软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(5):1078−1097 [doi: 10.3724/SP.J.1001.2013.04390]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 OpenFlow 的 SDN 技术研究*

左青云, 陈 鸣, 赵广松, 邢长友, 张国敏, 蒋培成

(解放军理工大学 指挥信息系统学院,江苏 南京 210007)

通讯作者: 左青云, E-mail: zuoqy@163.com

摘 要: 软件定义网络(software-defined networking,简称 SDN)技术分离了网络的控制平面和数据平面,为研发网络新应用和未来互联网技术提供了一种新的解决方案.综述了基于 OpenFlow 的 SDN 技术发展现状,首先总结了逻辑控制和数据转发分离架构的研究背景,并介绍了其关键组件和研究进展,包括 OpenFlow 交换机、控制器和 SDN 技术,然后从 4 个方面分析了基于 OpenFlow 的 SDN 技术目前所面临的问题和解决思路.结合近年来的发展现状,归纳了在校园网、数据中心以及面向网络管理和网络安全方面的应用,最后探讨了未来的研究趋势.

关键词: OpenFlow;未来互联网;控制器;虚拟化;软件定义网络

中图法分类号: TP393 **文献标识码**: A

中文引用格式: 左青云,陈鸣,赵广松,邢长友,张国敏,蒋培成.基于 OpenFlow 的 SDN 技术研究.软件学报,2013,24(5):1078−1097. http://www.jos.org.cn/1000-9825/4390.htm

英文引用格式: Zuo QY, Chen M, Zhao GS, Xing CY, Zhang GM, Jiang PC. Research on OpenFlow-based SDN technologies. Ruan Jian Xue Bao/Journal of Software, 2013,24(5):1078−1097 (in Chinese). http://www.jos.org.cn/1000-9825/4390.htm

Research on OpenFlow-Based SDN Technologies

ZUO Qing-Yun, CHEN Ming, ZHAO Guang-Song, XING Chang-You, ZHANG Guo-Min, JIANG Pei-Cheng

(College of Command Information Systems, PLA University of Science and Technology, Nanjing 210007, China)

Corresponding author: ZUO Qing-Yun, E-mail: zuoqy@163.com

Abstract: The control and data planes are decoupled in software-defined networking, which provide a new solution for research on new network applications and future Internet technologies. The development status of OpenFlow-based SDN technologies is surveyed in this paper. The research background of decoupled architecture of network control and data transmission in OpenFlow network is summarized first, and the key components and research progress including OpenFlow switch, controller, and SDN technologies are introduced. Moreover, current problems and solutions of OpenFlow-based SDN technologies are analyzed in four aspects. Combined with the development status in recent years, the applications used in campus, data center, network management and network security are summarized. Finally, future research trends are discussed.

Key words: OpenFlow; future Internet; controller; virtualization; software-defined networking (SDN)

随着网络规模的急剧膨胀和应用类型的不断丰富,因特网作为社会基础设施至关重要的一部分,结构和功能日趋复杂,管控能力日趋减弱.尤其作为网络核心的路由器,承载功能不断扩展,如分组过滤、区分服务、多播、服务质量(QoS)、流量工程等,路由器最初定义的“哑的、简单的”数据转发单元已经变得臃肿不堪.从路由器当前主要厂商的发展趋势来看,性能提升和功能扩展依然是其主要研发目标.而出于对自身技术和市场占有率的

* 基金项目: 国家重点基础研究发展计划(973)(2012CB315806); 国家自然科学基金(61070173, 61103225); 江苏省自然科学基金(BK2010133)

收稿时间: 2012-05-03; 修改时间: 2012-11-06, 2013-01-07; 定稿时间: 2013-02-05; jos 在线出版时间: 2013-03-29

CNKI 网络优先出版: 2013-03-29 15:19, http://www.cnki.net/kcms/detail/11.2560.TP.20130329.1519.004.html

考虑,路由器只能通过命令行接口(command-line interface,简称 CLI)等方式对外开放少量功能,研究人员难以在真实的网络中实验和部署新型网络体系结构和网络技术.当前网络中所大量部署的网络设备和网络协议也将在相当长一段时间内延续现有的网络体系结构[1].

为了解决现有 TCP/IP 体系结构面临的诸多难题,世界各国已经大规模开展未来互联网的研究,如美国的 GENI[2]、欧盟的 FIRE[3]、日本的 JGN2plus[4]和我国的 SOFIA[5]等.研究未来互联网体系结构首先考虑的是网络核心设备路由器的重新设计和部署,允许用户自行定义路由器功能模块,实现适应未来互联网发展的新型协议功能.目前,可编程虚拟化路由器已经受到广泛关注,而设计的开放性和可控性将决定其设计思想是否可以得到长远的发展.OpenFlow[1]技术概念最早由斯坦福大学的 Nick McKeown 教授提出,是斯坦福大学 Clean Slate 计划资助的一个开放式协议标准,后成为 GENI 计划的子项目.OpenFlow 将控制功能从网络设备中分离出来,在网络设备上维护流表(flow table)结构,数据分组按照流表进行转发,而流表的生成、维护、配置则由中央控制器来管理.OpenFlow 的流表结构将网络处理层次扁平化,使得网络数据的处理满足细粒度的处理要求.在这种控制转发分离架构下,网络的逻辑控制功能和高层策略可以通过中央控制器灵活地进行动态管理和配置,可在不影响传统网络正常流量的情况下,在现有的网络中实现和部署新型网络架构.

OpenFlow 最初是为校园网络研究人员设计其创新网络架构提供真实的实验平台,随后,McKeown 等研究者开始推广 SDN 概念,并引起学术界和产业界的广泛关注.SDN 将网络配置平面从嵌入式结点中独立出来,以开放软件模式的控制平面替代了传统基于系统嵌入的控制平面,由软件驱动的中央控制结点来自动化控制整个网络.OpenFlow 初步实现了 SDN 的原型设计思想,推动了 SDN 技术的快速发展,简化了网络的配置模式,增加了网络控制权的开放性,在某种程度上符合未来互联网的发展需求,也是目前 SDN 最通用的实现方式.基于 OpenFlow 的 SDN 技术通过软件平台来打造弹性化的可控互联网,在给网络的发展带来巨大冲击的同时,也为未来互联网的发展提供了一种新的解决思路.

本文第 1 节介绍逻辑控制和数据转发分离架构的研究背景.第 2 节综述基于 OpenFlow 的 SDN 关键组件与研究进展.第 3 节总结分析目前面临的问题和解决方案.第 4 节系统地给出当前的典型应用.第 5 节总结全文并探讨未来的研究重点和发展趋势.

1 SDN 研究背景:逻辑控制和数据转发分离

传统因特网把控制逻辑和数据转发紧耦合在网络设备上,导致网络控制平面管理的复杂化,也使得网络控制层面新技术的更新和发展很难直接部署在现有网络上,灵活性和扩展性很难适应网络的飞速发展.网络的控制转发分离架构提出由专有设备来部署高层策略,网络设备在高层策略指导下进行数据转发,减少了网络设备承载的诸多复杂功能,提高了网络新技术和新协议实现和部署的灵活性和可操作性.逻辑控制和数据转发分离的这种管控思想是 SDN 技术的研究基础,前期已经在学术界引起较大关注,典型工作包括 ForCES[6]、4D[7]架构、RCP[8]、SANE[9]和 Ethane[10].

IETF 的 ForCES(forwarding and control element separation)[6]基于开放可编程思想的网络体系结构,将网络元素分为控制件(control element,简称 CE)和转发件(forwarding element,简称 FE),用 ForCES 协议来实现各部件的协同和交互,以提高网络的可管可控功能,增强网络部署的灵活性和有效性.ForCES 虽然已经提交了多项 RFC 标准草案,但主要研究工作在于理论创新和功能建模,并没有面向真实网络的部署和实践.

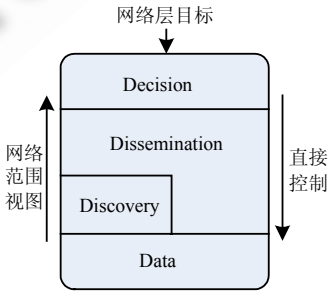


Fig.1 4D architecture

图 1 4D 架构

针对当前网络的逻辑决策平面和分布式的硬件设备结合过紧的问题, Greenberg 等人重新设计了互联网控制和管理结构,提出了如图 1 所示的 4D(decision,dissemination,discovery,data)的体系架构[7].在 4D 架构下,决策平面通过全局网络视图做出网络控制决策,并直接下发到数据平面;分发平面在决策平面和路由器之间

建立可靠的通信通道;发现平面负责发现网络中的物理组件,并为决策平面提供构建网络视图的基本信息;数据平面则实现数据转发功能.这种架构有助于实现健壮、安全的网络,便于对异构网络进行有效管理.

RCP(routing control platform)[8]是Caesar等人提出的基于AS结构的逻辑中央平台.它针对内部边界网关协议(internal border gateway protocol,简称iBGP)扩展性不强和容易造成协议不稳定、路由回路等缺点,部署路由控制服务器集中收集路由信息和AS内部拓扑结构信息,为AS范围内的路由器做出BGP路由决策.RCP通过实现的原型系统验证了控制转发分离架构的可行性,与4D架构的设计思想相比,RCP仅实现了BGP路由决策的集中管控.它在逻辑控制平面功能上还有很大的扩展空间.

SANE[9]和Ethane[10]是Casado等主要来自斯坦福大学的研究人员分别于2006年和2007年提出的面向企业网的管理架构.SANE主要面向企业网的安全管理,它在链路层和IP层之间定义了一个可以管理所有连接的保护层,所有路由和接入控制决策都通过这个保护层由一台逻辑中央服务器控制.SANE以4D架构为设计原则,由于主要面向企业网,因此其重点在于安全控制,还没有实现复杂的路由决策,同时没有经过大规模测试,实际部署还比较困难.Ethane在SANE的基础上进行了功能扩展,将安全管理策略添加到网络管理当中,扩充了中央控制器的管理功能,实现了更细粒度的流表转发策略.在Ethane网络中,中央控制器和Ethane交换机是两个主要部件.中央控制器实现网络主机认证、IP分配和产生交换机流表等基本功能,是整个网络的控制决策层.Ethane交换机根据控制器部署的流表进行报文转发,是一个简单的、哑的数据转发单元.Ethane实现了OpenFlow交换机和中央控制器的大部分功能,奠定了OpenFlow的技术基础.

针对目前日益增长的带宽需求,Casado重新总结了分组交换网络硬件的设计思想[11].现有路由器包含了报文转发决策的功能,复杂性过高,导致灵活性和有效性不够.因此,应当简化硬件转发功能,用软件来实现转发决策.硬件和软件之间应该相互独立,硬件只需缓存这些决策,而专注于报文的转发.由此引发的问题是,硬件和软件之间的交互和速度匹配需要考虑实际的硬件性能和网络流量的大小,报文处理在硬件中的匹配率将影响到软件的实际性能.这种软硬件功能分离的思想进一步明确了软件层做决策和硬件层控制分组转发的设计思路.

