doi:10.11959/j.issn.1000-436x.2017137

软件定义网络可扩展性研究综述

付永红[1,2,3]，毕军[1,2,3]，张克尧[1,2,3]，吴建平[1,2,3]
（1. 清华大学信息网络科学与网络空间研究院，北京 100084；
2. 清华大学计算机科学与技术系，北京 100084；3. 清华信息技术国家实验室，北京 100084）

摘 要：随着软件定义网络的发展，可扩展性成为软件定义网络面临的重要问题之一。主要研究由于 SDN 的自身特征导致应用于大规模网络时面临的可扩展性问题。首先，讨论了导致 SDN 可扩展问题的 3 个主要方面原因：控制平面和数据平面分离、逻辑集中的控制以及细粒度的流管控。其次，从性能可扩展、地理位置可扩展、管控可扩展 3 个方面介绍了当前 SDN 可扩展性问题的研究进展。再次，讨论了 SDN 可扩展性评价方面的进展。最后，展望了 SDN 可扩展性的未来研究方向。

关键词：软件定义网络；可扩展性；控制平面；数据平面；综述
中图分类号：TP302 **文献标识码**：A

Scalability of software defined network

FU Yong-hong[1,2,3], BI Jun[1,2,3], ZHANG Ke-yao[1,2,3], WU Jian-ping[1,2,3]
(1. Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China;
2. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China;
3. Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing 100084, China)

Abstract: With the development of software-defined networking (SDN), its scalability has become one of the most important issues of SDN. The features of SDN was studied which lead to its scalability problem when SDN was applied to large-scale network. The three main causes leading to scalability problem were discussed: control plane and data plane separation, logical centralized control and fine-grained flow control. Meanwhile, the studies which focus on the scalability of SDN from three aspects: scalability of performance, scalability of geographic and scalability of control was presented. Further, the studies on the performance evaluation of the scalability of SDN were introduced. Finally, the future work was discussed.

Key words: software defined networking, scalability, control plane, data plane, survey

1 引言

软件定义网络（SDN, software defined networking）提供了一种网络可编程方法，将网络的控制与管理逻辑从网络设备中抽离出来，并向用户提供网络可编程能力用于加速网络的创新[1]。SDN 具有如下 4 个主要特点。

1) 控制平面与转发平面分离

软件定义网络将用于网络决策的控制平面与进行转发的数据平面相分离。最初提出控制转发分离思想研究的是 Tempest[2,3]。Tempest 于 1998 年提出了一种数控分离的可编程网络环境，并将 ATM 网络中的交换机虚拟化后提供给多个控制器来组成虚拟网络。进一步，国际互联网工程任务组（IETF, Internet Engineering Task Force）的转发件与控制件分离（ForCES, forwarding and control element separation）工作组于 2004 年提出了一种转发件（FE, forwarding element）与控制件（CE, control element）分

收稿日期：2016-12-09；**修回日期**：2017-05-17
通信作者：毕军，junbi@tsinghua.edu.cn
基金项目：国家自然科学基金资助项目（No.61472213）
Foundation Item: The National Natural Science Foundation of China (No.61472213)

离的开放可编程路由器体系结构[4]。2005 年，路由控制平台（RCP）[5]提出了一种逻辑集中的控制平面，将控制与转发分离。RCP 用来收集外部的边界网络协议（BGP）路由信息，并利用逻辑集中的控制平面解决内部边界网关协议（iBGP）与内部网关协议（IGP）策略不一致问题。4D[6]提出了一种具有 4 个平面（决策平面、传播平面、发现平面和数据平面）的新型网络管控架构。2007 年，Casadd 等[7]提出了一种新型的网络体系结构 Ethane。在 Ethane 中，有 2 个功能实体：逻辑集中的具有全局网络拓扑的控制器以及简单的、哑的交换机。

2) 逻辑集中的控制

软件定义网络将交换机的控制平面从交换机中抽离出来组成逻辑集中的控制平面。控制平面通过掌握的全局网络视图对全网进行集中式的路由决策及全局管控，并将决策结果下发至数据平面的交换机由其进行高速转发。早期采用逻辑集中控制的研究主要包括 RCP[5]、4D[6]、Ethane[7]等。

