

软件定义网络中可扩展的流表项处理机制

朱向阳,陈 兵

(南京航空航天大学 计算机科学与技术学院,江苏 南京 211106)

摘要:数据中心内部的流量特性和 OpenFlow 交换机流表的超时机制及其容量的限制为软件定义网络(SDN)的应用带来了可扩展性难题,即由于老鼠流频繁调用控制器以及流表超时机制带来的重复路由请求,导致控制器平面和 OpenFlow 信道成为 SDN 架构的性能瓶颈而无法根据业务的需求提供有 QoS 保证的服务。针对此问题从两个角度提出了改进措施。为交换机配置以目的地址为导向的高优先级的通配符流表项,使得大部分老鼠流能够被转发平面直接处理而不需要询问控制器;在控制器和转发平面之间增加一个动态索引哈希缓存层,当流表项因超时或者流表空间耗尽被流表丢弃时将其缓存在缓存层中,此时收到重复的路由请求时能够避免重复计算。实验结果表明,提出的通配符流表项能够有效降低网络的延迟和流表项装载数量,而基于动态索引哈希的缓存层有着较高的命中率和查找速度。

关键词:软件定义网络;老鼠流;流表超时;通配符流表项;动态索引哈希

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2016)12-0012-05

doi:10.3969/j.issn.1673-629X.2016.12.003

Scalable Flow Table Entries Processing Mechanism in Software-defined Networks

ZHU Xiang-yang, CHEN Bing

(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics,
Nanjing 211106, China)

Abstract: The timeout mechanism and capacity limitation of flow table in OpenFlow switches, together with the traffic characteristics in data centers, have brought scalability problems in the application of Software-Defined Network (SDN). Due to the frequent and repeatable routing request invocations by mice flows and flow table timeouts, the SDN control plane and OpenFlow secure channel will become the performance bottleneck of whole SDN architecture, and thus cannot provide QoS guaranteed services according to network resources and business requests. Two improvement measures are put forward from two aspects. The switches are configured with destination address oriented wildcard flow table entries for mice flows, making that most of mice flows are handled by forwarding plane directly without asking control plane. A dynamic-index hash cache layer is added between the control plane and the forwarding plane, and flow table entries will be put into the cache layer when they are evicted from flow table because of timeout or other reasons, which can avoid repeatable routing calculations. Experimental results show that wildcard flow table entries can effectively reduce the network delay and the number of flow table entries, and the cache strategy proposed has higher hit rate and looking up speed.

Key words: SDN; mice flow; flow table timeout; wildcard flow table entry; dynamic-index hash

1 概述

随着互联网规模和复杂性的日益增长以及应用范围的不断扩大,传统网络架构正在发生巨大的变化。数据中心网络承载着诸如 Web 服务、网络游戏、电子商务及社交网络等众多应用,这些应用在数据中心网络中产生了大量的流量。最新研究表明,在数据中心网络中,90%的业务流小于 100 Mb(称为老鼠流),而

其余 10%的业务流(称为大象流)传输了超过 99%的网络流量^[1]。为了在数据中心网络中使得控制更加灵活、智能,需要采用更加细粒度的解决方案。

在传统网络中,诸多复杂的网络功能分散于各个网络设备上,全局网络视图的缺乏使得难以提供基于网络资源和业务流特征的细粒度服务,而网络结构的僵化使得难以对传统网络架构进行改进以提供灵活可

收稿日期:2016-02-29

修回日期:2016-06-16

网络出版时间:2016-11-22

基金项目:江苏省科技项目(未来网络前瞻性研究项目)(BY2013095-2-10)

作者简介:朱向阳(1987-),男,硕士研究生,研究方向为无线网络、SDN 网络;陈 兵,教授,研究方向为计算机网络、无线通信、信息安全。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20161122.1228.042.html>

区分服务。基于上述问题,软件定义网络(SDN)日益兴起并迅速吸引了学术界和工业界的注意。如图 1 所示,SDN 将控制平面与转发平面分离,网络层和控制层之间通过 OpenFlow 协议^[2]提供的安全信道交换信息,控制层通过 REST API 提供的北向接口向应用层提供服务。SDN 控制器负责执行网络决策任务,如负载均衡、接入控制、流量工程和路由计算,网络设备在控制器的指导下进行数据转发。网络管理者能够通过编写应用程序来配置、管理和优化底层的网络资源,从而实现灵活可控的网络以及提供高质量、细粒度和可订制的服务。