从以上相关文献可以看出,逻辑控制和数据转发分离架构的设计思想简化了网络管理和配置操作,实现了高层控制逻辑的健壮性,有利于异构网络的融合和创新应用的部署.但同时我们也必须看出,这种架构目前还将面临一些新的挑战,如逻辑控制平面的功能集中化和可扩展性问题、逻辑决策的实时响应和一致性问题、逻辑控制单元的失效引发的容错和恢复机制问题,都需要进行大量深入研究.而在实际应用方面,这些控制转发分离架构的相关研究主要集中于理论上,或者通过小范围的实验部署进行测试,缺乏支持复杂网管功能以及较大规模网络的部署经验,因此也缺乏更准确的性能测试数据.

在上述相关工作的基础上,斯坦福大学的研究者于2008年提出了OpenFlow技术[1],并逐渐推广SDN概念.OpenFlow作为SDN的原型实现方式,代表了SDN控制转发分离架构的技术实现.随着SDN技术得到认可,从严格定义上来讲,OpenFlow指的是SDN控制平面和数据平面之间多种通信协议之一,但实际上,OpenFlow以其良好的灵活性、规范性已被看作SDN通信协议事实上的标准,类似于TCP/IP协议作为互联网的通信标准.

基于OpenFlow的SDN技术推出之后,2009年被MIT评为十大前沿技术[12].2011年,McKeown等研究者组织成立开放式网络基金会(Open Networking Foundation,简称ONF)[13],专门负责相关标准的制定和推广,包括OpenFlow标准、OpenFlow配置协议和SDN白皮书,极大地推进了OpenFlow和SDN的标准化工作,也使其成为全球开放网络架构和网络虚拟化领域的研究热点.在学术界,美国GENI[2]、Internet2[14]、欧洲OFELIA[15]和日本的JGN2plus[4]先后展开对SDN的研究和部署,IETF,ITU,ETSI等标准组织开始关注SDN,讨论SDN在各自领域可能的发展场景和架构应用.在产业界,以Nicira[16](创始人实际为McKeown和Casado等人,已被VMware收购)和Big Switch[17]为代表的SDN创业公司不断涌现.为了在新技术上抢占先机,出于对自身市场推广的考虑,Juniper,NEC,HP,IBM等厂商先后发布了支持OpenFlow的SDN硬件,目前也在SDN研究领域进行了相关部署.Google宣布其广域网数据中心已经大规模使用基于OpenFlow的SDN技术,以实现其数据中心的流量工程和实时管控功能[18],为OpenFlow的应用和推广起着示范作用.Cisco沉寂多时后,在Cisco Live 2012上宣布了其开放网络环境(open network environment,简称Cisco ONE)计划,涵盖了从传输到管理和协调的整套解决方案,并

补充了当前的软件定义方法.Cisco ONE 包括 One Platform Kit(onePK),为开发人员提供思科各种路由器和交换机操作系统的通用 API,也发布了适用于 SDN 研发的概念验证控制器和 OpenFlow 代理,同时将其 Nexus 1000V 虚拟交换机上作为虚拟覆盖网络的基础设备.Cisco 为了维护目前的垄断地位,在推广 SDN 技术时主要面向自身的专利软件和技术规范,但同时也将 OpenFlow 看作面向学术研究市场的实现方式.

可以看出,基于 OpenFlow 的 SDN 技术打破了传统网络的分布式架构,颠覆了传统网络的运行模式,在实现方式上与上述文献的要求不完全相同,在面临类似挑战时还需要满足新的技术和市场需求,目前,学术界和产业界已经展开大量研究来寻找解决方案.

2 基于 OpenFlow 的 SDN 关键组件及架构

OpenFlow 最初作为 SDN 的原型提出时,主要由 OpenFlow 交换机、控制器两部分组成.OpenFlow 交换机根据流表来转发数据包,代表着数据转发平面;控制器通过全网络视图来实现管控功能,其控制逻辑表示控制平面.随着 SDN 概念的不断推广,ONF 也对 SDN 的定义和架构进行了详细介绍,进一步论述了 OpenFlow 和 SDN 的相互关系.下面首先介绍基于 OpenFlow 的 SDN 关键组件,包括 OpenFlow 交换机和控制器,然后对 SDN 的技术架构进行详细说明.

2.1 OpenFlow交换机

OpenFlow 交换机负责数据转发功能,主要技术细节由 3 部分组成[1]:流表(flow table)、安全信道(secure channel)和 OpenFlow 协议(OpenFlow protocol),如图 2 所示.

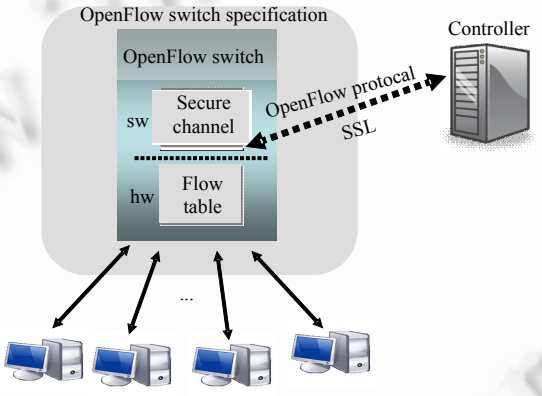


Fig.2 Structure of OpenFlow switch

图 2 OpenFlow 交换机结构

每个 OpenFlow 交换机的处理单元由流表构成,每个流表由许多流表项组成,流表项则代表转发规则.进入交换机的数据包通过查询流表来取得对应的操作.为了提升流量的查询效率,目前的流表查询通过多级流表和流水线模式来获得对应操作.流表项主要由匹配字段(match field)、计数器(counter)和操作(instruction)这 3 部分组成.匹配字段的结构包含很多匹配项,涵盖了链路层、网络层和传输层大部分标识.随着 OpenFlow 规约的不断更新,VLAN,MPLS 和 IPv6 等协议也逐渐扩展到 OpenFlow 标准当中.由于 OpenFlow 交换机采取流的匹配和转发模式,因此在 OpenFlow 网络中将不再区分路由器和交换机,而是统称为 OpenFlow 交换机.另外,计数器用来对数据流的基本数据进行统计,操作则表明了与该流表项匹配的数据包应该执行的下一步操作.

安全通道是连接 OpenFlow 交换机和控制器的接口,控制器通过这个接口,按照 OpenFlow 协议规定的格式来配置和管理 OpenFlow 交换机.目前,基于软件实现的 OpenFlow 交换机主要有两个版本[19],都部署于 Linux 系统:基于用户空间的软件 OpenFlow 交换机操作简单,便于修改,但性能较差;基于内核空间的软件 OpenFlow 交换机[20]速度较快,同时提供了虚拟化功能,使得每个虚拟机能够通过多个虚拟网卡传输流量,但实际的修改和操

作过程较复杂.另外,斯坦福大学基于 NetFPGA 实现了硬件加速的线速 OpenFlow 交换机[21],而网络硬件厂商如 NEC,HP 等公司也已相继推出了支持 OpenFlow 标准的硬件交换机.

2.2 控制器

在控制器中,网络操作系统(network operating system,简称 NOS)实现控制逻辑功能.NOX[22]最早引入这个概念,是 OpenFlow 网络中对网络实现可编程控制的中央执行单元.实际上,这里的 NOS 指的是 SDN 概念中的控制软件,通过在 NOS 上运行不同的应用程序能够实现不同的逻辑管控功能.

在基于 NOX 的 OpenFlow 网络中,NOX 是控制核心,OpenFlow 交换机是操作实体,如图 3 所示.NOX 通过维护网络视图(network view)来维护整个网络的基本信息,如拓扑、网络单元和提供的服务,运行在 NOX 之上的应用程序通过调用网络视图中的全局数据,进而操作 OpenFlow 交换机来对整个网络进行管理和控制.从 NOX 控制器完成的功能来看,NOX 实现了网络基本的管控功能,为 OpenFlow 网络提供了通用 API 的基础控制平台,但在性能上并没有太大的优势,未能提供充分的可靠性和灵活性来满足可扩展的需求.但是,NOX 在控制器设计方面实现得最早,目前已经作为 OpenFlow 网络控制器平台实现的基础和模板.

为使控制器能够直接部署在真实网络中,解决多控制器对 OpenFlow 交换机的控制共享问题,同时满足网络虚拟化的现实需求,FlowVisor[23]在控制器和 OpenFlow 交换机之间实现了基于 OpenFlow 的网络虚拟层.它使得硬件转发平面能够被多个逻辑网络切片(slice)共享,每个网络切片拥有不同的转发逻辑策略.在这种切片模式下,多个控制器能够同时管理一台交换机,多个网络实验能够同时运行在同一个真实网络中,网络管理者能够并行地控制网络,因此网络正常流量可以运行在独立的切片模式下,从而保证正常流量不受干扰(如图 4 所示).

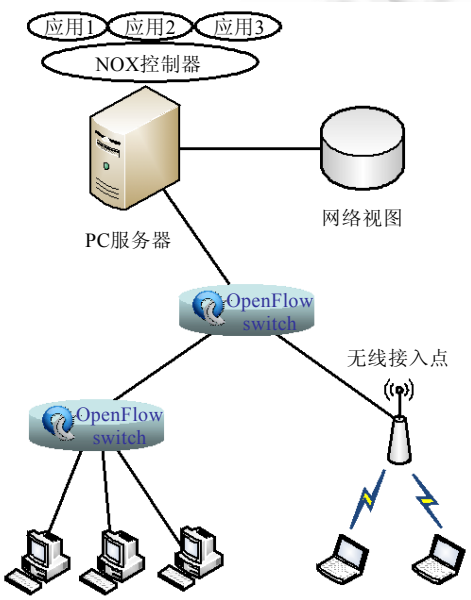


Fig.3 NOX-Based OpenFlow network

图 3 基于 NOX 的 OpenFlow 网络

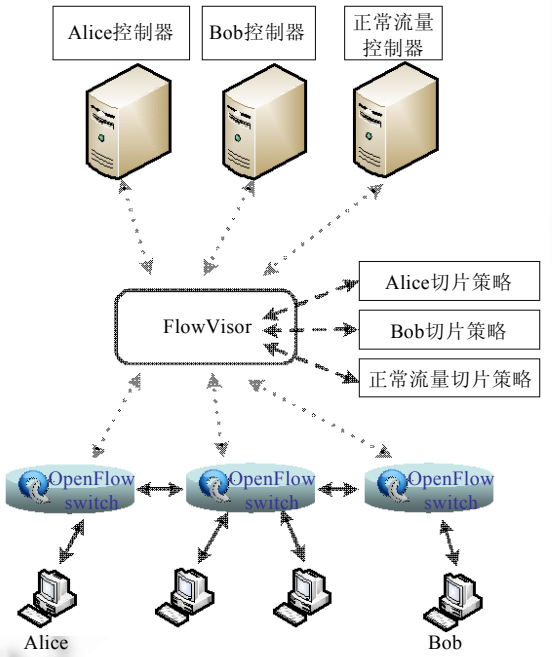


Fig.4 FlowVisor-Based OpenFlow virtualization

图 4 基于 FlowVisor 的 OpenFlow 虚拟化

目前,支持 OpenFlow 协议的多种控制软件已经得到了开发和推广.NOX 已经发布了多个版本[24],如 NOX Destiny,NOX Zach,POX 等.它们对 NOX 进行了性能上的优化,并逐渐支持更多的功能,如控制台操作、SNMP 控制等.其余的控制软件也得到了广泛应用,如 Maestro[25],Beacon[26],Floodlight[27],SNAC[28],Trema[29],RouteFlow[30]和 Onix[31]等.表 1 列举了当前广受关注的 OpenFlow 网络控制器的研究背景和主要技术特点.