3) 开放可编程

通过开放设备接口，用户可以直接将上层的策略下发到转发设备中。开放接口的思想最初始于 1995 年的 Open Signalling（OPENSIG）运动[8]。OPENSIG 运动的目标是使 ATM 网络、Internet 以及 mobile 网络更加开放、可编程。在 OPENSIG 运动中，很多研究如 Tempest[2]、Mobiware[9]等都提供了定义良好的可编程接口，为网络提供可编程能力。在 SDN 中，南向接口是 SDN 控制平面与转发设备之间交互的接口。通过开放南向接口，SDN 打破现有网络由厂商锁定的现状，将控制平面的策略下发到不同厂商的设备中，降低了网络管控的复杂度以及网络运营维护的成本，并为上层应用提供可编程能力。OpenFlow 协议[10]是 SDN 最早提出的标准化的南向接口协议。同时，其他协议如 NetConf[11]、OVSDB[12]等均可作为 SDN 的南向接口协议。在 SDN 中，通过北向接口 REST API[13]和可编程语言（如 Frenetic[14]、Maple[15]等）为网络提供可编程能力。

4) 细粒度流管控

软件定义网络可通过数据平面的多维数据协议（如 OpenFlow[10]、OVSDB[12]等）进行细粒度的管控，为用户提供细粒度的服务，如基于不同应用提供不同的 QoS 服务（FlowQoS[16]）和流量工程（MicroTE[17]）等。

2 软件定义网络可扩展性问题概述

可扩展性是大规模网络设计最重要的方面。Neuman[18]对分布式系统的可扩展性进行了定义：系统的可扩展性是指当系统的用户和资源增加时，不会使系统的性能显著下降或使管理复杂度显著增加。本文重点研究软件定义网络由于自身特征导致应用于大规模网络时所面临的可扩展性问题，并对问题的原因进行了分类和归纳。具体而言，主要包括有以下几个方面。

2.1 控制平面与数据平面分离带来的扩展性问题

在传统网络中，以路由器为代表的网络设备内部既包含数据平面，也包含控制平面，它们之间是紧耦合的。软件定义网络将控制平面与数据平面解耦合，并将控制平面从设备内部剥离出来，使数据平面的网络设备仅负责高速转发，而网络的控制与管理由逻辑集中的外部控制器负责。这种方式尽管具有灵活性等优点，但是带来了控制平面和数据平面之间额外的通信开销、传输延迟以及数据平面处理延迟。

2.1.1 通信开销

当前，OpenFlow 作为 SDN 最早提出的南向接口协议，已得到广泛应用，并得到了多数设备厂商的支持。因此，本文以基于 OpenFlow 的交换机与 SDN 控制器之间通信为例阐述 SDN 可扩展性问题。

首先，分析网络新流到达时给控制平面和数据平面带来的额外通信开销。当一条新流到达 OpenFlow 交换机时，交换机会查找其内部存储的流表是否存在匹配的条目。如果没有匹配的条目，OpenFlow 交换机会将该流的第一个分组整体封装入 Packet_In 消息中并发送至控制器，并将此流缓存在交换机的 Buffer 中等待控制器的回复。另一种处理方式为非缓存模式，即将所有未命中的分组放入不同的 Packet_In 消息中并发送给控制器。控制器收到 Packet_In 消息并解封装对分组做出决策后，将该分组送至原 OpenFlow 交换机的出端口，并将相应的转发流表规则下发给沿途所有的 OpenFlow 交换机。Curtis 等[19]分析了 SDN 为一条新流建立一条经过 N 台交换机的双向数据通道时具体的消息数量，其中包括 2 个初始请求分组、$2N$ 条流表安装消息、控制器将分组送至交换机出口的 2 条 Packet_Out 消息，共 $2N$+4 条消息。Benson 等[20]指出对于仅有 100 台边界交换机的数据中心网络，在最坏情

况下，控制器每秒可以遇到 1 000 万条新流。由此可见，数据平面与控制平面之间存在巨大的通信开销。SDN 通信开销问题及传输延迟问题的示例如图 1 所示。

其次，除了新流地处理过程之外，SDN 在实时更新全局的状态信息（如全局的网络视图、网络状态、网络检测）时，数据平面与控制平面之间也会产生额外的通信开销。

2.1.2 传输延迟

传输延迟可按照方向分为 2 类：上行传输延迟和下行传输延迟。上行传输延迟指从新流到达产生 Packet_In 消息到该消息传输到控制器的时间。下行传输延迟指从控制器完成处理计算好路径规则放入 Flow_Mod 消息到该消息到达数据平面的交换机的时间。由于控制平面和数据平面分离，数据平面的交换机将新流的第一个分组放入 Packet_In 请求发送给控制器，并将该流缓存在交换机的 Buffer 中，然后该交换机需要等待控制器的回复。若交换机距离控制器过远，则往返传输延迟时间过长。如设传输速率为光速 300 000 km/s，控制器距交换机 3 000 km，则往返时延为 20 ms。若网络突发流量非常大，传输延迟将导致交换机缓存溢出，进而所有收到的分组将通过 Packet_In 消息发送至控制平面，进一步增大了控制平面与数据平面之间的通信开销以及控制平面的负载压力。

