼠流这种数量和字节数上严重不平衡的现象被称为流的不平衡现象.这种现象在一定程度上造成网络流的局部特性:即在较短时间内通过同一个包分类设备的网络包有极大可能性属于少数几个网络流.因此,当网络流前面的网络包成功匹配某些规则后,后面的网络包也倾向于匹配这些规则以及这些规则所包含的字段.

基于网络流量的这个局部性特点,本文针对 OpenFlow 包分类设计了高速多维度包分类方法.深入方法之前,先介绍一下 OpenFlow 包分类问题.

3 OpenFlow 包分类

软件定义网络作为一种新型网络构架,其目的是通过分离网络控制和物理网络拓扑,从而摆脱硬件对网络架构的限制,实现网络流量的灵活控制.OpenFlow 作为软件定义网络控制平面与数据平面间的通信协议之一,因其良好的灵活性和规范性,逐渐成为 SDN 实际上的协议标准.OpenFlow 网络由 3 部分组成:OpenFlow 交换机、网络虚拟化层和控制器.控制器负责网络的集中控制,网络虚拟化层负责对网络进行虚拟化,而 OpenFlow 交换机是整个 OpenFlow 网络的核心,它由流表(FlowTable)、安全通道(Secure Channel)和 OpenFlow 协议(OpenFlow Protocol)3 部分组成,其主要功能是根据流表分类和转发网络包.

OpenFlow 流表支持包括输入端口、源 MAC 地址、目的 MAC 地址、以太网类型、VLAN ID、VLAN

优先级、源 IP 地址、目的 IP 地址、IP 协议、IP TOS 位、TCP/UDP 源端口和 TCP/UDP 目的端口在内的 12 个字段①，如表 2 所示，这些字段长度不等，匹配方式也有所区别. 因此，OpenFlow 包分类相对五元组包分类更加复杂. 这种复杂的包分类方式可以根据流表规则中多样化的匹配字段组合对网络流量进行更加细粒度的分类，实现比传统分类技术更加灵活的转发策略. OpenFlow 在 1.2 版本后增加了使用 TLV(类型、长度、值)格式描述流表项的方式，提高了规则表达的灵活性. TLV 格式只是流表描述方式不同，包分类算法需要对 TLV 进行解析，这部分属于工程问题，本文不赘述.

表 2　OpenFlow 1.0 包含字段

包头字段	标记	位数	匹配方式	匹配算法
进入端口	Ingr	32	精确匹配	哈希
源 MAC 地址	SrcMAC	48	精确匹配	哈希
目的 MAC 地址	DstMAC	48	精确匹配	哈希
以太网类型	Eth_type	16	精确匹配	哈希
VLAN ID	VID	12	精确匹配	哈希
VLAN 优先级	Vprty	3	精确匹配	哈希
源 IP 地址	SA	32	前缀匹配	查找树
目的 IP 地址	DA	32	前缀匹配	查找树
IP 协议	Ptrl	8	精确匹配	哈希
IP TOS 位	ToS	6	精确匹配	哈希
源端口	SP	16	范围匹配	范围树
目的端口	DP	16	范围匹配	范围树

OpenFlow 流表的流表项包含对数据包进行匹配的匹配域(Header Fields)、统计匹配数据包个数的计数器(Counters)、用于对数据包如何处理的动作指令(Instructions)、决定流表项匹配次序的优先级(Priority)、表示流的最长有效时间的超时定时器(Timeouts)和保存控制器选择的不透明数据值(Cookie)，本文只讨论与网络包分类相关的部分，即表 3 中所包含的部分.

如表 3 所示，在 OpenFlow 流表中，每个字段都可以为通配字段，因此网络管理者可以灵活地使用这些字段，从而决定使用和管理何种粒度的网络流. 例如，管理者可以在流表规则中设置只有 IP 目的地址字段的规则，进而仅根据 IP 地址进行路由，规则中其他字段均为通配，如表 3 中规则 3 所示. 对于根据五元组进行的防火墙过滤应用，相关流表规则只有五元组或其部分是有效的，其余字段均为通配，如表 3 中规则 2 所示. 事实上，网络管理者一般只对网络包的少部分字段感兴趣. Gupta 在其论文[20]中对从 101 个不同的 ISP 和商业网络中收集到的 793 个分类器和 41 505 条规则进行了统计，结果显示，在这些分类器的规则中 17%的分类规则仅指定 1 个字段，23%的分类规则指定了 3 个字段，60%的分类规则指定了 4 个字段.

表 3　OpenFlow 流表规则举例

编号	Ingr	SrcMAC	DstMAC	Eth_type	VID	Vprty	SA	DA	Ptrl	ToS	SP	DP	优先级	决策
1	3	50:20:AA:5C:2D:60	31:32:45:9A:91:A1	0x0800	*	5	175.77.88.172/32	113.64.60.0/24	TCP	0	0～1024	750～750	0	转发端口1
2	*	*	*	*	*	*	175.77.88.0/24	113.64.60.32/32	TCP	*	0～1024	760～760	1	转发端口2
3	*	*	*	*	*	*	95.105.142.0/23	*	UDP	*	*	0～50	2	拒绝
4	*	*	*	*	*	*	*	204.14.27.39/32	*	*	*	*	3	转发端口3
5	*	*	44:33:02:DA:A7:0C	*	*	*	*	*	*	*	*	*	4	转发端口4
6	*	*	*	*	*	*	*	*	*	*	50～100	0～1024	5	丢弃

注：'*'表示通配字段.

综上所述，OpenFlow 包分类有 3 个特点：

(1) OpenFlow 网络同样存在流的局部特性. 在网络中，局部性特点是网络协议和传输数据的应用程序带来的自然结果.