Table 1 Comparison of control softwares in OpenFlow network

表 1 OpenFlow 网络控制软件对比

控制器	研究背景介绍			主要技术特点
	开发语言	平台	开发团队	
NOX	Python/C++	Linux	Nicira	最早实现的控制器,当前控制器的模板平台,单线程操作,版本还在不断更新,性能还在继续完善
Maestro	Java	Win/Mac/Linux	Rice	跨平台,易于开发和部署,支持多线程操作,功能丰富,可以加入新的应用程序进行扩展
Beacon	Java	Win/Mac/Linux	Stanford	跨平台,易于开发和部署,采用模块化功能实现了基于事件和多线程操作的处理平台,可定制易于扩展的界面框架
Floodlight	Java	Win/Mac/Linux	Big switch	跨平台,基于 Beacon 开发,由开源社区进行维护,遵循 Apache 开源规范,适合推广
SNAC	Python/C++	Linux	Nicira	基于 NOX 平台开发,集成了可扩展的策略定义语言,通过策略管理器来管理网络,用户界面友好
Trema	Ruby/C	Linux	NEC	为开发者提供了丰富的 API,让开发者轻松构建自己的控制器平台.可添加自定义功能模块,测试和调试工具丰富
RouteFlow	C++	Linux	CPqD (Brazil)	基于 NOX 开发,采用 Quagga 设备进行虚拟化,实现虚拟网络环境
Onix	Python/C++/Java	Linux	Nicira	利用分布式系统实现中央控制平面,面向较大规模真实网络的部署方案

2.3 SDN

因特网的高速发展可以归结于细腰的 TCP/IP 架构和开放的应用层软件设计,但从网络核心来讲,由于专有的硬件设备和操作系统,网络在很大程度上是封闭的.SDN 将控制功能从传统的分布式网络设备中迁移到可控的计算设备中,使得底层的网络基础设施能够被上层的网络服务和应用程序所抽象,最终通过开放可编程的软件模式来实现网络的自动化控制功能.

OpenFlow 实现了 SDN 可编程网络的思想,代表了 SDN 技术的实现原型和部署实例.但从整个 SDN 架构来看,OpenFlow 特指控制平面和数据平面的某一种通信协议.图 5 描述了 SDN 架构的逻辑视图[13],主要分为基础设施层、控制层和应用层.基础设施层表示网络的底层转发设备,包含了特定的转发面抽象(如 OpenFlow 交换机中流表的匹配字段设计).中间的控制层集中维护网络状态,并通过南向接口(控制和数据平面接口,如 OpenFlow)获取底层基础设施信息,同时为应用层提供可扩展的北向接口.目前,ONF 仍在制定和完善南向接口 OpenFlow 协议,面向应用的可编程北向接口仍处在需求讨论阶段.应用层根据网络不同的应用需求,调用控制层的北向接口,实现不同功能的应用程序.通过这种软件模式,网络管理者能够通过动态的 SDN 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,这也是 SDN 开放性和可编程性最重要的体现.

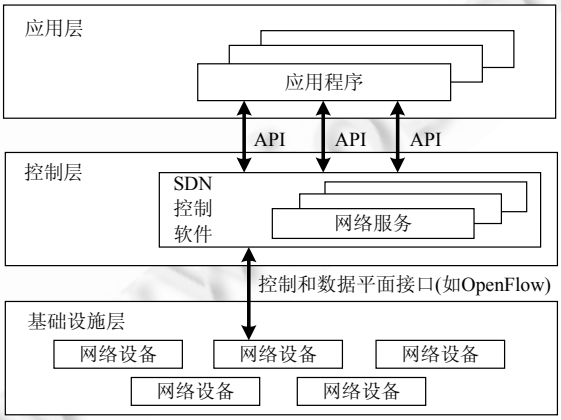


Fig.5 SDN architecture

图 5 SDN 架构

在 SDN 的这种 3 层架构下,网络的运行维护仅需通过软件的更新来实现网络功能的升级,网络配置将通过网络服务和应用程序的形式直接得到部署,网络管理者无须再针对每一个硬件设备进行配置或者等待网络设备厂商硬件的发布,从而加速网络部署周期.同时,SDN 降低了网络复杂度,使得网络设备从封闭走向开放,底层的网络设备能够专注于数据转发而使得功能简化,有效降低网络构建成本.另一方面,传统网络中的结点只能通过局部状态和分布式算法来实现数据转发,因而很难达到最优性能.SDN 通过软件来实现集中控制,使得网络具备集中协调点,因而能够通过软件形式达到最优性能,从而加速网络创新周期.

3 基于 OpenFlow 的 SDN 面临的问题和解决思路

虽然目前基于 OpenFlow 的 SDN 已经引起较大的关注,但无论是 OpenFlow 协议本身,还是 SDN 这种管控分离架构,不仅在技术上面临着许多还未解决的问题,在具体的运作模式和演进趋势上也和当前网络设备厂商的生产理念相违背,这使得其大规模应用还需要等待技术的成熟和市场的推广.目前来看,基于 OpenFlow 的 SDN 技术遇到的主要问题包括以下几点:

(1) SDN 转发平面的设计问题.OpenFlow 交换机作为 SDN 转发平面抽象的实际载体,协议标准处在不断更新当中.随着 OpenFlow 规约的不断发布,OpenFlow 交换机流表从最初的单表结构变为多表结构,流表项匹配字段从最初的十元组到支持 IPv6,MPLS 等,这些都表明 SDN 转发平面功能的逐渐扩展,意味着 OpenFlow 交换机结构设计的复杂化,因此必须认识到由此带来的新的问题和挑战.

(2) 控制平面的可扩展性.OpenFlow 设计之初,仅需通过单控制器来实现网络的管控功能.显然,随着网络规模的增大和业务需求的增加,需要研究控制平面的可扩展性解决方案,即多控制器解决方案.而控制单元的数量和它们之间网络状态(包括拓扑、传输能力、路由限制等)的协同和交互应该如何实现,以保证网络状态的一致性和可扩展性,还需要进行大量深入的研究.

(3) SDN 控制逻辑的一致性.虽然控制平面能够将控制逻辑集中部署到整个网络,但数据平面转发设备仍然是一个分布式系统.控制逻辑的先后配置顺序以及控制平面和数据平面之间可能存在的时延将难以保证控制逻辑更新的一致性,有可能造成网络出现断路、丢包、环路等现象.

(4) 运作模式和演进趋势问题.SDN 技术颠覆了网络设备的设计理念,带来了新的市场需求,同时也对传统的网络设备制造商提出了挑战.同时,OpenFlow 自身设计标准的不稳定性和转发设备硬件的复杂化趋势也为 OpenFlow 的演进趋势带来了不确定性.

根据上面提到的这些技术问题和需求,下面分别就当前的热点难点问题和相关的研究思路进行总结.

3.1 SDN转发平面的设计问题

OpenFlow 交换机作为目前 SDN 转发平面的实现方式,已被广为接受.由于 OpenFlow 交换机采用流表结构处理数据分组,因此特定的应用有可能导致交换机流表迅速膨胀,从而影响 OpenFlow 交换机的性能.从目前实际部署来看,基于 NetFPGA 的 OpenFlow 交换机[21]在性能上可以满足校园内的流量需求,但在实现上,当通配符查询耗尽 NetFPGA 硬件资源后,OpenFlow 交换机将以线性查找的软件方式对流表进行匹配查找.这种方式在网络规模增大、细粒度流表需求增加的网络环境下将产生性能瓶颈.

这样的性能瓶颈同样会发生在商用硬件 OpenFlow 交换机上,其主要原因在于 OpenFlow 转发抽象对实际硬件资源的过度消耗.2008 年,OpenFlow 规约 0.8.9 版只支持最初的十元组,随后几年中,0.9 版增加了 VLAN 优先级,1.0 版增加了 IP 服务类型,1.1 版增加了元数据(metadata)、MPLS 标签和 MPLS 流量类别这 3 个字段,1.2 版和 1.3 版则列出了需要实现的 13 个字段和可选实现的众多字段,前 13 个字段里包含 IPv6 字段.可以看出,为了实现功能的扩展,流表项的匹配长度一直在增加.路由器通常利用 TCAM 来检索路由器转发表,从而获得下一跳地址.传统的转发表匹配字段长度一般以 IPv4 的 32 位为主,而 OpenFlow 规约 1.0 版里的十二元组长度为 250 比特左右,OpenFlow1.1 版里则长达 356 比特.在增加了 IPv6 之后,流表项将更长.这种不断扩充的转发平面抽象,将极大地增加硬件成本.OpenFlow1.1 版提出了多级流表和流水线结构,通过分域多级流表查询来压缩流表空间,在一定程度上减少 OpenFlow 交换机中 TCAM 资源的消耗,从而给报文的匹配操作增加了足够的灵活性.但

需要看到,实际的流表硬件资源开销将与具体的流表设计有关,需要根据实际的转发需求设计跳转算法,优化内部流表查找过程.这超出了传统交换芯片的数据转发处理过程,将导致匹配时延的增加,也将增加 OpenFlow 交换机硬件设计的复杂度.

另一方面,控制平面需要知晓每个 OpenFlow 交换机的流水线结构,这需要控制平面进行相应的协议扩展.然而,OpenFlow 规约标准还在不断更新,更多的扩展功能将添加到 OpenFlow 标准当中.由于目前尚没有一个成熟统一的标准发布,这也导致不同版本 OpenFlow 交换机的流表结构不尽相同.随着不同版本的 OpenFlow 交换机逐渐投入使用,控制平面需要维护异构的 OpenFlow 交换机流表结构,这无疑将增加控制平面的维护成本和复杂性.这也是未来 OpenFlow 交换机设计需要考虑的问题.

3.2 控制平面的可扩展性

制约 SDN 控制平面可扩展性的主要原因有以下几点:

(1) 流的细粒度处理需求使得控制器需要响应更多的流请求事件.虽然控制器可以通过主动决策机制提前将控制逻辑部署到数据转发单元,减少数据平面和控制器之间的处理开销,但控制逻辑的变化通常是动态的,尤其是当网络拓扑改变或者存在移动结点时.在 OpenFlow 网络中,提前安装流表项也将使大量流表空间无法释放,浪费资源,而实际上大部分流的持续时间是很短的.

(2) 接入控制、负载均衡、资源迁移等新型应用需求逐渐增加到控制平面当中,控制器需要对日趋复杂的管控功能进行有效的整合,这进一步增加了控制平面的处理开销.

(3) 传统分布式网络设备仅根据局部的路由信息来实现路由转发,而控制平面需要维护全局的网络状态信息,这也使得控制平面的可扩展性不仅需要考虑性能的需求,而且要考虑网络状态的一致性.

(4) 在网络规模增大、数据平面转发设备数量增多的环境下,单控制器设备可能难以满足性能需求.

OpenFlow 最初的研究出发点是面向校园网或企业网的创新需求,但随着基于 OpenFlow 的 SDN 应用范围逐渐扩大,控制平面的可扩展性已经成为当前研究热点,下面分别介绍当前针对控制平面可扩展性的相关研究.

3.2.1 DIFANE

尽管可以通过提前在 OpenFlow 交换机中安装流表来减小控制器负载,但真实网络通常是按需安装流表[32],需要控制器进行实时响应,而这无疑将增加控制器的处理开销.