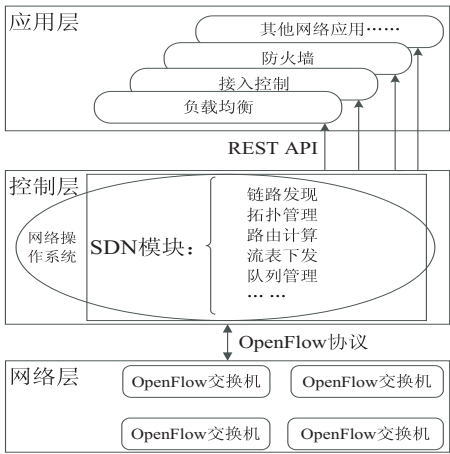


图 1 SDN 网络架构

OpenFlow 交换机中维护着一张流表,数据包通过匹配流表中的流表项执行相应操作,如广播、转发或丢弃。流表空间是有限的(大约 1 K 到 4 K),例如支持 OpenFlow 协议的 HP 5406zl 仅仅能支持 1 500 个流表项^[3],因为流表是由非常昂贵的支持并发查找和通配符的 TCAM 制作而成,其容量远远无法满足网络中业务量的需求。假设网络中有 N 台主机,那么所需要的总流表项数量大约为: $F * H * N(N - 1)/2$ 。其中, F 表示主机对之间平均的业务流数量; H 表示主机对间的平均跳数。显然总的流表项数目和 N^2 成正比,要远远大于流表空间的容量。为此,OpenFlow 协议中规定了两种流表项超时机制,空闲超时表示如果一个流表项在规定的时间内未获得匹配则被删除,硬超时意味着无论该流表项被匹配多少次,在一定的时间以后都会被丢弃。

数据中心业务流的特点和流表的特性会带来可扩展性问题:

(1) 数据中心中存在大量的老鼠流,这些老鼠流会频繁地调用控制器计算最优路径,从而导致 SDN 控制平面的性能下降以及延迟的增加,大量的路由计算请求还会造成 OpenFlow 安全信道的阻塞;

(2) 由于流表超时机制,已经被计算过的流表项

可能会被重复提交到控制平面,同时大象流在数据传输过程中可能会被老鼠流挤占流表项空间而导致传输中断。

由于在数据中心网络中存在大量的老鼠流,为每个老鼠流在 OpenFlow 交换机中分别制定一个确切的转发规则的代价过于昂贵,因此文中在识别出老鼠流的基础上,在交换机中为老鼠流建立一个以目的 IP 地址为导向的通配符流表项,在减少路由计算请求的同时降低流表空间的占用。此外如上文所述,流表空间的大小相对于网络中流的数量可以说微乎其微,因此流表的超时是无法避免的。在流表超时发生时,为了缓解流表项重复计算带来的开销,提出在控制器和交换机中间增加一个动态索引哈希流表项缓存器,当流表项因超时被移除时并不是简单地将其丢弃,而是先把它存入缓存器中。通过为老鼠流建立通配符流表项及使用缓存器,能够有效地解决上述可扩展性问题。

2 相关技术简介

SDN 是由美国斯坦福大学 Clean Slate 研究组提出的一种新型网络创新架构,其核心技术 OpenFlow 通过将网络设备控制面与数据面分离开来,从而实现了网络流量的灵活控制,为核心网络及应用的创新提供了良好的平台。OpenFlow 协议是 SDN 事实上的标准和实现方式,它允许研究人员使用软件搭建起媲美物理硬件的实验环境,控制平面和转发平面之间正是通过 OpenFlow 协议实现的安全信道来进行通信。每个 OpenFlow 交换机中都维护一张流表,流表中的每个表项代表一个转发规则,业务流通过和流表项的字段进行哈希查找和匹配以决定具体的转发规则,如果在流表中不存在相应的流表项,则该业务流将被交换机提交至控制器发起一次路由计算请求。