(2) OpenFlow 包分类的匹配字段有数量多、宽度不等、匹配方式多样等特点. 这是由 OpenFlow 网络特性所造成的，OpenFlow 网络要加强对流量的管理与控制，需要进行细粒度的流分类，因此需要匹配较多的网络协议字段信息.

(3) OpenFlow 包分类的每条规则中仅含有较少的字段. OpenFlow 流表支持 12 种及更多字段不代表这些字段在每条规则中都要使用，实际规则所包含字段由现实网络需求所决定.

4　流量自适应的多维包分类算法

4.1　算法设计思想

OpenFlow 包分类在分类一个网络流时很有可能只匹配少部分字段的流表规则，这类规则大部分字段为通配字段，如果不加区分地对所有字段进行匹配，会造成计算资源的浪费，进而影响包分类性能. 由前文所述流量的局部特性可知，一段时间内网络包分类有较大可能匹配相同字段. 因此，可以利

① 本文此处提到并在实验中使用的 OpenFlow 协议版本是 v1.0，它是目前使用和支持最广泛的 OpenFlow 版本，本文所提出的方法具有更广泛通用性，不限于这个版本.

用时间局部性,后续匹配中优先匹配前一段成功匹配网络包所使用字段.具体来说,可以记录成功匹配网络包所用字段的频率,并利用命中频率对字段进行排序,使随后到达的网络包优先匹配那些最近最常使用的字段,即通过少部分字段就可以成功匹配分类规则,剩余字段就不再需要匹配,进而减少匹配时间,提高包分类效率.

根据流量的时间局部特性,最近一段时间经常使用的字段很有可能会被再次匹配,因此关键步骤是确定最近一段时间内最常使用的字段.由于网络流量的局部性由网络包构成,流量匹配以网络包为基础,因此,本方法以最近一段时间内字段使用频率为依据,通过对字段使用进行排序来确定字段使用的局部性.由于网络数据不停变化,每隔一段时间需要对字段重新进行排序.

4.2 方法详细描述

4.2.1 算法流程

本方法利用 BV 算法思想,为每个字段建立一个数据结构,数据结构可以根据字段的特点和匹配方式(精确匹配、前缀匹配、范围匹配)灵活选用,具体在 4.2.2 节讨论.在字段之间,则根据流的局部特性对字段进行排序,按照顺序进行先后匹配.

图 2 是 OpenFlow 网络包分类算法的框架,与大部分网络包分类算法相同,该分类算法的输入是网络包的包头信息,不同的只是本算法依据的是 OpenFlow 中 12 个包头字段而不是五元组,输出是对网络包的处理决策,即决定网络包接下来的处理方式,如包被转发到某端口或是被丢弃等.关键部分是通过规则库分类网络包的分类算法.算法用到了一些辅助数据结构,表 4 给出了对这些算法辅助数据结构的说明.整体流程如下:

(1)预处理.根据每个字段特点选取一个匹配算法,并建立一个相应的数据结构来存储规则在字段上的匹配内容;同时,在每个字段上建立一个 n 位的匹配向量,代表规则库中的 n 条规则,按优先级从高到低排列,标记当前输入字段的匹配结果.

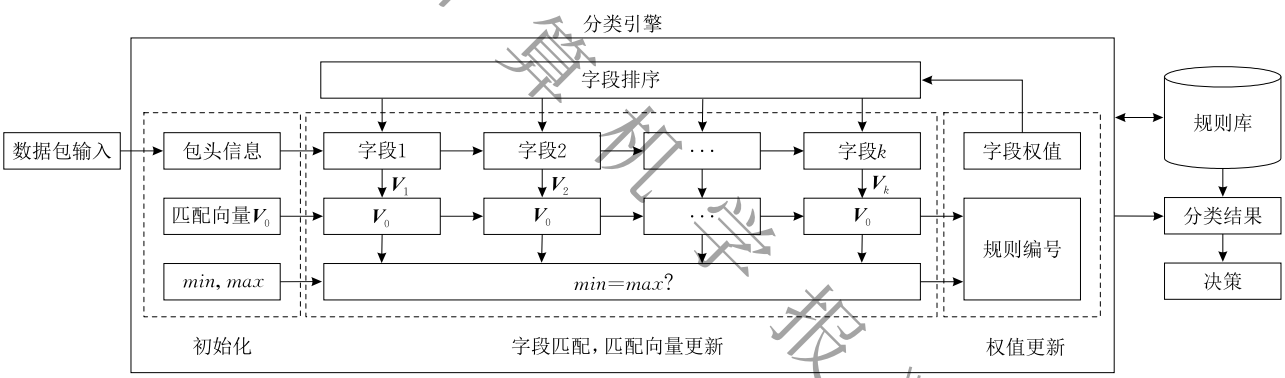


图 2 数据包分类系统框架

表 4 算法辅助数据结构说明

数据结构名称	说明
匹配向量 $\boldsymbol{V}_0$	位数与流表规则数相同,每一位代表一条规则,某一位为 1 代表匹配相应的规则.
辅助变量 min	匹配向量中第一条有效匹配规则的编号.
辅助变量 max	匹配向量中最后一条有效匹配规则的编号.
字段权值 W	字段权值越高代表该字段匹配的优先级越高.