针对当前基于流的网络对控制器依赖过重的实际场景,DIFANE[33]结合了主动和被动两种安装流表的方式将流量保持在数据平面,从而减小控制器负载.DIFANE 首先在 OpenFlow 交换机中选出权威交换机(authority switch),每个权威交换机管理一定区域内的 OpenFlow 交换机.控制器主动将分区规则安装到所有的 OpenFlow 交换机上,并根据全局网络信息主动在权威交换机上安装权威规则.当普通交换机产生新的数据流时,它根据自身的分区规则直接和自己分区内的权威交换机进行通信.由于权威交换机已提前部署了权威规则,因而可以向普通交换机安装缓存规则,同时,直接将请求数据转发给目的地而无须再返回给源交换机,从而去掉了传统流请求建立过程中数据包经过控制器的往返时延,也减少了控制器需要实时处理的控制流,如图 6 所示.在每个交换机内部,由于存在多种不同的规则,因此通过优先级进行先后处理.缓存规则的优先级最高,因为它直接管理数据的转发;权威交换机中的权威规则优先级次之,它控制自己域内的转发规则;分区规则优先级最低,仅在前面两种规则都不存在时才将数据包转发给域内的权威交换机.

在 DIFANE 系统中,由于权威交换机能够管理普通交换机的流建立请求,因此,控制器仅需要管理整个网络的区域划分和权威交换机的流触发规则.DIFANE 的区域划分根据流空间(flow space)定义,而流空间则根据具体的流表项来划分,一般由 7 个字段组成(源和目的 IP 地址、MAC 地址、端口号,协议),形成七维流空间.以图 7 的二维流空间(F_1,F_2)为例,F_1 和 F_2 分别代表源 IP 地址和目的 IP 地址,范围为 0~15,即(F_1=F_2=[0,15]).那么,规则 R_2 表示所有源 IP 地址为 1、目的 IP 地址为 0~7 的数据包.在图 7 中,控制器将整个网络划分为 4 个区域.具体的划分规则将根据实际区域的规则类型和范围来决定.

DIFANE 中的分区规则和权威交换机的权威规则一般仅需在网络发生变化时进行处理,因此不需要频繁地更新,从而减轻了控制器负载.实际上,DIFANE 并未完全将逻辑控制功能分配给权威交换机.它巧妙地利用流表

项的优先级特点来区分不同的规则,从而将实时安装流表项的开销分担给了权威交换机.然而,这种实现方式需要权威交换机具备规则安装功能,而传统的 OpenFlow 交换机无法实现,因此降低了实际部署过程的通用性.

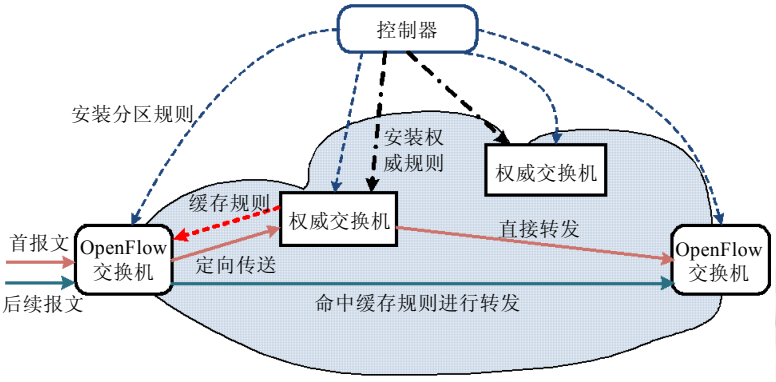


Fig.6 DIFANE flow management architecture

图 6 DIFANE 流管理架构

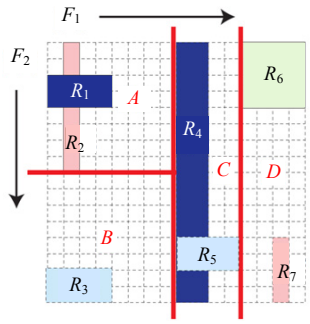


Fig.7 Rule and partition

图 7 规则和分区

3.2.2 DevoFlow

Andrew 等人考虑到当前的 OpenFlow 交换机流建立过程和统计信息收集过程将会消耗大量数据平面和控制平面之间的带宽,无法满足高性能网络的性能需求,提出了 DevoFlow[34]的设计方案.DevoFlow 采取了两种方式来减小 OpenFlow 交换机和控制器的信息交互:规则复制和局部操作.

规则复制方式在包含通配符的流表项中“操作”字段上增加了 CLONE 标志,如果该标志清零,则匹配该流表项的报文按正常情况处理;如果该标志被置位,则直接根据匹配报文建立精确匹配的微流,从而细化每一条微流的统计信息.由于带通配符的流表项一般由硬件 TCAM 实现,而精确匹配的流表项由软件实现,因而同时也减少了 TCAM 资源的消耗.在这种方式下,OpenFlow 交换机只需提前安装带有通配符的流表项,即可大量减少同控制器的报文交互,同时节省硬件资源开销.

局部操作方式包括多路径支持和快速重路由.多路径支持是指为 OpenFlow 交换机中可复制的通配符流表项提供多个可能的输出端口,DevoFlow 根据概率分布将报文输出到特定端口中,而不是采用传统的等价多路径路由方式(ECMP).快速重路由给 OpenFlow 交换机指定了 1 条到多条备用路径,从而在链路失效时立即转用备用路径,而不是转发给控制器.快速重路由可通过安装不同优先级的流表来实现,但是需要在链路失败时将高优先级的流表项删除或者进行覆盖.

针对统计信息收集过程的开销,DevoFlow 采用 3 种方法来提高统计信息收集效率:采样、触发和报告、近似统计.采样方法将统计信息按照一定的样本概率转发给控制器;触发和报告通过设置阈值,在统计信息满足阈值条件时将统计信息发给控制器;近似统计则只将流量最大的 k 条流的统计信息发给控制器.通常情况下,这 k 条流包含了 80%~99%的数据流.

DevoFlow 设计方案的初衷与 DIFANE 类似,都是为了减小数据平面和控制平面的数据交互,从而减小控制平面负载.DevoFlow 考虑了流建立请求和获取统计信息的双重开销,因而直接在本地减小请求报文的数量;DIFANE 则将流建立请求的开销转交给权威交换机.前者的侧重点是从源头减少开销,后者的侧重点是分担开销.然而,DevoFlow 目前在硬件上尚未实际部署,其功能需要通过修改 OpenFlow 交换机的流表结构和硬件架构来实现,实际上增加了数据平面的部分控制功能,同样降低了实际部署过程的通用性.

3.2.3 HyperFlow

通过减小控制器的处理开销,无法从根源上解决单点性能瓶颈问题.多控制器管控的分布式控制平面的设计思想是未来基于 OpenFlow 的 SDN 面向较大规模网络部署的必经之路.HyperFlow[35]通过部署多台控制器来管理 OpenFlow 交换机,在每台控制器能够同步全网络视图的同时,只需要管理特定区域中的 OpenFlow 交换机.

如图 8 所示,左图中整个 OpenFlow 网络仅有 1 台控制器,因此跨域的服务请求将产生额外的时延,同时对控制器性能造成影响;右图在每个域内都部署 1 台~多台控制器,控制器之间通过消息的发布-订阅模式来来传输网络事件.在 HyperFlow 中,每个控制器将订阅数据信道、控制信道和自身信道.域内的网络和应用程序事件将发布给数据信道,针对特定控制器的事件将分别发送给该控制器信道.另外,每台控制器必须将自身的网络状态信息定期发布给到控制信道,以利于控制器的发现和故障检测.实际上,HyperFlow 是基于为广域网设计的分布式文件系统 WheelFS 而设计的,网络事件在不同控制器之间以文件的更新形式来实现.

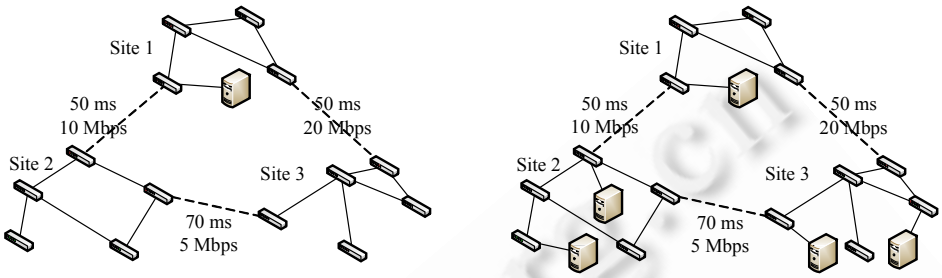


Fig.8 A multi-site OpenFlow network with single and multiple controllers

图 8 一个和多个控制器的多域 OpenFlow 网络

HyperFlow 通过 NOX 上的应用程序方式实现,因此实现过程简单,只需对 NOX 进行少量修改.从实现和测试的性能来看,在保证控制带宽和限定网络延迟的情况下,HyperFlow 能够处理的网络事件小于 1 000 次/秒,性能还不够高.另一方面,全网络视图的更新将与网络状态信息发布的周期时间和传输时延相关,这由 WheelFS 的实现机制决定.因此,这种实现方式适用于网络状态事件更新不频繁、对网络状态一致性要求不高的网络.对于网络状态事件频繁更新的网络,或者对于数据中心或较大规模网络,HyperFlow 有可能面临性能瓶颈.

3.2.4 Onix

针对控制平面在可扩展性、可靠性和通用性等方面的不足,Onix[31]提出了一整套面向大规模网络的分布式 SDN 部署方案.Onix 网络架构主要由物理网络基础设施、网络连接基础设施、Onix 和网络控制逻辑这 4 部分组成,如图 9 所示.物理网络基础设施允许 Onix 读写网络状态;网络连接基础设施提供 Onix 和物理网络基础设施之间的通信连接;Onix 采用分布式架构向上层控制逻辑提供网络状态的编程接口;网络控制逻辑则通过 Onix 提供的 API 来决策网络行为.网络信息库(network information base,简称 NIB)用于维护网络全局的状态,Onix 的设计关键就在于维护 NIB 的分发机制,从而保证整个网络状态信息的一致性.

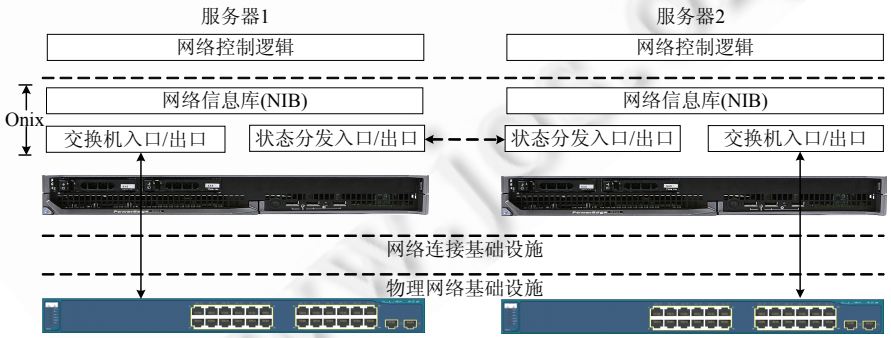


Fig.9 Onix network architecture

图 9 Onix 网络架构

Onix 主要通过 3 种策略来实现控制平面的可扩展性:

(1) 分区.和 HyperFlow 一样,随着网络规模的增大,数据转发平面必须由不同的控制器进行分区管理.Onix 通过不同的实例管理每个域,并根据 NIB 来配置数据平面.