当前的多路径路由技术如 ECMP^[4] 随机地将流量分配到等价路径上,这种未考虑路径特征和流量特征的随机分配会导致某些链路的拥堵。对大象流和老鼠流分别进行处理,首先需要区分出老鼠流。研究表明,对老鼠流的检测可在边界交换机中(如 Hedera^[5])或终端设备中(如 Mahout^[6])进行。首先设定一个数据传输量的阈值,如果约定时间内传输的数据量少于该阈值则被标记为老鼠流。例如 Hedera 使用等价多路径传输(ECMP)来处理老鼠流,其规定当 OpenFlow 交换机处理数据量超过 100 MB 时则识别为大象流。

Martin 等^[7]使用遗传算法来归纳大象流的特征向量并使用这些向量获取连续的高命中流,这种方法宣称能够获得接近最佳的命中率,但是其缺点是训练过程需要大量的流特征数据,以获得最佳的性能。DIFANE^[8]通过使用一种称为“权威交换机”的专用交换

机来降低控制器平面的负载,这种控制器负责缓存部分流表并试图在转发平面处理所有的数据包,其缺点是控制器需要将部分控制功能下放到转发平面,这不仅增加了 OpenFlow 交换机的复杂性,也违背了 SDN 控制转发分离的思想。T. Pan 等^[9]提出了一种基于流长度和流缓存移除次数统计的流缓存替换规则来识别出大象流并提高缓存命中率,其采用自适应的最少命中数驱逐(ALFE)缓存替换策略为大象流赋予了更高的优先级。

针对 TCAM 造成的流表空间受限问题,学术界提出了各种改进方案。V. C. Ravikumar 等^[10]提出了一个多级管道流表架构,通过前缀压缩和分割来降低 TCAM 单元能量的消耗,这种方式无法应对网络中的突发流量。W. Wu 等^[11]提出了一种划分路由表的算法,并且把整个路由匹配过程分为两个阶段(索引 TCAM 和子 TCAM)以提高流表查找速度,降低能量消耗。F. Zane 等^[12]提出将 TCAM 分为基于位选择的两个层次,其中第一层是第二层分区的索引。缺点是使路由表更新变得复杂,无法应对超大规模的流量。

3 流表项处理机制

3.1 大象流检测

如上文所述,数据中心大量老鼠流会频繁调用控制器,从而导致控制器性能降低和网络时延的增加。于是提出为老鼠流制订专用的通配符流表项,使大多数老鼠流由转发平面直接处理,在提高控制器工作效率的同时,降低流表空间的占用和 OpenFlow 安全信道发生阻塞的风险。

大象流的检测有多种方法:

(1)计数检测在交换机的端口统计每个汇入流量的信息,然后将这些信息报告给控制器,如 Helios^[13]就使用这种机制。但是这种方法无法扩展到大规模网络中,因为其不仅消耗交换机控制器之间的带宽,而且会增加交换机和控制器的负担。

(2)标记检测是通过打标记的方法来识别大象流,其因为原理和实现简单而被广泛应用于网络研究中,但缺点是需要根据研究内容修改应用程序而无法在数据中心网络中部署。标记算法还可以根据 TCP 端口号等标记来对流量进行分类,这种方案较适合企业应用场景,但因为数据中心流量巨大且难以训练分类算法,仍然无法应用于数据中心网络。

(3)终端检测是通过在网络通信模块上新增一个垫片层,用于监控主机的 TCP 套接字缓冲区,当缓冲区的字节数在规定时间内超过了预定的阈值后,就会将对应的流标记为大象流。该方法有很多优点,终端可以根据用户的行为判断流量的类型,其检测开销很

小,适合于数据中心这样复杂的网络环境,因此文中采用终端检测方法检测大象流。

3.2 老鼠流聚合

由于数据中心存在大量的老鼠流,为每个老鼠流分别制定转发规则及建立流表项会带来极大的控制器计算资源和流表空间资源的开销。文中利用 OpenFlow 协议增加老鼠流聚合模块,为相同目的 IP 地址的老鼠流建立通配符流表项,OpenFlow 1.1 提出结合组表、流表及动作表能够支持多路径路由。多路径路由的实现是为流表项增加一个指针以表明其对应的组表项,并且为组表项增加一个字段指向其对应的动作表,动作表中每个表项都包含三个字段:路径、权重和动作。通配符流表项通过目的地址字段来匹配老鼠流,因此交换机流入的所有老鼠流都会匹配到某个流表项中,并执行对应组表项所指定的操作,如图 2 所示。