(2)初始化.当网络包需要匹配时,首先为其建立一个 n 位的初始匹配向量 $\boldsymbol{V}_0$,每一位代表一条流表规则,构造顺序与各字段匹配向量相同.此向量的所有位都初始化为"1",表明可以匹配所有规则;然后为其设置两个辅助变量 min 和 max[26],代表匹配向量中第一条和最后一条有效匹配结果的编号,初始时 $min=1$,$max=n$.这里 min 和 max 有两个作用:第一,可以减少向量间"与运算"的位数,即"与运算"从第 min 位开始到第 max 位结束,不需要对全部向量进行运算;第二,当 min 和 max 相等时即可确定向量中只剩一条规则相匹配.

(3)字段排序.为每个字段 R_i 设置一个权值 W_i,W_i 越大的字段匹配优先级越高.选定时间周期 T,在一个时间周期开始时,对字段的权值进行清零,在包匹配的同时对字段权值进行更新(例如计数操作),在这个时间周期结束时,按照权值的大小对字段进行重新排序,可以根据这个顺序确定下一个时间周期内字段的匹配顺序.

(4)字段匹配.按照字段排序的顺序对各个字段逐个进行匹配,可以根据各个字段的特点采用不同的匹配算法,如表 2 所示.每个字段匹配完成后获得数据包在该字段上的匹配结果向量 $\boldsymbol{V}_i$.

(5)向量更新.当一个字段匹配完成后,将网络包的匹配向量 $\boldsymbol{V}_0$ 和该字段匹配结果向量 $\boldsymbol{V}_i$ 进行与运算,对 $\boldsymbol{V}_0$ 进行更新,同时更新辅助变量 min 和 max.此时,若 min 和 max 不存在,则说明没有相匹

《计算机学报》

配的规则，匹配结果为空. 若该字段是最后一个字段，编号为 min 的规则即为最终匹配结果. 若 min 和 max 相等，那么说明只剩一条规则相匹配，则只需比较数据包和该条规则的剩余字段.

(6) 规则比较. 当只剩一条规则和数据包相匹配时，不需要在剩余字段继续进行匹配，可以直接将网络包与该条规则的剩余字段进行比较. 若所有字段均匹配，那么该条规则即为匹配结果；若存在字段不匹配，那么最终匹配结果为空.

(7) 权值更新. 每当一个网络包匹配一条规则后，对字段权值进行更新，更新方法为将该规则中的所有非通配字段的权值加 1.

流量自适应的 OpenFlow 包分类算法如算法 1 所示，下面举例对整个算法流程进行说明. 所用规则库以表 3 为例，在预处理阶段，各字段根据选用的匹配算法初始化数据结构，在源 IP 地址和目的 IP 地址字段建立查找树，在源端口号和目的端口号字段建立范围树，在其他字段建立哈希表.

算法 1.　流量自适应的 OpenFlow 包分类算法.

输入：网络包头 H，规则库 $C(R[1...n])$
输出：与数据包匹配的规则编号 num

```
1.  initialize V0, min, max, W[1...k]
2.  IF T≤(current_time−last_time) THEN
3.     sort F[1...k] according to W[1...k]
4.     last_time←current_time
5.     W[1...k]=0
6.  END IF
7.  FOR i=1 TO k DO
8.     Vi←match H in field i
9.     V0←V0 & Vi
10.    update min, max
11.    IF i=k THEN
12.          num←min
13.          update W[1...k] according to R[i]
14.    ELSE
15.          IF min=max THEN
16.             IF R[min] and H is matched THEN
17.                num←min
18.                update W[1...k] according to R[i]
19.             ELSE
20.                num←NULL
21.             END IF
22.          END IF
23.    END IF
24. END FOR
```

例如，经过解析获取一个网络包各个字段的内容，得到：进入端口：19，源 MAC 地址：50:20:AA:5C:2D:60，目的 MAC 地址：31:32:45:2C:19:8D，以太网类型：0x0800，源 IP 地址 175.77.88.172，目的 IP 地址：113.64.60.32，IP 协议：TCP，IP TOS 位：0，源端口号：120，目的端口号：760（由于以太网类型是 0x0800，所以不含有 VLAN ID 和 VLAN 优先级字段）. 首先为网络包建立初始匹配向量 $\boldsymbol{V}_0=(111111)$，辅助变量 $min=1$，$max=6$. 假设当前时间周期内字段的匹配顺序是目的 IP 地址、目的端口号、源端口号、目的 MAC 地址、源 IP 地址、IP 协议、IP TOS 位、进入端口、源 MAC 地址、以太网类型、VLAN ID、VLAN 优先级，这个顺序是根据上个时间周期内各字段权值由高到低排序产生.

按照当前周期字段顺序对网络包的各字段进行匹配：首先是目的 IP 字段，目的 IP 字段采用是查找树匹配算法，通过查找树可以获取匹配结果 $\boldsymbol{V}_1=(111011)$，$\boldsymbol{V}_0$ 和 $\boldsymbol{V}_1$ 进行“与运算”，结果保存在 $\boldsymbol{V}_0$ 中，此时 $\boldsymbol{V}_0=(111011)$，同时更新 min 和 max，二值未发生变化；接下来匹配目的端口号字段，目的端口号字段采用的是范围树匹配算法，通过匹配获取向量 $\boldsymbol{V}_2=(010111)$，对 $\boldsymbol{V}_0$ 更新后 $\boldsymbol{V}_0=(010011)$，其中，$min=2$，$max=6$；然后是源端口号字段，同样采用范围树查找算法，查找结果为 $\boldsymbol{V}_3=(111110)$，更新后 $\boldsymbol{V}_0=(010010)$，$min=2$，$max=5$；其次是目的 MAC 地址字段，采用的是哈希算法，在哈希表中进行匹配，查找结果为 $\boldsymbol{V}_4=(111101)$，更新后 $\boldsymbol{V}_0=(010000)$，$min=2$，$max=2$. 此时发现 min 和 max 的值相等，都为 2，不再继续进行其他字段匹配，而是直接比较网络包头和规则 2 的剩余 8 个字段，比较发现二者的剩余字段均匹配，则说明数据包最后匹配的结果是规则 2，要执行的决策是转发至端口 2.