(2) 聚合.整个 Onix 网络将形成一个分层的拓扑结构,每个 Onix 实例需要维护本分区的路由决策,分区间的路由决策则由 Onix 实例构成的集群共同决策.在网络规模增大、网络事件更新频繁的情况下,顶层的控制逻辑将无法完全匹配网络事件的更新速度.为此,Onix 实例需要知道本域内的拓扑情况,而将其余的 Onix 实例管理的网络看作是一个聚合的逻辑结点.

(3) 一致性和稳定性.根据不同的网络需求,Onix 提供两种方式来实现 NIB 的更新分发机制:带复制的事务性数据库模式和 DHT 模式.前者的主要目标是提供一种可靠的分发机制,适用于网络事件更新缓慢、对稳定性和一致性要求较高的网络;后者则通过通用 API、软状态触发器和坐标机制等 DHT 实现机理,构建快速响应的分布式拓扑结构,适用于网络事件更新频繁、对网络可用性要求较高的网络.

Onix 目前采用的 NIB 分发机制实际上都是比较成熟的分布式系统解决方案,因此也将承担这些方案自身的优缺点,如数据库模式自身存在严重的性能限制、多个 Onix 实例对 DHT 更新可能导致状态不一致等.因此,在 Onix 实际部署过程中需要综合考虑网络拓扑环境和性能需求,或者考虑采用其他更有优势的分布式系统.

Onix 已经作为许多组织机构构建商业应用的基础平台.从 Onix 的应用场景来看,每一个 Onix 实例能够管理多达 64 台 OpenFlow 交换机,由 5 个 Onix 实例组成的集群已经通过了测试.以每台 OpenFlow 交换机能够连接 48 台服务器来看,Onix 能够应用于主机数量达到数万量级的较大规模网络.

和前面相比,DIFANE 和 DevoFlow 是通过减小控制器负载或者优化 OpenFlow 交换机结构来实现控制平面的可扩展性;Onix 则为基于 OpenFlow 的 SDN 提出了一种更为通用的扩展方法,即通过多控制器分域进行管控.与 HyperFlow 类似,这也是未来 SDN 面向大规模网络部署可扩展性研究的主要发展方向.

3.2.5 Devolved controller

HyperFlow 和 Onix 并未针对控制器如何分区提出具体的分区算法.鉴于多路径区域划分问题在图论中是 NP 难问题,Devolved controller[36]针对多控制器管控区域划分问题提出了两种启发式算法:路径-分区算法(path-partition approach)和分区-路径算法(partition-path approach),如图 10 所示.

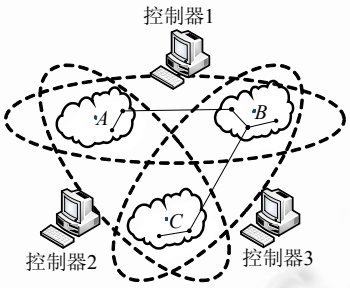


Fig.10 Control area partition of multi-controller

图 10 多控制器管控区域划分

路径-分区算法首先计算每对结点之间多路径分配到不同控制器上的开销,并选取其中开销最小的控制器来进行管理.针对每个控制器 i,开销 c_i 的计算如下式所示:

$$c_i=\alpha\nu_i(M)+\mu_i,$$

其中,μ_i 表示网络中已经被控制器 i 监控的链路数量,$\nu_i(M)$表示多路径 M 中还未被控制器监控的链路数量,参数 α表示权重因子.α趋近于 0,则不考虑多路径中已被控制器监控路径的重用优势;α趋近于无穷大,则不考虑控制器管控范围的平衡性.

分区-路径算法首先将所有的链路分配给每个控制器,在每对结点之间的多路径分配到不同控制器之前,每个控制器对已划分到自己域内的链路设置优先权重,然后通过上面的等式计算多路径分配到不同控制器的开

销,因此,多路径区域划分将优先考虑每个控制器域内最初划分的链路.

这两种启发式算法都降低了实际分区算法的复杂度.从分区结果来看,分区-路径算法使得每个控制器监控的链路更少,从而减小了控制器开销;但路径-分区算法不需要考虑每个控制器已经划分的链路,因此通常可以得到最短路径.然而,这两种算法都会导致覆盖范围重叠,即有些链路将会同时划分给多个控制器.从理论上讲,这将增加关键链路的健壮性,但实际决策过程仅需由 1 个控制器产生,因此需要结点或控制器维护额外的映射机制来避免决策冲突.

3.2.6 小 结

从上述解决方案来看,目前,基于 OpenFlow 的 SDN 可扩展性研究的主要思路有如下两点:

(1) 修改 OpenFlow 交换机的处理流程或硬件架构,或者给 OpenFlow 交换机增加部分控制功能,从而减少控制器和 OpenFlow 交换机之间的信息交互,分担控制器的处理开销,提高控制器的可扩展性能.实际上,优化控制器性能在一定程度上也能提高控制器的管控范围,如 NOX-MT[37]等通过提升控制软件性能的研究.这种思路属于纵向扩展,适用于校园网、企业网等规模不大的网络.

(2) 多控制器的分布式管控平面,通过分域管理网络,控制器之间实现基本的状态分发过程.这种思路属于横向扩展,是 SDN 可扩展性研究的主流方向,适用于数据中心、广域网等规模较大的网络.

表 2 总结了以上基于 OpenFlow 的 SDN 可扩展性解决方案的相关研究,主要从扩展方案、通用性、全局状态维护、实现复杂性、部署环境、规模和主要技术特点等方面进行比较.

Table 2 Comparison of scalability schemes of OpenFlow-based SDN

表 2 基于 OpenFlow 的 SDN 可扩展方案对比

名称	实现方式				部署环境	规模	主要技术特点
	扩展方案	通用性	全局状态维护	实现复杂性			
DIFANE	纵向	较低	No	中	企业网	较小	控制器对规则进行区域划分,通过权威交换机实现规则安装功能
DevoFlow	纵向	低	No	中	数据中心	中等	规则复制、快速重路由、多路径支持、采样、触发报告、近似统计
HyperFlow	横向	较高	Yes	中	企业网 数据中心	中等	通过分布式存储系统 WheelFS 维护全网络视图
Onix	横向	高	Yes	较高	广域网 数据中心	较大	提供带复本的事务性数据库和 DHT 两种模式分发网络信息库
Devolved controller	横向	中	N/A	较低	数据中心	中等	多控制器管控区域划分,提出两种启发式分区算法

3.3 SDN控制逻辑的一致性

基于 OpenFlow 的 SDN 技术虽然通过控制器集中操作,但本质上还是分布式和异步操作的.由于网络事件有可能发生在任何一台交换机或端主机上,控制器和交换机之间存在的时延将有可能影响到控制器接收事件的次序以及控制器规则在交换机上的安装次序,进而影响到控制逻辑的一致性.从本质上讲,这是由 SDN 控制逻辑安装不具备原子性造成的.如图 11 所示[38],由于时延的影响,报文在交换机 2 安装规则之前已经到达,这将导致交换机 2 重新将报文发送到控制器上,有可能造成部分报文在交换机 2 上经历的控制逻辑不一致.实际上,前面提到的 DIFANE 和 DevoFlow 等通过在 OpenFlow 交换机上增加自治智能的方法能够防止规则安装时延,但这违背了 SDN 控制转发分离的原则.

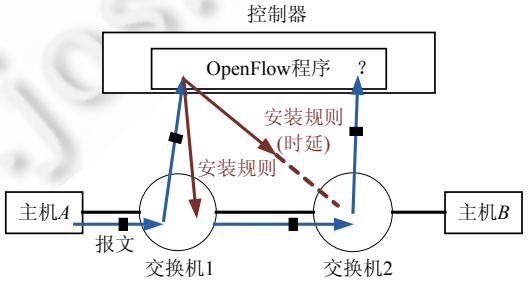
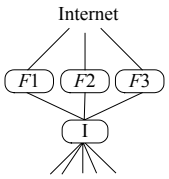


Fig.11 Inconsistency of control logic caused by delay

图 11 时延引发的控制逻辑不一致

另一方面,即使不受时延的影响,控制逻辑本身执行的顺序也有可能导致网络进入暂时的中间状态,从而影响到控制逻辑的一致性[39,40].以接入控制为例,图

12 表示某网络通过交换机 I 和 3 台过滤交换机 *F*1,*F*2 和 *F*3 连接到互联网上.配置 1 表示网络当前的网络配置:未知(unknown)和访客(guest)主机的流量转发到 *F*1 交换机,并过滤掉 SSH 流量,学生(student)和老师(faculty)主机的流量分别转发到 *F*2 和 *F*3 进行常规处理.在某时刻,网络负载发生变化,网络的逻辑控制需要切换到配置 2:未知主机流量通过 *F*1 交换机过滤 SSH 流量,访客主机通过 *F*2 交换机进行过滤,其余主机则全部转发给 *F*3 正常处理.如果不考虑每台交换机的配置顺序,则将导致控制逻辑的不一致.例如,首先增加 *F*2 的 SSH 过滤功能,则学生的正常流量会被暂时过滤掉;如果首先将访客的流量转发给 *F*2,那么访客的 SSH 流量将暂时无法过滤.因此,控制逻辑的一致性需要精心考虑每个更新步骤的执行顺序.



配置 1

	类型	操作
I	U,G	Forward *F*1
	S	Forward *F*2
	F	Forward *F*3
*F*1	SSH	Monitor
	*	Allow
*F*2	*	Allow
*F*3	*	Allow

配置 2

	类型	操作
I	U	Forward *F*1
	G	Forward *F*2
	S,F	Forward *F*3
*F*1	SSH	Monitor
	*	Allow
*F*2	SSH	Monitor
	*	Allow
*F*3	*	Allow

Fig.12 Inconsistency of control logic caused by sequence of rules

图 12 规则执行次序引发的控制逻辑不一致

下面就目前针对控制逻辑的一致性实现和验证等方面的相关研究进行介绍和分析.

3.3.1 控制器部署位置问题

在 SDN 中,控制平面和数据平面之间的时延将影响到控制逻辑能否有效部署到转发设备当中.尤其在流量激增、实时响应需求增加和广域网的情况下,响应时延越长,控制逻辑的一致性将更加难以保证.文献[41]引入平均时延和最坏情况时延两个指标来分析 Internet 2 上 OpenFlow 控制器的部署位置问题.

平均时延是指每个结点到控制器的最小时延的平均值.对于网络图 $G(V,E)$,边权重代表传播时延,$d(v,s)$表示结点 v 到 s 的最小路径,n 表示结点数量.对于控制器的部署位置集合,平均时延表示如下:

$$L_{avg}(S')=\frac{1}{n}\sum_{v\in V}\min_{(s\in S')}d(v,s).$$

最坏情况时延是指每个结点到控制器最小时延中的最大值,定义如下:

$$L_{wc}(S')=\max_{(v\in V')}\min_{(s\in S')}d(v,s).$$

文献[41]通过在 Internet 2 中进行大量的时延测量,并分别以最小化平均时延或最坏情况时延为标准,计算出不同数量控制器的部署位置.从时延分析结果可以看出,控制器的部署数量和位置都将极大地影响平均时延和最坏情况时延.根据成本收益比,即部署的 k 个控制器能否使平均时延或最坏情况时延减小到原来的 $1/k$,控制器数量在 3 或 4 时成本收益比趋于稳定,这也与目前 Internet 2 所推荐的部署 3 个控制器和 1 个备份控制器相吻合.针对更多不同的拓扑结构(8 个~200 个结点),一个控制器通常能够满足基本时延需求,但是随机部署位置通常很难达到最优时延分布.而从大部分拓扑结构来看,增加控制器的数量能使时延接近于按比例减少.