2.1.3 数据平面处理延迟

在 SDN 中，数据平面的交换机对分组的转发依赖于控制平面的决策结果。这一过程相对于传统交换机内部的总线交互方式带来了更大的处理延迟。

当一个分组到达 OpenFlow 交换机之后，交换机会首先查询其硬件转发表，若转发表中存在与该分组相匹配的转发表规则，则该分组按照线速进行转发。若不存在相匹配的转发规则，则该分组的转发将经历以下 3 个步骤：1) ASIC 交换芯片通过 PCIe 总线将该分组发送给交换机的 CPU；2) 引发一个系统中断，ASIC SDK 获取这个分组并将其传送给交换机侧的 OpenFlow 代理；3) OpenFlow 代理处理分组，并将元数据以及分组的前 128 B 封装成 Packet_In 消息发送给控制器。控制器收到 Packet_In 消息后，为该分组计算转发规则，通过 Flow_Mod 消息将计算结果下发给沿途所有的 OpenFlow 交换机。当 OpenFlow 交换机收到从控制器下发的 Flow_Mod 消息后，需要通过以下 4 个步骤进行处理：4) 交换机的 CPU 上运行的 OpenFlow 代理解析消息内容；5) OpenFlow 代理在诸如三态内容寻址寄存器（TCAM）等硬件上通过调度增加或删除一条转发表规则；6) 根据需要下发规则的性质，交换芯片的 SDK 对转发表中已有的规则进行重排（如