图 2 通配符流表项聚合

假设某交换机中有两个老鼠流表项,流表项 1 为: in_port = 1, actoins = output; 2, hard_timeout = 15, idle_timeout = 25, ip_src = 192. 168. 80. 1, ip_dst = 172. 16. 0. 1, vlan_id = 100, priority = 100, ip_tos_bits = 0。流表项 2 为: in_port = 3, actions = output; 4, hard_timeout = 15, idle_timeout = 25, ip_src = 192. 168. 81. 1, ip_dst = 172. 16. 0. 1, vlan_id = 101, priority = 200, ip_tos_bits = 0。此时可以为它们建立一个通配符流表项: inport = *, dst_ip = 172. 16. 0. 1, actions = output; 5, idle_timeout = 100, ip_src = *, ip_tos_bits = 1, priority = 10, vlan_id = *。可以看到通配符流表项拥有较高的优先级,且其超时时间更长。原本属于老鼠流 1 和老鼠流 2 的流表项在硬超时时间内未获得匹配后,将被流量统计管理器从流表中删除,从而降低流表空间的占用。

3.3 流表项缓存机制

针对流表空间有限带来的可扩展性问题,提出在转发平面和控制器之间增加一个流表项缓存层。每个交换机对应一个缓存器,多个缓存器共同构成缓存层。为了充分利用缓存空间,每个缓存器大小根据流表项数量进行动态分配,当流表中的流表项因为缺乏存储空间或者超时而被删除时,该流表项会被存储在相应的缓存器中,而每当缓存器收到一个流表存储请求,就申请一个缓存单元用于存储该流表项。此时当 Open-

Flow 交换机收到 PACKET_IN 路由请求数据包时,首先会在缓存器中进行匹配,如果存在对应流表项,则由控制器将该流表项装载到交换机中,否则将发起一次路由计算请求。

在各个缓存器中,老鼠流和大象流被分配独立的缓存单元,虽然整个缓存器的存储空间有限,但是可以根据需求进行动态分配。为了快速查找缓存中的流表项,使用动态指数哈希将流映射到缓存单元,并且建立一张映射目录,目录中直接保存缓存单元的地址。流的 m 位二进制哈希指数是动态的,而 m 的大小取决于用于该类型的流的缓存单元的数量,每个存储单元都有一个 m 位二进制标记,该存储单元中存储的流就是拥有相同标记的流。安全哈希算法 (SHA) 用来产生随机 k 位哈希指数,从而能够保证缓存器中存储的均衡性。如图 3 所示,流表项的缓存过程如下:

- (1)使用 SHA 对流的包头特征进行计算得到哈希值,SHA 的优点是能够随机地将流映射到 160 位二进制存储空间中。此时,哈希值的前 n 位 (n 是当前缓存器的最大深度, 2^n 是其存储容量) 被提出来并加上以为标志前缀,其中“0”表示老鼠流,“1”表示大象流。
- (2)根据得到的 $n+1$ 位二进制索引,查询目录获得缓存器单元地址。
- (3)查找对应缓存单元地址,如果未命中则发起一次路由计算请求并将最优路径对应的流表项存入该缓存单元中。如果缓存单元已满,则将缓存器中存储单元的个数翻倍。

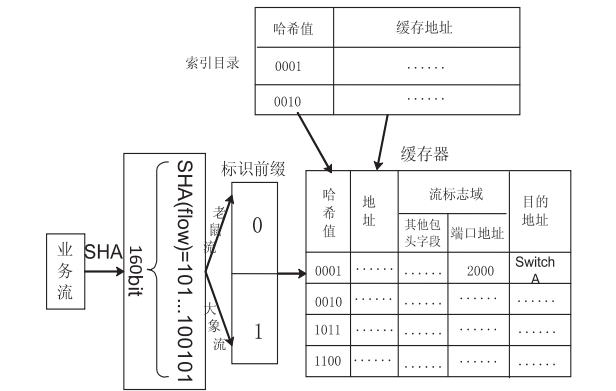


图 3 基于动态索引哈希的流缓存机制

缓存空间的大小相比并发业务流的数量较小,因此需要应用缓存替换策略。为了公平对待老鼠流和大象流,规定大象流仅能替换大象流,而老鼠流也仅能替换老鼠流。采用本地最近最久未使用 (LRU) 替换策略,当缓存未命中或缓存单元满的时候最近最长时间未被命中过的流表项将被移除。