研究控制器的部署位置问题,主要针对 SDN 部署在广域网的情况.实际上,控制器的部署位置在控制器数量确定的情况下属于 NP 难问题,但就目前数据平面转发设备不多的现状而言,根据不同的时延指标计算出最优的部署位置是能够实现的.

3.3.2 每报文和每流一致性

针对网络控制逻辑更新的一致性,文献[39,40]提出了每报文(per-packet)一致性和每流(per-flow)一致性的概念.每报文一致性是指每个报文传输过程中要么执行旧的控制逻辑,要么执行新的控制逻辑,而不能混杂在一起执行.即使控制逻辑具备原子性,即规则安装在瞬时同步完成,也会造成传输中的报文经历不同的控制逻辑.如图 11 所示,假设规则安装是原子的,图中交换机 1 和交换机 2 之间链路上正在传输的报文也将经历不同的控

制逻辑.每流一致性指是指一系列相关的报文(指同一条流)在传输过程中要么执行旧的控制逻辑,要么执行新的控制逻辑,而不能混杂在一起执行.

文献[39,40]通过形式化模型证明了每报文一致性能够维护网络所有的控制逻辑属性,并提出两阶段更新方法(静默更新和单触更新),并证明其满足每报文一致性.静默更新首先部署新的网络配置,同时保留旧的网络配置,新旧配置采用不同的标记(如 VLAN,MPLS 等);静默更新完成后,单触更新则将所有的输入报文打上新配置的标记,随后所有报文将按照新配置执行.两阶段更新方法利用了 OpenFlow 网络的标记更新功能,采用平行配置的思路,相当于将报文和控制逻辑整体无缝迁移到新的控制平面,从而保证了每报文一致性.

每流一致性比每报文一致性的标准更加严格,通常适用于 TCP 流前后控制逻辑的一致性[42].通过对 OpenFlow 交换机中的新配置设置较低的优先级,可以等待旧配置在超时后自动删除.如果旧配置的流表项是粗粒度的,难以超时删除,则可以先将流表项细化,或者采用 DevoFlow[34]中规则复制的方法.由于每流一致性需要考虑整条流所有报文的一致性,因此通常无法保证控制逻辑得到实时更新.

3.3.3 NICE

SDN 控制逻辑主要通过控制器上运行的应用程序来执行.文献[38]设计了 NICE 方法来测试这些应用程序的正确性.NICE 结合了模型检查(model checking,简称 MC)和符号执行(symbolic execution,简称 SE)两种技术.首先通过模型检查方法搜索出整个系统(包括控制器程序、OpenFlow 交换机和端主机)的状态空间.考虑到控制器程序的事件处理器可能收到大量可能的输入报文,NICE 采用符号执行技术从事件处理器中得到不同的代码路径,并分离出报文等价类;针对每一种等价类,NICE 注入一个报文生成新的状态变迁.另外,NICE 还采用针对特定域的启发式算法来减少事件排序空间.通过提前定义应用程序的正确性属性,如无转发环路、无黑洞、直接路径等,NICE 最终生成包含应用程序缺陷的 trace 文件.NICE 的执行过程如图 13 所示.

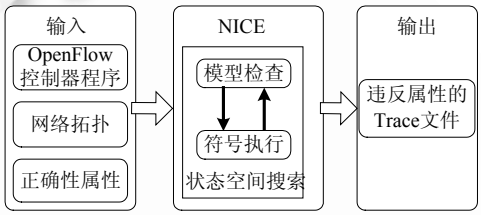


Fig.13 Execution course of NICE

图 13 NICE 执行过程

从测试对象来看,NICE 以 Python 语言开发,仅适用于 NOX 平台的 Python 程序.它主要通过状态和流程控制等协议验证方法测试控制器程序是否出现缺陷,侧重于控制器平台程序开发的正确性验证.

3.3.4 小 结

在基于 OpenFlow 的 SDN 技术中,控制平面的分离、网络状态分发机制和时延的影响,都对 SDN 控制逻辑的一致性需求提出了挑战.通过在 Internet 2 中进行的时延测量表明,控制器的部署数量和位置都极大地影响平均时延和最坏情况时延[41].在企业网和数据中心,OpenFlow 交换机一般都在本地,传播时延的影响可以忽略不计.但要注意到,减小时延只能尽量减小异常发生的可能性,无法在根源上满足控制逻辑的一致性.文献[39]提出的两阶段更新方法能够从根源上解决每报文一致性的问题,适用于网络整体控制逻辑切换的场景,然而在网络控制逻辑更新频繁,甚至需要实时配置的环境中,需要考虑两阶段分别执行的时延开销.NICE 则是从程序验证的角度检测控制器程序的正确性,实际上侧重于检测由程序带来的有可能违反控制逻辑一致性的代码片段.

3.4 运作模式和演进趋势

基于 OpenFlow 的 SDN 技术最初由斯坦福大学提出,并由 Nicira[16]和 Big Switch[17]等 SDN 公司推动发展.随着 Juniper,NEC,HP,IBM 等公司相继推出支持 OpenFlow 的硬件设备,Cisco 也投入大量资金研究部署 SDN.目前,ONF[13]成员基本涵盖了网络设备、运营商和互联网领域的大部分商业公司.这种以学术界为技术依托,并注重产业界的推广和应用的运作模式,是基于 OpenFlow 的 SDN 技术取得快速发展的根本原因.

基于 OpenFlow 的 SDN 技术颠覆了传统网络厂商的硬件设计模式,在实际部署、安全保障、分布式协调机制等方面还存在一系列问题,这很难被网络运营商所接受.SDN 在要求逻辑控制与数据转发分离的前提下,数据平面应当维持尽量简单的转发功能.然而,为了增加数据平面转发功能的向后兼容性,OpenFlow 标准不断扩充

转发平面匹配字段,这进一步增加了未来OpenFlow交换机的硬件设计成本,在一定程度上也增加了控制平面维护的复杂度,不利于OpenFlow的未来演进发展.在OpenFlow标准的不断演进中,是维持简单的硬件结构还是在OpenFlow 硬件中继续增加更多的功能,需要学术界和产业界的共同关注.文献[43]考虑到目前数据平面的设计复杂趋势,提出了一种新的概念:网络结构(network fabric).如图 14 所示,网络结构将网络服务等诸多复杂功能(如过滤、隔离或策略路由等功能)集中到边缘交换机,并通过边缘控制器来管理.网络核心即结构元素则通过结构控制器部署简单的路由机制(如 MPLS)来实现数据的转发功能.在这种架构下,边缘交换机能够通过主机软件实现,结构元素通过硬件成本更低的 MPLS 实现数据转发;同时,新协议的应用只需要在边缘交换机通过软件更新来实现,有利于网络的进一步升级和演化,符合目前网络边缘复杂、网络核心简单的设计原则.这种演进趋势意味着 OpenFlow 交换机需要划分为边缘和核心两个版本:边缘 OpenFlow 交换机更通用,核心 OpenFlow 交换机更简单.当然,这种方式需要考虑边缘到核心转发平面的映射机制,同时,边缘交换机的软件实现形式有可能产生性能瓶颈.

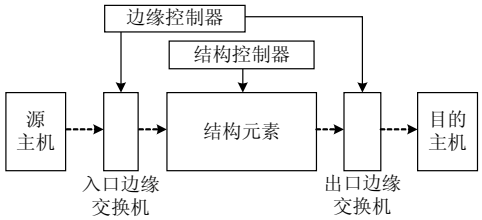


Fig.14 Network fabric

图 14 网络结构

4 基于 OpenFlow 的 SDN 应用

基于OpenFlow的SDN技术在解决当前存在的实际问题和开拓网络新应用等方面取得了不少成果,从部署的区域范围来看,它主要在校园网用于研究,在数据中心用于解决数据和控制密集型关键问题;从实现的功能来讲,它主要面向网络管理和安全控制.下面我们以不同的侧重点分类简述这 4 种应用.

4.1 面向校园网的部署

在校园网中部署 OpenFlow 网络,是 OpenFlow 设计之初应用较多的场所.它为学校的科研人员构建了一个可以部署网络新协议和新算法的创新平台,并实现了基本的网络管理和安全控制功能.目前,已经有包括斯坦福大学在内的多所高校部署了 OpenFlow 网络,并搭建了应用环境.

Plug-n-Serve[44]在斯坦福大学的计算机系大楼部署.它设计了一种新型的面向无结构网络的负载均衡算法,向系统中动态添加和移除计算资源,增加了请求的传输率,改变了每个请求的CPU和网络负载,最终实现了网络的负载均衡.OpenRoads[45]同时在斯坦福大学的计算机系和电机工程系大楼部署,利用 OpenFlow 和 SNMP 在异构无线网络(如 WiFi,WiMAX 等)中实现了网络虚拟划分和移动管理,简化了网络管控的方法.Resonance[46]是佐治亚理工大学的研究者在校园内部署的一整套 OpenFlow 网络动态接入控制系统,通过在高层部署安全策略和分布式监控推断系统,实现了更细粒度的分布式安全接入功能.

4.2 面向数据中心的部署

随着云计算模式和数据中心的发展,将基于 OpenFlow 的 SDN 应用于数据中心网络已经成为研究热点.这是由于数据中心的数据流量大,交换机层次管理结构复杂,服务器和虚拟机需要快速配置和数据迁移.如果不能在庞大的服务器机群中进行高效的寻址和数据传输,则很容易造成网络拥塞和性能瓶颈.将OpenFlow交换机部署到数据中心网络,可以实现高效寻址、优化传输路径、负载均衡等功能,从而进一步提高数据交换的效率,增加数据中心的可控性.

文献[47]首次提出将OpenFlow技术引入到数据中心网络,并采用NOX控制器实现了两种比较典型的数据中心网络 PortLand[48]和 VL2[49]的高效寻址和路由机制.Ripcord[50]同样实现了这两种数据中心的路由引擎原型系统,并支持网络动态管理,增加了网络健康度监控和自动报警功能.ElasticTree[51]设计了一个在数据中心部署的能量管理器,动态调节网络元素(链路和交换机)的活动情况,在保证数据中心的流量负载平衡的情况下,达到了节能的目的.Hedera[52]在PortLand的实验床上实现了一种可扩展的动态流调度系统,通过OpenFlow交换机获

取流的最新动态,采用全局最先匹配(global first fit)的贪婪算法和模拟退火(simulated annealing)算法实现了数据中心的负载平衡.Virtue[53]利用 OpenFlow 交换机控制网络流量,实现了数据中心内不同的虚拟机布置算法的比较,并通过真实的数据中心和模拟器两种方式进行了实验.另外,文献[34,36]也都是在数据中心部署 OpenFlow 网络的相关研究.

4.3 面向网络管理的应用

OpenFlow 网络的数据流由控制器做出转发决定,使得网络管理技术在 OpenFlow 网络中易于实现,尤其是流量管理、负载平衡、动态路由等功能,通过配置控制器提前部署转发策略,将实现更加直观的网络管控模式.