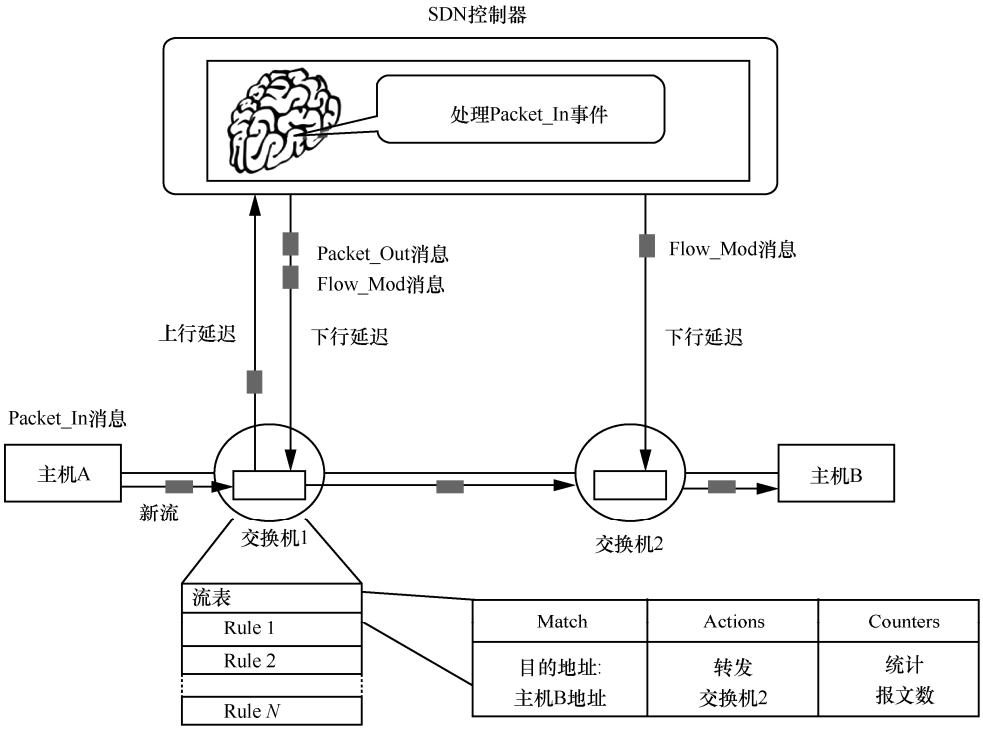


图 1　SDN 通信开销及传输延迟问题的示例

根据规则的优先级重排）；7) 在硬件转发表上插入（或移除）规则，如图 2 所示。

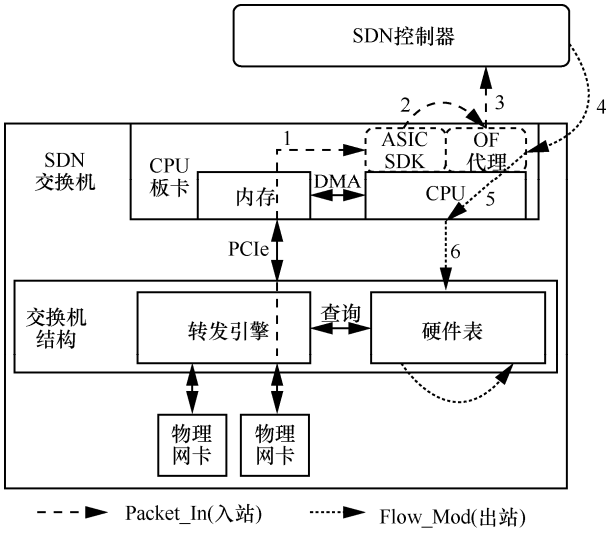


图 2 数据平面处理延迟

He 等[21]将数据平面内部的处理延迟分成入站延迟和出站延迟，其中，入站延迟指 SDN 交换机产生各种事件并向控制器发出分组的延迟；出站延迟指 SDN 交换机增加、修改、删除从控制器下转发规则的延迟。上面步骤 1)~步骤 3)产生的延迟总体构成了产生 Packet_In 消息的延迟，这个延迟称为入站延迟。He 等[21]在 Intel FM6000 的交换机上测试得到的平均每个分组的入站延迟为 8 ms。步骤 4)~步骤 7) 产生的延迟构成了交换机在处理流表安装过程的延迟，这个延迟称为出站延迟。He 等[21]在 Broadcom 956 846 KB 的交换机上测得插入一条流表和更新流表规则的平均延迟分别为 3 ms 和 30 ms。

2.2 逻辑集中的控制带来的扩展性问题

SDN 中逻辑集中的控制平面可以收集到全网的资源状态信息，并维护一个全局的网络视图，极大地简化了上层应用的开发，但是这也造成了中央控制器负载过大，限制了网络的规模。

2.2.1 控制器负载过大

当 SDN 将管理和控制的功能都交由一个中心控制器完成时，随着网络规模不断扩大、数据请求不断增多以至于中心控制器负载过重时，中心控制器就会出现明显的性能下降，控制平面的处理延迟增加，导致无法正常处理到达的请求。

Guan 等[22]使用了 GENI 平台构建了专门的测试网络，对中心控制器下 SDN 的可扩展性和可靠性进行了定量测试。测试结果显示当过量的新流进入 OpenFlow 交换机并产生大量的 Packet_In 消息到达控制器后，控制器 CPU 占用率处于高负载状态，同时数据平面由于流表规则没有及时下发导致高分组丢失率。

Tavakoli 等[23]对 NOX 早期的版本进行了测量，在保持流安装时间小于 10 ms 的前提下，每秒只能处理 30 KB 的新流初始化。而在大规模的网络中，在不妥协服务质量的前提下，所期望的控制器处理能力每秒应至少达到百万级别[20]。由此可见，中心控制器无法满足大规模网络的部署要求。

2.2.2 控制平面路由计算复杂度超线性增长

基于多维流转发的 SDN 网络，当网络规模扩展时，存在控制平面处理能力以线性增长，而控制平面的计算复杂度以超线性增长的问题[24]。假设一个 SDN 控制器管理一个有 M 台设备的网络，采用迪杰斯特拉算法计算路由，计算复杂度为 $O(M^2)$。当网络规模扩大 N 倍时，网络中有 N 台控制器和 NM 台网络设备，则控制平面的计算复杂度为 $O(N^2M^2)$。此时控制平面仅增加了 N 台控制器，控制平面的处理能力以 $O(N)$线性增长，而计算复杂度以 $O(N^2)$增长。图 3 给出了 SDN 控制平面路由计算复杂度超线性增长问题的示例。