最后，对当前周期内的权值进行更新，以便用于下一个周期的字段排序，将规则 2 中不为通配字段的权值加 1，即源 IP 地址、目的 IP 地址、IP 协议、源端口、目的端口 5 个字段的权值均加 1.

4.2.2　字段内部匹配算法

本文所提方法可以根据每个字段匹配方式不同使用不同的字段匹配算法，如表 2 中所示. 对于精确匹配，采用基于哈希(Hash)的算法；对于前缀匹配，采用基于特里树(Trie)[27-29]的算法；对于范围匹配，采用基于范围树(Range Tree)[30]的算法. 这些算法在各匹配问题中效果较好，比较成熟，这里仅做介绍，对某一字段的匹配方法可以灵活配置.

(1) 哈希算法. OpenFlow 包分类中大部分字段被定义为精确匹配，这些字段采用哈希算法. 哈希表是用来实现哈希算法的数据结构，它可以实现 $O(1)$ 级别的快速插入操作和查找操作，非常适合精确信息查找. 建立哈希表时，把相应字段的值作为关键字，通过哈希函数计算出存储地址，再把关键字中信息存储在其中. 查找时用同样的关键字和哈希函

《计算机学报》

数即可找到关键字的存储地址.

哈希算法的优点是实现简单、分类速度快,缺点是当发生哈希冲突(Collision)时效率较低,所谓哈希冲突就是指不同的关键字可能得到同一哈希地址.因此哈希算法的一个关键点是冲突处理,一方面,要求采用的哈希函数尽可能减小冲突概率,使各关键字尽可能均匀的分布在哈希表中,另一方面,在产生哈希冲突的时候要采用正确的冲突处理方法.本算法采用的哈希函数是信息摘要算法第 5 版(Message Digest Algorithm 5,MD5),MD5 算法在计算机领域和信息安全领域被广泛使用,它具有如下特点:易于计算,任意长度的原始数据都可以很容易地计算出固定长度的 MD5 值;不易冲突,对原始数据的微小改动都会产生完全不同的 MD5 值.

哈希算法的冲突处理方法包括开放定址法:当产生哈希冲突时以当前的哈希地址为基础产生一个新的哈希地址直到不再冲突;公共溢出区法:专门建立一个公共溢出表用来存放产生冲突的关键字;再哈希法:同时构造多个哈希函数,每当产生冲突就使用一个新的哈希函数,直到找到一个不产生冲突的哈希地址;链地址法:将产生冲突的关键字保存在同一个哈希地址所指向的链表中.本文采用链地址法,该方法将彼此产生冲突的关键字保存在同一个单链表中,而在哈希表的地址单元中,则保存着相应的单链表表头指针,如果地址中没有关键字,那么表头指针为空指针.链地址法相对其他冲突处理方法例如开放定址法需要更多的存储空间来存储链表指针,但它在查找、插入和删除时只需比较同一个链表中的节点,查找速度更快,更加适合软件算法.图 3 展示了使用链地址法生成哈希表的简单示例.

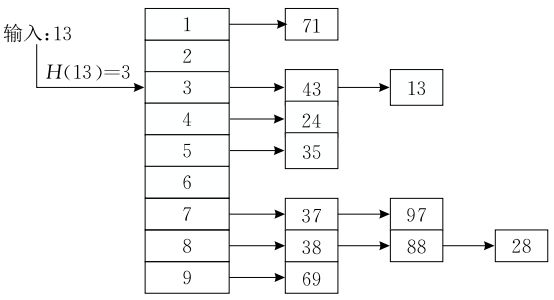


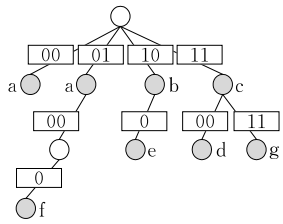
图 3 使用链地址法的哈希函数示例

(2)特里树算法. 特里树(也称查找树)是一种简单有效进行前缀匹配的算法,经常被用于文本词频统计和 IP 地址查找.它的核心思想是使用空间换时间,利用存储的数据公共前缀来减少查询时间,减少不必要的比较,提高查找效率.该算法基于 IP 地址前缀构建一棵特里树,这棵树的特点是在根节点中不包含数据,除根节点外的其他节点都包含一部分数据,每个节点的子节点包含的数据都不相同,从根节点到某一个节点所有的数据按顺序连接在一起就是这个子节点对应的前缀数据.查找时,从根节点开始对查找树进行遍历,每当搜索到一个前缀都把它当作最长前缀并根据当前节点子节点中的数据选择相应的子树继续向下查找,直到无法找到更长的前缀为止,这时所记录的前缀即为 IP 地址的最长匹配前缀.

特里树算法经过了多年发展已经非常完善,有较多的优化方法来提高查找速度.特里树一次查找的位数称为步长(Strides),标准特里树是一棵二叉树,它的步长为 1,即一次查找一位,可以通过增加步长使特里树变为多叉树来增加一次查找的位数,从而减少总的查找次数.多叉特里树可以为定步长,也可以为变步长,变步长相比定步长更加地灵活,但实现也更加地复杂.图 4(b)即是根据图 4(a)中地址前缀生成的最大步长为 2 的可变步长四叉特里树.本文采用图 4(b)所述的可变步长四叉特里树处理前缀匹配.