4 实验结果及分析

首先实现系统原型,然后从不同的方面来考察文

中方案的性能。为了搭建仿真环境,使用 Mininet 2.2.0^[14]网络仿真工具搭建实验网络,利用开源控制器 Floodlight 3.5 构建控制器平面^[15],使用支持多层虚拟交换的 OpenVSwitch 2.3.1 作为转发设备,而缓存层被作为控制器模块来实现。实验网络如图 4 所示。

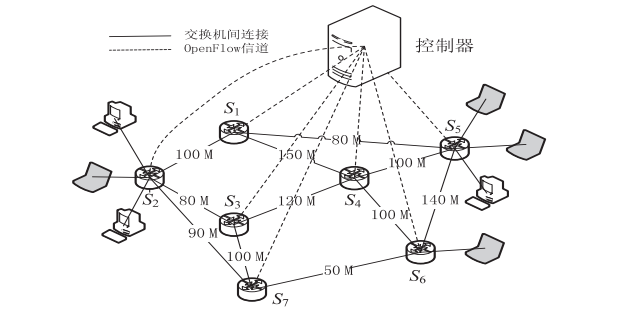


图 4 实验网络拓扑

4.1 通配符流表项对延迟的影响

测试了安装通配符流表项对主机间传输时延和单位时间内流表项数量的影响,如图 5 所示。在通配符流表项建立之前 (0 ~ 15 s),相距跳数为 10 的主机对的平均时延大约为 850 ms,而当在对应路径上安装优先级较高的通配符流表项后,时延很快降低至 10 ms。可以看出,当通配符流表项被建立以后,大量的老鼠流被交换机直接转发而不经控制器平面,在降低控制器负载的同时,大大提高了数据的传输效率。单位时间内的流表项数也表现出了类似的特性,在前 15 s 内,随着流量的增加流表项数量也急剧增加,而当通配符流表项建立以后,流表项数目随着时间平滑下降。

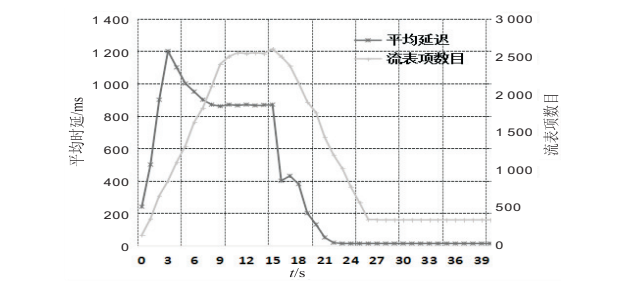


图 5 通配符流表项对时延的影响

4.2 缓存命中率对比

将文中提出的流缓存策略与基于 OpenFlow 通配符感知的线性搜索表结合先进先出 (FIFO) 及随机替换策略的方式进行对比。FIFO 将最先到达的流表项删除,而随机替换策略是随机选择一个流表项并丢弃。为了模拟数据中心的流量环境,使用 Mausezahl 生成含有 10% 大象流和 90% 老鼠流的流量模式。LRU、基于通配符线性搜索表的 FIFO 和随机丢弃三种方式的对比结果如图 6 所示。

显然 LRU 的性能要好于 FIFO 和随机替换算法,并且尽管老鼠流数量巨大,但在文中提出的流缓存方案中,老鼠流和大象流使用独立的缓存器,老鼠流不会

挤占大象流的缓存空间,因此并未对大象流造成明显的影响。

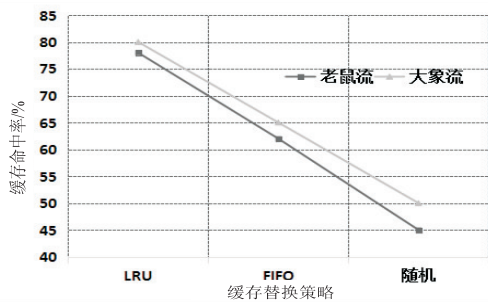


图6 缓存命中率对比

4.3 缓存查找速度对比

将文中提出的动态索引哈希和基于 OpenFlow 通配符感知的线性搜索表进行对比,如图7所示。动态索引哈希的查找速度相比于线性搜索有了较大提升,且单位时间内查找的次数越多,查找速度的差距越明显。从理论上来说,基于通配符的线性搜索表对于大象流和老鼠流的查找速度并非是相同的,原因在于其采用的是步进查找的方式,而老鼠流表项的数量要多于大象流表项的数量。可以看出,无论在命中率还是查找速度上,文中提出的缓存策略都具有明显优势,而且使老鼠流对大象流的负面影响降到最低。