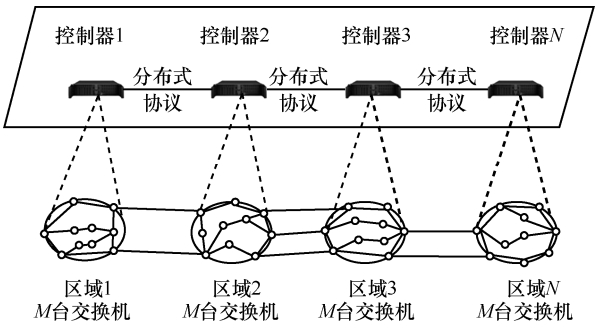


图 3 路由计算复杂度超线性增长示例

2.2.3 网络规模受限

现有互联网由 5 万多个自治系统（AS, autonomous system）互连构成，AS 之间是相互独立的，各 AS 通过运行分布式的 BGP 协议交换域间路由信息。如果将 SDN 应用到分布式的运行大量 AS 的广域网中，则其集中式的管控方式将限制网络规模增长。

2.3 细粒度的流管控带来的扩展性问题

当采用 SDN 进行细粒度的流管控时，会带来控制平面路由规则空间膨胀问题。

传统网络在进行路由时，域内路由协议（如 OSPF）或域间路由协议（如 BGP）均基于一维的目的 IP 地址或目的 IP 地址前缀进行转发。在传统路由器中，控制平面采用路由表存储路由信息，转发平面采用 TCAM 中的 FIB 表存储 IP 地址、下一跳以及出口信息。在 SDN 中，原有路由器中的路由表存放在控制器中，转发平面的 FIB 表中的规则存储于交换机的流表中。若 SDN 基于源 IP 地址、目的 IP 地址、端口号等进行多维转发，控制平面的路由表项以及数据平面的流表项均将以超线性膨胀。进一步，OpenFlow 协议提供十几个维度的转发，若根据这些维度进行转发将会加剧数据平面流表规则的膨胀。由于 TCAM 的存储空间有限，当存储的条目膨胀时，数据平面存储空间不足将导致大量流表规则无法安装在数据平面。当大量的新流到来时，没有匹配流表规则的分组将会被放入 Packet_In 消息并发送到控制平面，将给控制平面带来巨大的压力。

3　SDN 可扩展问题研究进展

本节从性能可扩展、规模可扩展、管控可扩展 3 个方面介绍了 SDN 可扩展问题研究进展。

3.1　性能可扩展

3.1.1　控制平面性能可扩展

1) 通过提高单控制器的性能提高控制平面的可扩展性

Cai 等[25]提出控制器设计的 3 个基本原则为：公平地处理不同交换机的请求、实现尽可能小的请求处理延迟、在多核处理器上能有效扩展。该研究基于上述 3 个方面提出了 Maestro 控制器，采用多核并行处理架构，同时使用轮询方式来进行负载均衡，从而兼顾了公平性、低延时和高吞吐率，其性能测试结果也超过了早期版本的 NOX[26]控制器。

Tootoonchian 等[27]在 NOX 基础上引入了多线程支持，提出了 NOX-MT，并采用了 I/O 批处理等优化技术，在 2 台 4 核处理器服务器上的测试表明，改进后控制器的基准性能相对于 NOX 提高了 33%。

Beacon[28]控制器引入了共享队列（shared queue）和运行到完成（run-to-completion）等设计方式进行 OpenFlow 消息的解析和处理，达到了应用完全多线程化，大幅提升了控制器的性能。

Rosemary[29]是一个健壮、安全且高性能的控制器，它采用了请求流水线（request pipelining）的方法提高了控制器的吞吐率和可靠性，同时对于一些延迟敏感的应用提供了信任执行（trusted execution）的模式。

Renart 等[30]在 SDN 控制器上利用 GPU 进行分组处理，控制器的代码同时运行在 CPU 和 GPU 上，其中，CPU 线程分成生产者线程和消费者线程 2 种；GPU 则负责分组的处理工作。

Ivashchenko 等[31]设计了一个内核级别的控制器，从而避免了由于系统调用导致用户空间和内核空间的切换开销，同时也避免了用户空间程序所使用的虚拟内存所需的额外内存转换以及隔离机制。利用内核级别的控制器可以达到更高的吞吐率，并且最低延时比已有的 OpenFlow 控制器缩短了$\frac{1}{2}$。

Zhao 等[32]在同样的条件下对主流控制器吞吐率和延时做了测试。对比了主流控制器 NOX[26]、Beacon[28]、OpenDayLight[33]等多个开源控制器的测试结果。