编号	地址前缀
a	0*
b	10*
c	11*
d	1100*
e	100*
f	010001
g	1111*

(a) 地址前缀列表



(b) 可变步长四叉特里树

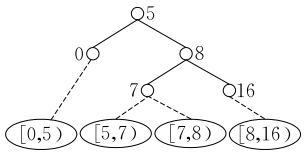
图 4 优化的特里树算法示例

(3)范围树算法. 范围树是一种用来进行范围查找的树结构,它将一个大范围区间划分成一个个小的范围,每个单元范围对应一个叶子节点.本文用它来对源端口号和目的端口号进行范围匹配.建立范围树的方法是:首先,取所有区间的上界和下界进行排序,建立一个有序数列,假设数列中有 m 个互不相同的值,那它们就可以构成 $m-1$ 个互不重叠的区间;然后,以这 m 个互不相同的值为节点建立一棵平衡查找树,并将这 m 个不同的值存储在叶子节点左右子节点中,每两个子节点之间就是一个区间,各区间互不重叠;最后,将规则库中的规则映射到区间上.在范围树中,每个叶子节点上存储着一个范围区间,每个非叶子结点上储存着一个精确的数值.

当进行查找操作时，从根节点开始进行向下搜索，如果要查找的数值（即源端口号和目的端口号）小于节点中数值，就在该节点的左子树向下查找；如果要查找的数值大于等于节点中数值，则从该节点的右子树向下查找. 直到到达一个叶子节点，这个数值就被包含在叶子节点所代表的区间中，该叶子节点中的规则就是这个数值要匹配的规则. 例如，宽度为 4 位数值生成的区间范围[0,16)，其中有如图 5(a)所示的 3 个区间，则它生成的范围树由图 5(b)所示.

编号	区间
a	[0,7)
b	[5,16)
c	[8,16)

(a) 区间范围列表



(b) 范围树

图 5　优化的范围树算法示例

5　实验分析

5.1　实验环境

为了验证本文方法效果，我们在 Open vSwitch 2.3.1[①] 软件上实现了文中方法. Open vSwitch(OVS)是一款由 C 语言编写的虚拟交换机软件，它是 OpenFlow 1.0 协议的重要开源实现.

我们使用校园网出口真实采集的网络流量，该出口带宽为 1 Gbps，流量占用带宽超过 80%，流量长度为 1 小时. OVS 流表规则集由 ClassBench[②][31] 和 FRuG[③][32] 生成. ClassBench 是包分类领域的规则生成器，其产生的规则集被广泛应用在包分类研究领域，它可以生成 ACL(Access Control List)、FW(Fire Wall)、IPC(Linux IP Chains) 3 种类型的规则，但是它生成的规则只包含五元组，不能够满足验证本文方法的需求，因此本文还利用 FRuG 生成规则的其他部分字段.

本文生成的实验规则主要包括以下几部分：

(1) 1 个字段：目的 IP 地址，用于 IP 地址查找；目的 MAC 地址，用于 MAC 地址查找.

(2) 2 个字段：源端口号、目的端口号，一般用于应用防火墙.

(3) 5 至 7 个字段：源 IP 地址、目的 IP 地址、源端口号、目的端口号、协议类型、TOS 位，其中大部分为五元组，少部分不包含协议类型字段，部分增加了 TOS 位字段，主要用于防火墙和 ACL.

(4) 其他：各个字段均有可能包含，用于 OpenFlow 网络中的其他应用.

根据规则集大小和测试所需规则集所述情况(上述(1)～(4)情况)，生成规则数量在每字段上概率相等，即对于 1 字段规则，产生相同的含有 IP 或 MAC 的规则. 规则之间没有包含关系.

包分类算法最重要的性能指标是包分类速度，它是评价包分类算法的最重要参数；其次是内存占用量，它是评测包分类算法的重要指标. 随着内容容量、速度和缓存的发展，算法可支配的内存有了较大提高，除处理大规模规则集时内存消耗至关重要外，一般算法对内存要求并不苛刻.

5.2　包分类方法的速度

5.2.1　同 OVS 原有算法的速度比较

Open vSwitch(OVS)原有的包分类算法分为两部分：用户态和内核态. 用户态采用改进的 TSS 算法匹配网络包，内核态则缓存部分近期使用的 TSS 算法所使用的哈希表，便于快速查找. 严格说，OVS 已经考虑了利用网络流量局部性对查找速度进行优化. 当进行包分类时，先在内核态进行快速匹配，如果匹配失败再到用户态进行匹配. 实验通过在 OVS 中运行本文算法来实现 OpenFlow 包分类，具体来说，实验在保留 OVS 原有包分类算法的基础上，在用户态增加了本文算法，对比两个算法在用户态的匹配性能. 由于内核态是用户态部分数据结构的缓存，缓存对性能的提升不在本文研究范围内，未考虑缓存效果对本文算法的进一步提升.