OpenPipes[54]实现了一个高速网络的原型系统.它可以通过硬件和软件两种方式来实现系统的功能模块,并通过 OpenFlow 技术实现功能模块在网络运行时的动态划分.文献[55]通过 OpenFlow 来控制移动用户和虚拟机之间的连接,并根据移动用户的位置进行重路由和移动管理,使得它们保持源 IP 不变并保证始终通过最短路径,改变了传统移动 IP 的路由策略.OpenTM[56]利用 OpenFlow 控制器中的路由信息,分析了从不同交换机获取流统计数据的网络负载问题,从而构建整个网络的流量矩阵.文献[57]在 NOX 上实现了多个应用程序,提供了收集数据和配置 OpenFlow 网络的 API,并开发出 Web 界面对 OpenFlow 网络进行管理.文献[58]首次将 OpenFlow 技术应用到无线 Mesh 网络中,并解决了移动管理和客户 IP 迁移的相关问题.文献[59]提出了一种基于 OpenFlow 技术,支持 QoS 的可扩展流媒体应用优化框架,为视频数据流提供了非最短路径的动态重路由功能.文献[60]将 OpenFlow 应用于家庭网络管理当中,实现了新型的流测量和管理接口.文献[61]在 OpenFlow 交换机和 NOX 上实现了 MPLS 流量工程和基于 MPLS 的 VPN 功能.

4.4 面向安全控制的应用

随着 OpenFlow 在网络管理方面的应用日益丰富,OpenFlow 的流管理功能很容易进行扩展,从而实现数据流的安全控制机制.实际上,在面向校园网的部署环境里,有很多应用都是针对安全管控的.

文献[62]提出了一套轻量级的 DDOS 攻击检测方法.它通过提取 OpenFlow 流统计信息中与 DDOS 攻击相关的六元组,采用人工神经网络方法 SOM(self organizing map)进行降维处理,从而识别 DDOS 攻击.文献[63]利用 OpenFlow 交换机能够统计流数据的功能,采用 trie 的数据结构设计了一种识别大的聚集流量的功能,可应用于网络的异常检测.文献[64]考虑到目前源地址验证标准(SAVI)的不足,通过 NOX 上开发的应用程序来获取全局网络视图,从而决策 SAVI 设备的验证规则.

5 总结与展望

基于 OpenFlow 的 SDN 技术是控制转发分离架构的延续和深化,在提出时就受到广泛关注,为未来互联网的发展提供了一种新的解决思路.目前的 SDN 主要由 OpenFlow 实现其设计思想,但在面向真实网络的部署过程中,性能优化、可扩展性、安全性、分布式控制等需求都需要重新考虑.尽管已经取得了长足的进步,但从目前的研究和应用来看,基于 OpenFlow 的 SDN 技术无论在学术界还是产业界都仍处于发展阶段,还有很多问题需要深入研究.下面对其未来研究重点和发展趋势进行探讨.

(1) OpenFlow 标准的推进和控制软件的开发

OpenFlow 标准取决于业界共识,其决定了未来的发展方向,将是各大网络设备厂商关注的焦点.应当以实际的功能需求和演进趋势为导向来决定流表结构设计需要支持哪些功能.对于核心部件控制器,其控制软件的开发将以现有版本为基础,性能优化、易于部署、扩展性强将是主要研究目标.随着主流网络设备厂商的加入,控制软件将会出现更多性能更好的版本.

(2) 基于 OpenFlow 实现 SDN

OpenFlow 并不是支撑 SDN 技术的唯一标准,但其相关规范已经得到普遍承认.基于 OpenFlow 实现 SDN,在网络中实现软硬件的分离以及底层硬件的虚拟化,打破了传统网络的封闭性,适应了目前降低网络复杂度、提高网络开放性和虚拟化的需求,为网络的发展和创新应用提供了一个良好的基础平台.目前来看,在 SDN 控

制软件上,通过开发不同的应用程序来实现对网络的管控,将成为实现可编程网络的关键步骤.同时,当网络配置从设备迁移到软件平台时,整个网络的逻辑管控功能将不再受限于专用硬件,而是通过一系列软件来部署和实现,使得独立可扩展软件将在控制软件上成为网络核心,增加了网络的可管可控性,这也是未来基于OpenFlow 实现 SDN 的发展趋势.

(3) 网络管理和安全控制

通过控制器对网络流量进行监控和调度,是实现未来互联网网络管理和安全控制功能的易行方法.当前,基于 OpenFlow 实现的网管和安全功能主要集中在接入控制、流量转发和负载均衡等方面,而在安全性机制设计、异常检测和恶意攻击防护等方面都可以进行更深入的研究.此外,在 OpenFlow 自身设计中,对安全性问题考虑还不够,应当特别关注控制器结点和 OpenFlow 交换机结构的安全防护问题.

(4) 数据中心网络部署

在云计算集群服务器大规模部署的今天,如何实现一个高效、稳定、可控的数据中心网络一直处在研究前沿.数据中心与因特网结构独立.它能够根据实际性能需求部署具备革命性的全新架构,这也是基于 OpenFlow 的 SDN 技术目前能够在数据中心得到推广的原因.另一方面,SDN 控制转发分离的技术特点满足了数据中心密集型服务器需要集中管控的需求,增加了数据中心实际配置和操作的灵活性.Google 在其数据中心全面采用基于 OpenFlow 的 SDN 技术,大幅度地提高其数据中心之间的链路利用率,起到了很好的示范作用.同时,针对数据中心动态路由、负载均衡和能量管理等方面的研究也有相关实例部署,这将是基于 OpenFlow 的 SDN 技术近几年的研究热点.

(5) 面向大规模网络的部署

目前来看,基于 OpenFlow 的 SDN 部署环境主要面向校园网、企业网和数据中心,还缺乏针对大规模网络部署的相关经验.真实网络面临的异构环境、性能需求、可扩展性、海量数据和域间路由等都可能成为制约其发展的因素.Onix[31]已经提出一套面向较大规模真实网络的部署方案,并提出新的实现技术,为 SDN 的大规模实际部署提供了技术指导.但真实网络中外部控制器是否能够真的扩展,能否为大型网络提供控制路径指引,仍然是亟待解决的问题,还需要深入研究和实验.

(6) 面向未来互联网研究的部署

目前,基于 OpenFlow 的未来互联网测试床平台已经在世界各国逐渐建立起来.Internet 2[14]已基本完成 100G 接入链路的 OpenFlow 网络建设,其部署范围包括全美国的 30 多个结点.另外还有 CONET[65],OFELIA[15]等.在面向未来互联网的实验平台下,通过基于 OpenFlow 的 SDN 控制转发分离架构,将有利于实现新型网络控制协议和相关的网络测量机制.在世界各国针对未来互联网设计蓬勃发展的推动下,未来互联网研究将在增加网络可控性的基础上逐渐展开,基于 OpenFlow 的 SDN 技术有可能发展成为面向未来互联网的新型设计标准.

References:

[1] Mckeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008,38(2):69−74. [doi: 10.1145/1355734.1355746]

[2] Elliott C. GENI: Opening up new classes of experiments in global networking. IEEE Internet Computing, 2010,14(1):39−42.

[3] Gavras A, Karila A, Fdida S, May M, Potts M. Future Internet research and experimentation: The FIRE initiative. ACM SIGCOMM Computer Communication Review, 2007,37(3):89−92. [doi: 10.1145/1273445.1273460]

[4] JGN2plus. 2012. http://www.jgn.nict.go.jp/english/index.html

[5] SOFIA. 2012. http://fi.ict.ac.cn/research/sofia_overview.htm

[6] Yang L, Dantu R, Anderson T, Gopal R. Forwarding and Control Element Separation (ForCES) Framework. RFC 3746, 2004. http://tools.ietf.org/html/rfc3746

[7] Greenberg A, Hjalmtysson G, Maltz DA, Myers A, Rexford J, Xie G, Yan H, Zhan J, Zhang H. A clean slate 4D approach to network control and management. ACM SIGCOMM Computer Communication Review, 2005,35(5):41−54. [doi: 10.1145/1096536.1096541]

[8] Caesar M, Caldwell D, Feamster N, Rexford J, Shaikh A, Merwe J. Design and implementation of a routing control platform. In: Proc. of the 2rd USENIX Symp. on Networked Systems Design and Implementation (NSDI). Boston: USENIX Association, 2005. 15−28.

[9] Casado M, Garfinkel T, Akella A, Freedman MJ, Boneh D, Mckeown N, Shenker S. SANE: A protection architecture for enterprise networks. In: Proc. of the 15th Conf. on USENIX Security Symp. Vancouver: USENIX Association, 2006. 137−151.

[10] Casado M, Freedman MJ, Pettit J, Luo J, Mckeown N, Shenker S. Ethane: Taking control of the enterprise. In: Proc. of the SIGCOMM 2007. Kyoto: ACM Press, 2007. 1−12. [doi: 10.1145/1282380.1282382]

[11] Casado M, Koponen T, Moon T, Shenker S. Rethinking packet forwarding hardware. In: Proc. of the 7th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets). Calgary: ACM Press, 2008. http://conferences.sigcomm.org/hotnets/2008/papers/1.pdf

[12] MIT technology review. 2009. http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/

[13] Open networking fundation. 2012. http://www.opennetworking.org

[14] Internet 2. 2012. http://www.internet2.edu

[15] OFELIA. 2012. http://www.fp7-ofelia.eu

[16] Nicira. 2012. http://www.nicira.com

[17] Big switch. 2012. http://www.bigswitch.com

[18] Open networking summit. 2012. http://opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf

[19] OpenFlow setup. 2011. http://www.openflow.org/wp/deploy-labsetup

[20] Pfaff B, Pettit J, Koponen T, Amidon K, Casado M, Shenker S. Extending networking into the virtualization layer. In: Proc. of the 7th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets). New York: ACM Press, 2009. http://conferences.sigcomm.org/hotnets/2009/papers/hotnets2009-final143.pdf

[21] Naous J, Erickson D, Covington G, Appenzeller G, McKeown N. Implementing an OpenFlow switch on the NetFPGA. In: Proc. of the 4th ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS). San Jose: ACM Press, 2008. 1−9. [doi: 10.1145/1477942.1477944]

[22] Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S. Nox: Towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 2008,38(3):105−110. [doi: 10.1145/1384609.1384625]

[23] Sherwood R, Gibb G, Yap KK, Appenzeller G, Casado M, McKeown N, Parulkar G. Can the production network be the testbed? In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation (OSDI). Vancouver: USENIX Association, 2010. http://dl.acm.org/citation.cfm?id=1924969

[24] NOX. 2012. http://noxrepo.org

[25] Cai Z, Dinu F, Zheng J, Cox AL, Eugene TS. The preliminary design and implementation of the maestro network control platform. Technical Report, Department of Computer Science, Rice University, 2008. http://www.cs.rice.edu/~eugeneng/papers/Maestro-TR.pdf

[26] Beacon. 2012. http://www.beaconcontroller.net

[27] Floodlight. 2012. http://floodlight.openflowhub.org

[28] SNAC. 2012. http://www.openflow.org/wp/snac

[29] Trema. 2012. http://trema.github.com/trema

[30] Nascimento M, Rothenberg C, Salvador M, Corrêa C, Lucena S, Magalhães M. Virtual routers as a service: The RouteFlow approach leveraging software-defined networks. In: Proc. of the 6th Int'l Conf. on Future Internet Technologies (CFI). Seoul: ACM Press, 2011. 34−37. [doi: 10.1145/2002396.2002405]