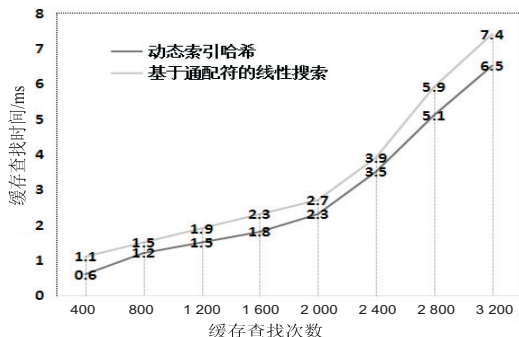


图7 缓存查找速度对比

5 结束语

针对SDN应用于数据中心网络中遇到的可扩展性问题,从两个方面提出了改进措施。为了解决数据中心内部存在大量老鼠流导致的控制器性能下降的问题,提出区分老鼠流,并为其建立以目的地址为导向的通配符流表项,大大降低了传输时延和流表项数量;针对流表空间有限性和流表超时带来的路由重复请求问题,在控制器平面和转发平面之间增加了基于动态索引哈希的缓存层,给出了详细的实现过程。实验结果表明其具有较高的命中率和查找速度。

参考文献:

[1] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild[C]//ACM SIGCOMM conference

on internet measurement 2010. Melbourne, Australia: ACM, 2010:114-115.

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

[3] Curtis A R, Mogul J C, Tourrilhes J, et al. DevoFlow: scaling flow management for high-performance networks[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 254-265.

[4] Chandra R, Bahl P, Bahl P. MultiNet: connecting to multiple IEEE 802.11 networks using a single wireless card[C]//Twenty-third annual joint conference of the IEEE computer and communications societies. [s.l.]: IEEE, 2004: 882-893.

[5] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: dynamic flow scheduling for data center networks[C]//USENIX symposium on networked systems design and implementation. San Jose, Ca, USA: USENIX, 2010: 19.

[6] Curtis A R, Kim W, Yalagandula P. Mahout: low-overhead datacenter traffic management using end-host-based elephant detection[C]//INFOCOM. [s.l.]: IEEE, 2011: 1629-1637.

[7] Zadnik M, Canini M. Evolution of cache replacement policies to track heavy-hitter flows[M]//International conference on passive and active measurement. [s.l.]: IEEE, 2011: 1-2.

[8] Yu M, Rexford J, Freedman M J, et al. Scalable flow-based networking with DIFANE[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 351-362.

[9] Pan T, Guo X, Zhang C, et al. ALFE: a replacement policy to cache elephant flows in the presence of mice flooding[C]//IEEE international conference on communications. [s.l.]: IEEE, 2012: 2961-2965.

[10] Ravikumar V C, Mahapatra R N. TCAM architecture for IP lookup using prefix properties[J]. IEEE Micro, 2004, 24(2): 60-69.

[11] Wu W, Ji D, Lan Y, et al. Power-aware TCAMs for routing table lookup[C]//Green computing and communications. [s.l.]: IEEE, 2010: 425-429.

[12] Zane F, Narlikar G, Basu A. Coolcams: power-efficient TCAMs for forwarding engines[C]//Joint conference of the IEEE computer and communications. [s.l.]: IEEE, 2003: 42-52.

[13] Farrington N, Porter G, Radhakrishnan S. Helios: a hybrid electrical/optical switch architecture for modular data centers[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 339-350.

[14] Oliveira R L S D, Schweitzer C M, Shinoda A A, et al. Using mininet for emulation and prototyping software-defined networks[C]//IEEE Colombian conference on communications and computing. [s.l.]: IEEE, 2014: 1-6.

[15] Chiu H W, Wang S Y. Boosting the OpenFlow control-plane message exchange performance of OpenvSwitch[C]//IEEE international conference on communications. [s.l.]: IEEE, 2015.