图 6 在不同的规则集规模下分别对算法性能进行了测试和对比，其中，本文算法采用了 1 s 为周期对匹配字段进行排序. 由图 6 可以看出，本文提出算

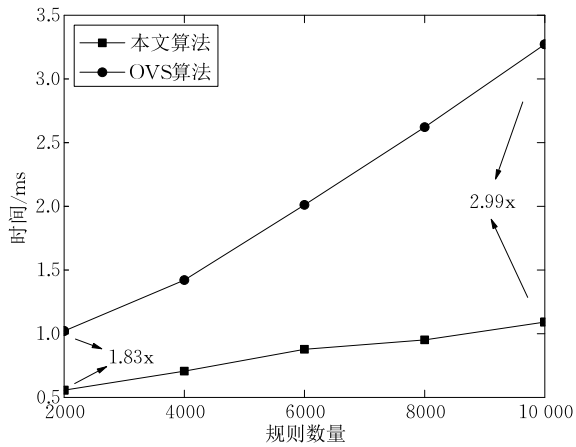


图 6　本文算法与 OVS 原算法时间对比图

① http://openvswitch.org/
② http://www.arl.wustl.edu/~det3/ClassBench
③ http://sites.google.com/site/thilangane/research

《计算机学报》

法在用户态实现的情况下性能要好于 OVS 原有用户态算法.随着规则库规模的增大加速比有增大趋势,本文算法更为优越.在 2 千规模的规则集中,本文算法相比 OVS 的加速比为 1.88,在 1 万规模的规则集中加速比为 2.99.

5.2.2 同五元组扩展算法的速度比较

本文算法还分别在五元组和 OpenFlow 十二元组中同几种常用的包分类算法进行了比较.其中 RFC 算法由于本身的局限性,在字段较多情况下会产生内存爆炸,无法在 OpenFlow 环境下使用,故本文仅给出了其在五元组包分类中的性能.图 7 列出了在 10K 规模规则集下本文算法和其他 4 种常见包分类算法的时间性能比较.

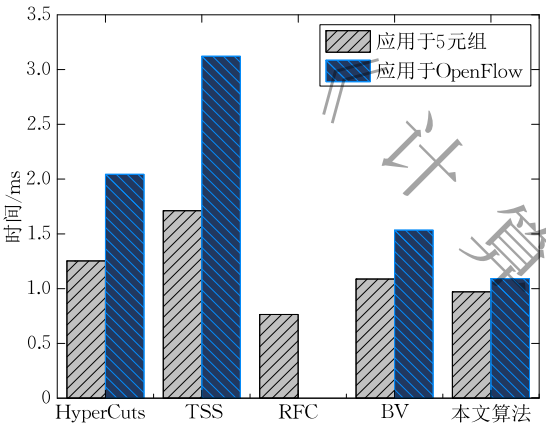


图 7 各算法时间性能对比图

由图 7 可以看出,针对五元组包分类,本文算法查找速度仅逊于 RFC 算法,比基于空间划分并建立决策树的 HyperCuts 算法和基于哈希的 TSS 算法的性能要好,和 BV 算法相差不多;在十二元组包分类中,本文算法时间性能最优,其中 HyperCuts 由于字段增多导致建立决策树深度增加,访存次数增多,而 TSS 算法在字段较多的情况下元组划分也急剧增多,导致在 OpenFlow 中性能出现下降.图 7 中还可以看到,本文算法从维度为 5 到 12 的变化过程中,实际匹配性能降低不到 10%,在维度扩展上具有较好效果.

5.3 包分类方法的存储占用

5.3.1 同 OVS 原有算法的存储比较

图 8 是本文算法和 OVS 原算法存储空间的对比图,由图 8 可以看出,本文算法的存储空间要优于 OVS 原有算法,这里仅比较了用户态存储空间.在实际 OVS 系统中,规则集在用户态和内核态均有存储,内核态存储的内容是用户态数据结构的缓存,因此,比较用户态存储空间可以代表算法效果.

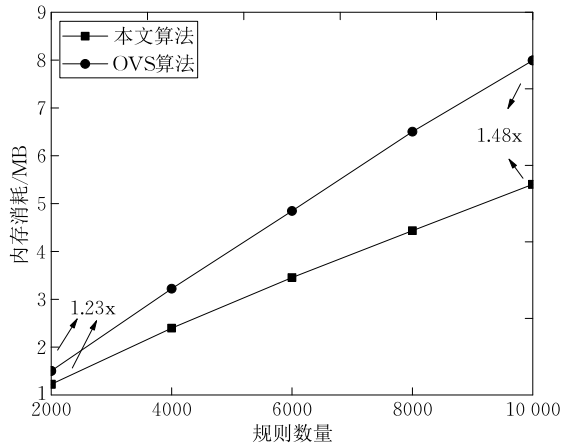


图 8 本文算法与原算法存储空间对比图

5.3.2 同五元组扩展算法的存储比较

图 9 给出了在 2k 规模规则集下五元组和十二元组中本文算法的存储占用和其他算法的对比.可以看到,针对五元组包分类,本文算法和 BV 算法存储占用类似,要多于 HyperCuts 算法,这是因为尽管这三个方法在五元组字段上都采用树来进行存储,但本文算法和 BV 算法需要更多的存储空间来保存位向量;在十二元组包分类中,由于字段增多导致 HyperCuts 算法建立的决策树深度增加,存储占用大大增多,而本文算法和 BV 算法增加的字段均采用哈希结构,存储占用增加可控.完全采用哈希进行存储的 TSS 算法无论在五元组中还是在十二元组中储存占用都是最少,而需要建立多个预处理表和交叉乘积表的 RFC 算法占用的存储空间要远远大于其他算法,在十二元组中已经无法使用.