[31] Koponen T, Casado M, Gude N, Stribling J, Poutievski L, Zhu M, Ramanathan R, Iwata Y, Inoue H, Hama T, Shenker S. Onix: A distributed control platform for large-scale production networks. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation (OSDI). Vancouver: USENIX Association, 2010. http://dl.acm.org/citation.cfm?id=1924968

[32] Casado M, Freedman MJ, Pettit J, Luo J, Gude N, McKeown N, Shenker S. Rethinking enterprise network control. IEEE/ACM Trans. on Networking, 2009,17(4):1270−1283. [doi: 10.1109/TNET.2009.2026415]

[33] Yu M, Rexford J, Freedman MJ, Wang J. Scalable flow-based networking with DIFANE. In: Proc. of the SIGCOMM 2010. New Delhi: ACM Press, 2010. 351−362. [doi: 10.1145/1851182.1851224]

[34] Curtis AR, Mogul JC, Tourrilhes J, Yalagandula P, Sharma P, Banerjee S. DevoFlow: Scaling flow management for high-performance networks. In: Proc. of the SIGCOMM 2011. Toronto: ACM Press, 2011. 254−265. [doi: 10.1145/2018436.2018466]

[35] Tootoonchian A, Ganjali Y. HyperFlow: A distributed control plane for OpenFlow. In: Proc. of the 2010 Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN). San Jose: USENIX Association, 2010. http://dl.acm.org/citation.cfm?id=1863136

[36] Tam A, Xi K, Chao HJ. Use of devolved controllers in data center networks. In: Proc. of the IEEE INFOCOM 2011 Workshop on Cloud Computing. Shanghai: IEEE Press, 2011. 596−601. [doi: 10.1109/INFCOMW.2011.5928883]

[37] Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R. On controller performance in software-defined networks. In: Proc. of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE). San Jose: USENIX Association, 2012. http://dl.acm.org/citation.cfm?id=2228297

[38] Canini M, Venzano D, Peresini P, Kostic D, Rexford J. A NICE way to test OpenFlow applications. In: Proc. of the 9th USENIX Symp. on Networked Systems Design and Implementation (NSDI). San Jose: USENIX Association, 2012. http://dl.acm.org/citation.cfm?id=2228312

[39] Reitblatt M, Foster N, Rexford J, Schlesinger C, Walker D. Abstractions for network update. In: Proc. of the SIGCOMM 2012. Helsinki: ACM Press, 2012. 323−334. [doi: 10.1145/2342356.2342427]

[40] Reitblatt M, Foster N, Rexford J, Walker D. Consistent updates for software-defined networks: Change you can believe in! In: Proc. of the 10th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets). Cambridge: ACM Press, 2011. [doi: 10.1145/2070562.2070569]

[41] Heller B, Sherwood R, McKeown N. The controller placement problem. In: Proc. of the SIGCOMM 2012 Workshop on Hot Topics in Software Defined Networking (HotSDN). Helsinki: ACM Press, 2012. 7−12. [doi: 10.1145/2342441.2342444]

[42] Wang R, Butnariu D, Rexford J. OpenFlow-Based server load balancing gone wild. In: Proc. of the 1st USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE). Boston: USENIX Association, 2011. http://dl.acm.org/citation.cfm?id=1972438

[43] Casado M, Koponen T, Shenker S, Tootoonchian A. Fabric: A retrospective on evolving SDN. In: Proc. of the SIGCOMM 2012 Workshop on Hot Topics in Software Defined Networking (HotSDN). Helsinki: ACM Press, 2012. 85−90. [doi: 10.1145/2342441.2342459]

[44] Handigol N, Seetharaman S, Flajslik M, McKeown N, Johari R. Plug-*n*-Serve: Load-Balancing Web traffic using OpenFlow. In: Proc. of the SIGCOMM 2009 (Demo). Barcelona: ACM Press, 2009. http://conferences.sigcomm.org/sigcomm/2009/demos/sigcomm-pd-2009-final26.pdf

[45] Yap KK, Kobayashi M, Sherwood R, Handigol N, Huang TY, Chan M, McKeown N. OpenRoads: Empowering research in mobile networks. ACM SIGCOMM Computer Communication Review, 2010,40(1):125−126.

[46] Nayak A, Reimers A, Feamster N, Clark R. Resonance: Dynamic access control for enterprise networks. In: Proc. of the SIGCOMM 2009 Workshop on Research on Enterprise Networking (WREN). Barcelona: ACM Press, 2009. 11−18. [doi: 10.1145/1592681.1592684]

[47] Tavakoli A, Casado M, Koponen T, Shenker S. Applying NOX to the datacenter. In: Proc. of the 10th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets). New York City: ACM Press, 2009. http://conferences.sigcomm.org/hotnets/2009/papers/hotnets2009-final103.pdf

[48] Mysore RN, Pamboris A, Farrington N, Huang N, Miri P, Radhakrishnan S, Subramanya V, Vahdat A. PortLand: A scalable fault-tolerant layer 2 data center network fabric. In: Proc. of the SIGCOMM 2009. Barcelona: ACM Press, 2009. 39−50. [doi: 10.1145/1592568.1592575]

[49] Greenberg A, Jain N, Kandula S, Kim C, Lahiri P, Maltz DA, Patel P, Sengupta S. VL2: A scalable and flexible data center network. In: Proc. of the SIGCOMM 2009. Barcelona: ACM Press, 2009. 51−62. [doi: 10.1145/1592568.1592576]

[50] Heller B, Erickson D, McKeown N. Ripcord: A modular platform for data center networking. In: Proc. of the SIGCOMM 2010 (Demo). New Delhi: ACM Press, 2010. 457−458.

[51] Heller B, Seetharaman S, Mahadevan P, Yiakoumis Y, Sharma P, Banerjee S, McKeown N. ElasticTree: Saving energy in data center networks. In: Proc. of the 7th USENIX Symp. on Networked Systems Design and Implementation (NSDI). San Jose: USENIX Association, 2010. [doi: 10.1145/1851182.1851261]

[52] Al-Fares M, Radhakrishnan S, Raghavan B, Huang N, Vahdat A. Hedera: Dynamic flow scheduling for data center networks. In: Proc. of the 7th USENIX Symp. on Networked Systems Design and Implementation (NSDI). San Jose: USENIX Association, 2010. http://dl.acm.org/citation.cfm?id=1855730

[53] Erickson D, Heller B, Yang S, Chu J, Ellithorpe J, McKeown N, Parulkar G, Rosenblum M, Whyte S, Stuart S. Optimizing a virtualized data center. In: Proc. of the SIGCOMM 2011 (Demo). Toronto: ACM Press, 2011. 478−479. [doi: 10.1145/2018436.2018530]

[54] Gibb G, Underhill D, Covington A, Yabe T, McKeown N. OpenPipes: Prototyping high-speed networking systems. In: Proc. of the SIGCOMM 2009 (Demo). Barcelona: ACM Press, 2009. http://conferences.sigcomm.org/sigcomm/2009/demos/sigcomm-pd-2009-final50.pdf

[55] Erickson D, Gibb G, Heller B, Underhill D, Naous J, Appenzeller G, Parulkar G, McKeown N, Rosenblum M, Lam M, Kumar S, Alaria V, Monclus P, Bonomi F, Tourrilhes J, Yalagandula P, Banerjee S, Clark C, McGeer R. A demonstration of virtual machine mobility in an OpenFlow network. In: Proc. of the SIGCOMM 2008 (Demo). Seattle: ACM Press, 2008. http://conferences.sigcomm.org/sigcomm/2008/papers/p513-ericksonA.pdf

[56] Tootoonchian A, Ghobadi M, Ganjali Y. OpenTM: Traffic matrix estimator for OpenFlow networks. In: Proc. of the 11th Int'l Conf. on Passive and Active Measurement (PAM). Zurich: Springer-Verlag, 2010. 201−210. [doi: 10.1007/978-3-642-12334-4_21]

[57] Mattos D, Fernandes N, Costa V, Cardoso L, Campista M, Costa L, Duarte O. OMNI: OpenFlow MaNagement infrastructure. In: Proc. of Int'l Conf. on the Network of the Future (NOF). Paris: IEEE Press, 2011. 52−56. [doi: 10.1109/NOF.2011.6126682]

[58] Dely P, Kassler A, Bayer N. OpenFlow for wireless mesh networks. In: Proc. of the 20th Int'l Conf. on Computer Communications and Networks (ICCCN). Maui: IEEE Press, 2011. 1−6. [doi: 10.1109/ICCCN.2011.6006100]

[59] Egilmez HE, Gorkemli B, Tekalp AM, Civanlar S. Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing. In: Proc. of the 18th IEEE Int'l Conf. on Image Processing (ICIP). Brussels: IEEE Press, 2011. 2241−2244. [doi: 10.1109/ICIP.2011.6116083]

[60] Mortier R, Bedwell B, Glover K, Lodge T, Rodden T, Rotsos C, Moore A, Koliousis A, Sventek J. Supporting novel home network management interfaces with OpenFlow and NOX. In: Proc. of the SIGCOMM 2011 (Demo). Toronto: ACM Press, 2011. 464−465. [doi: 10.1145/2018436.2018523]

[61] Sharafat AR, Das S, Parulkar G, McKeown N. MPLS-TE and MPLS VPNs with OpenFlow. In: Proc. of the SIGCOMM 2011 (Demo). Toronto: ACM Press, 2011. 452−453. [doi: 10.1145/2018436.2018516]

[62] Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proc. of the 35th Conf. on Local Computer Networks (LCN). Denver: IEEE Press, 2010. 408−415. [doi: 10.1109/LCN.2010.5735752]

[63] Jose L, Yu M, Rexford J. Online measurement of large traffic aggregates on commodity switches. In: Proc. of the 1st USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE). Boston: USENIX Association, 2011. http://dl.acm.org/citation.cfm?id=1972439

[64] Yao G, Bi J, Xiao PY. Source address validation solution with OpenFlow/NOX architecture. In: Proc. of the 19th IEEE Int'l Conf. on Network Protocols (ICNP). Vancouver: IEEE Press, 2011. 7−12. [doi: 10.1109/ICNP.2011.6089085]

[65] Detti A, Blefari-Melazzi N, Salsano S, Pomposini M. CONET: A content centric inter-networking architecture. In: Proc. of the SIGCOMM 2011 Workshop on Information-Centric Networking (ICN). Toronto: ACM Press, 2011. 50−55. [doi: 10.1145/2018584.2018598]

左青云(1986－),男,湖北大冶人,博士生,主要研究领域为网络管理与测量,软件定义网络,未来互联网.
E-mail: zuoqy@163.com

邢长友(1982－),男,博士,讲师,CCF 会员,主要研究领域为网络与分布式计算,未来网络,网络流媒体.
E-mail: changyouxing@126.com

陈鸣(1956－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络测量,网络性能分析与建模,分布式系统,未来网络.
E-mail: mingchennj@163.com

张国敏(1979－),男,博士,讲师,主要研究领域为网络管理,分布式计算.
E-mail: zhang_gmwn@163.com

赵广松(1984－),男,博士生,主要研究领域为网络性能建模,时延容忍网络.
E-mail: guangsongzhao@126.com

蒋培成(1988－),男,硕士生,主要研究领域为 SDN 网络,分布式网络与计算.
E-mail: jiangpeicheng@gmail.com