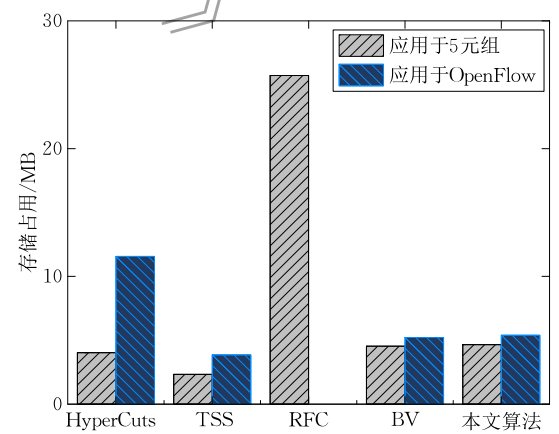


图 9 各算法存储占用对比图

5.3.3 OVS 实现讨论和分析

本文包分类算法主要实现在 OVS 的用户态,其内核态主要是对用户态算法数据结构的缓存,从而通过缓存提高分类速度.即使原有 OVS 算法也只是在用户态实现.本文沿用 OVS 这种设计理念,

在用户态实现算法.如果希望在内核态引入本文算法,需要设计本文算法从用户态到内核态缓存方式.在具体实现中,可以将字段进行组合,从而减少字段个数,从而进一步加快包分类速度.

从字段角度来看,如果字段数为 k,本文算法存在 $O(k)$的时间和空间复杂度,具体到每个字段,其匹配的时间和空间复杂度取决于所使用算法本身.当更新流表项时,本文算法需要对于更新流表项所覆盖字段进行匹配数据结构更新,计算复杂性取决于各字段使用的匹配算法.本文算法对各字段匹配结果联合时不额外增加计算复杂性.

6　结论和展望

针对 OpenFlow 网络中数据包转发性能难以满足当前网络需求的问题,本文介绍了一种应用于 OpenFlow 网络环境的流量自适应包分类算法,该算法充分考虑了 OpenFlow 包分类字段数量多、字段宽度不同和匹配方式多样的特点,利用网络流的局部特性,通过在不同字段上使用不同算法并且优先匹配最近经常使用的字段而忽略通配字段来达到提高查找速度的目的.本文方法在 Open vSwitch 中实现,相比原方法在用户态模式下约有 2 倍的性能提升,相比其他从五元组包分类扩展的算法有 40%以上的性能提升.

原有网络协议和框架在不断地更新,而新的网络框架在不停地涌现,网络包分类技术的研究与发展还有很多挑战.首先,网络需求的增加和网络技术的发展使得网络包分类技术在转发性能上需要不断地改进以满足更大的网络带宽.其次,新兴网络应用结构的出现对网络包分类技术有了更高的要求.例如,软件定义网络要求网络能够进行虚拟化和可编程化,业务感知网络[33]要求对网络数据流中的业务进行感知、分析、决策和控制;数据中心网络(Data Center Network)[34]要求有灵活的链路容量控制,不同服务间流量可以进行隔离.这些都需要网络包分类技术提供更加灵活和多样的支持.最后,网络包分类技术能够应用在具备网络包分类功能的设备之中,性能和功能的提高可以辅助网络设备改进性能和功能,推动硬件设备的更新与发展.

参　考　文　献

[1] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

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

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

[4] Guerin R A, Orda A, Williams D. QoS routing mechanisms and OSPF extensions//Proceedings of the Global Telecommunications Conference. Phoenix, USA, 1997: 1903-1908

[5] Yeh J, Chow R, Newman-Wolfe R. Interdomain access control with policy routing//Proceedings of the IEEE Computer Society Workshop on Future Trends of Distributed Computing System. Tunis, Tunisia, 1997: 46-52

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

[7] Xu Ke, Wu Jian-Ping, Xu Ming-Wei. Advanced Computer Networks: Architecture, Protocol Mechanism, Algorithm and Routing Technology. 2nd Edition. Beiing: China Machine Press, 2009(in Chinese)
(徐恪, 吴建平, 徐明伟. 高等计算机网络: 体系结构、协议机制、算法设计与路由器技术. 第 2 版. 北京: 机械工业出版社, 2009)

[8] Ruiz-Sanchez M A, Biersack E W, Dabbous W. Survey and taxonomy of IP address lookup algorithms. IEEE Network, 2001, 15(2): 8-23

[9] Taylor D E, Turner J S, Lockwood J W, et al. Scalable IP lookup for Internet routers. IEEE Journal on Selected Areas in Communications, 2003, 21(4): 522-534

[10] Lakshminarayanan K, Rangarajan A, Venkatachary S. Algorithms for advanced packet classification with ternary CAMs. ACM SIGCOMM Computer Communication Review, 2005, 35(4): 193-204

[11] Pérez K G, Yang X, Scott-Hayward S, et al. Optimized packet classification for software-defined networking//Proceedings of the IEEE International Conference on Communications (ICC). Sydney, Australia, 2014: 859-864

[12] Qi Ya-Xuan, Li Jun. Theoretical analysis and algorithm design of high-performance packet classification algorithms. Chinese Journal of Computers, 2013, 36(2): 408-421(in Chinese)
(亓亚烜, 李军. 高性能网包分类理论与算法综述. 计算机学报, 2013, 36(2): 408-421)

[13] Gupta P, Mckeown N. Classifying packets with hierarchical intelligent cuttings. IEEE Micro, 2000, 20(1): 34-41

[14] Singh S, Baboescu F, Varghese G, et al. Packet classification using multidimensional cutting//Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. Karlsruhe, Germany, 2003: 213-224

[15] Han Wei-Tao, Yi Peng, Zhang Xia. Hybrid cutting algorithm for packet classification. Journal of Software, 2014, 25(11): 2616-2626(in Chinese)

(韩伟涛，伊鹏，张霞. 一种采用混合切分法的报文分类算法. 软件学报，2014，25(11)：2616-2626)

[16] Kogan K，Nikolenko S，Rottenstreich O，et al. SAX-PAC (scalable and expressive packet classification). ACM SIGCOMM Computer Communication Review，2014，44(4)：15-26

[17] Srinivasan V，Suri S，Varghese G. Packet classification using tuple space search. ACM SIGCOMM Computer Communication Review，1999，29(4)：135-146

[18] Pfaff B，Pettit J，Koponen T，et al. The design and implementation of open vswitch//Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). Santa Clara，USA，2015：117-130

[19] Yang B，Fong J，Jiang W，et al. Practical multituple packet classification using dynamic discrete bit selection. IEEE Transactions on Computers，2014，63(2)：424-434

[20] Gupta P，Mckeown N. Packet classification on multiple fields. ACM SIGCOMM Computer Communication Review，1999，29(4)：147-160

[21] Baboescu F，Varghese G. Scalable packet classification. IEEE/ACM Transactions on Networking，2005，13(1)：2-14

[22] Lakshman T V，Stiliadis D. High-speed policy-based packet forward using efficient multi-dimensional range matching. ACM SIGCOMM Computer Communication Review，1998，28(4)：203-214

[23] Qu Y，Zhou S，Prasanna V K. Scalable many-field packet classification on multi-core processors//Proceedings of the 25th International Symposium on Computer Architecture and High Performance Computing. Porto de Galinhas，Brazil，2013：33-40

[24] Zhou S，Qu Y R，Prasanna V K. Multi-core implementation of decomposition-based packet classification algorithms//Proceedings of the International Conference on Parallel Computing Technologies. Petersburg，Russia，2013：105-119

[25] Fang W，Peterson L. Inter-AS traffic patterns and their implication//Proceedings of the Global Telecommunications Conference. Rio de Janerio，Brazil，1999：1859-1868

[26] Trivedi U，Jangir M L. EQC16：An optimized packet classification algorithm for large rule-sets//Proceedings of the International Conference on Advanced in Computing，Communications and Informatics. New Delhi，India，2014：112-119

[27] Srinivasan V，Varghese G. Fast IP lookups using controlled prefix expansion. ACM SIGMETRICS Performance Evaluation Review，1998，26(1)：1-10

[28] Nilsson S，Karlsson G. IP address lookup using LC-tries. IEEE Journal on Selected Areas in Communications，1999，17(6)：1083-1092

[29] Morrison D R. PATRICIA-practical algorithm to retrieve information coded in alphanumeric. Journal of the ACM，1968，15(4)：514-534

[30] Warkhede P，Suri S，Varghese G. Multiway range trees：Scalable IP lookup with fast updates. Computer Networks，2004，44(3)：289-303

[31] Taylor D E，Turner J S. ClassBench：A packet classification benchmark//Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Florida，USA，2005，3：2068-2079

[32] Ganegedara T，Jiang W，Prasanna V. Frug：A benchmark for packet forwarding in future networks//Proceedings of the International Performance Computing and Communication Conference. Albuquerque，USA，2010：231-238

[33] Casado M，Freedman M J，Pettit J，et al. Ethane：Taking control of the enterprise. ACM SIGCOMM Computer Communication Review，2007，37(4)：1-12

[34] Joseph D A，Tavakoli A，Stoica I. A policy-aware switching layer for data centers. ACM SIGCOMM Computer Communication Review，2008，38(4)：51-62

WAN Yun-Kai，born in 1990，M. S. His research interests include Internet architecture and software-defined networks.

LIU Miao-Miao，born in 1992，M. S. candidate. Her research interests include Internet architecture and next-generation Internet.

LIU Yi，born in 1982，Ph. D. candidate，engineer. His research interests include Internet traffic analysis and modelling，deep packet inspection，network traffic management，and network security.

LI Dan，born in 1981，Ph. D.，associate professor. His research interests include Internet architecture，data center networks，software-defined network and cloud computing.

SONG Tian，born in 1980，Ph. D.，associate professor. His research interests include network and information security，Internet architecture and microprocessor design.

Background

In the road towards efficiently administering and securing network traffics，software-defined network (SDN) successfully inspired novel applications of network management and security by decoupling the control plane from the data plane

in the network middleboxes. Besides flexibility to configure and deploy, SDN requires a high-speed data plane to catch up the network traffic rate. In some scenarios, the performance even dominates the feasibility of deployment with SDN. The data plane of SDN has its computing-intensive function to classify packets against tens of fields of packet headers, namely SDN packet classification. Therefore, SDN requires a high speed algorithm for multi- dimensional (tens of) packet classification.

For conventional networks, many excellent algorithms to classify packets have been proposed, and it is regarded as a solved problem for some time. However, those algorithms are all designed to handle packets against classification rules with five or less tuples (fields) of packet header, while SDN requires a classification against twelve or even more fields (tuples). Due to the difference of the number of fields, most of the existing algorithms are no longer suitable for SDN. There are broadly two directions, extending five-tuple algorithms to more tuples and designing novel algorithms directly to tens of fields. The most recent work shows that algorithms based on bit vector is a promising direction to solve this issue.

Our work puts forward to advance this key computing bottleneck by proposing an extension to bit vector based algorithm for SDN packet classification based on bit vector which can adapt its classification process to the locality of network flow. The key point of our algorithm is to dynamically reorder the matching priority of different fields and accelerate the classification by intentionally skipping some wildcard matching fields. Our algorithm shows about two times speedup to the original algorithm in Open vSwitch which is a widely used open source prototype for SDN. Our work will reinforce the deployment of SDN in more scenarios.

Our group in Beijing Institute of Technology has been work on the data plane for about ten years. Our research interests include the architectures and algorithms of data planes in router, switch and security devices of all kinds of networks, such as SDN, NDN, ICN, IoT and green networks.

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61272510, 61432002, 61522205, and the Scientific Research Project of Shaanxi Education Department No. 14JK1825.

《计算机学报》