2) 控制器可采用主动预下发流表的方式提高 SDN 的可扩展性

SDN 控制器可以主动地预定义可能到达新流的流表转发规则并提前下发到交换机上，而不是被动地等待新流到达。通过预下发，Packet_In 事件就不会被触发，那些匹配了流表规则的新流均可以线速转发。

Fernandez[34]对比了主动下发和被动下发 2 种策略，测试结果表明在多种控制器上主动式策略均具有更好的性能表现。

3) 将部分控制功能下放给数据平面，减少控制平面负担

通过将部分功能下放到数据平面，可减少数据平面 Packet_In 到控制平面的分组数量。一方面分担了中央控制器的负载，另一方面也降低了控制器与交换机之间的通信开销。

Yu 等[35]提出了一种将所有数据分组都保持在数据平面的体系结构。该研究设置权威交换机，对控制器规则空间进行划分并分给几台权威交换机，使用多台权威交换机分担控制器的工作，由权威交换机负责向其区域内的交换机下发流表规则。通过将控制器的工作分发给这些额外的权威交换机，将流量尽量留在数据平面，避免频繁的数据平面和控制平面的通信开销，从而增强了整

体系统的可扩展性。

DevoFlow[36]通过将一部分功能下放到数据平面的交换机来减少控制平面与数据平面的通信开销，进而提高系统的可扩展性。DevoFlow 将小流的控制功能下放给交换机，并在交换机中增加了有效的流信息统计机制去识别那些重要的大流，控制器则只对重要的大流进行管理。这种方式一方面减少了一部分流统计信息的传输开销，另一方面也减少了请求控制平面进行流安装的开销。

3.1.2 数据平面性能可扩展

首先，一些研究通过向交换机中增加性能更强大的 CPU 提高数据平面的可扩展性。SDCs 则在转发设备中引入更强大的通用 CPU，基于软件方式实现了数据聚合和压缩功能，从而减轻控制平面的负载[37]。

其次，一些新的 SDN 交换机设计则将其他硬件与 TCAM 结合起来使用，如 SRAM、DRAM、GPU、FPGA、网络处理器等。

Ferkouss 等[38]运用 SRAM 与 TCAM 协同工作，共同存储查找表，提出递归流分类方法改善 OpenFlow 1.1 交换机的分类过程，实现对硬件资源更好地利用。

Memon 等[39]提出 FlashFlow 交换机，使用 GPU 加速分组处理，对于包含多达 10^6 条精确匹配条目和 1 000 条通配条目流表的交换机可以达到 20 Gbit/s 的吞吐能力。

Luo 等[40]使用基于网络处理器的加速卡改变了网络系统中分组的处理方式，将分组处理从主机 CPU 上转移给网络接口卡，从而提高了 OpenFlow 的交换机性能，并将分组的延时降低了 20%。

Stephens[41]为未来大型数据中心网络设计了一种全新的可扩展 SDN 体系结构——雨人，通过使用 L2 的表运行“例外路由算法”，其网络规模以近乎最优的性能扩展到数万级别的主机。

Li 等[42]则提出了一种全新的并行查找模型——拆分路由查找模型。其基本思想是将所有的前缀拆分成更小的部分，这样就会因为前缀的相似性产生冗余，再经过消除冗余压缩片内结构，这样使查找过程可以并行在 2 个更小的前缀完成，从而大大提高了性能。而且，减少 TCAM 中流表记录的数量也提高了数据平面可扩展性。

Braun 等[43]利用了 Espresso 启发式方法对域间路由表进行了压缩，其主要思想是对原有的基于前缀匹配的转发信息表（FIB）在逻辑上最小化，然后将结果表达为通配符的流表。最终在压缩时间限制到 1~2 s 的情况下，将当前 FIB 的规模减少了 17%，最多节省了 40 000 余条目。

Agarwal 等[44]提出了 shadow MAC，使用了一种标签交换的结构，利用虚拟的 MAC 地址（类似 MPLS 的不透明值）作为标签，易于在商用硬件上实现，而且这种标签结构不需要使用昂贵的 TCAM 资源，只需要使用 L2 转发表。这种标签交换还可以解决诸如网络一致性更新、可扩展的细粒度路由、快速重路由、多路径等一系列 SDN 问题。

3.2 规模可扩展

3.2.1 多控制器体系结构

现有研究采用 3 种控制平面体系结构提高控制平面的可扩展性：扁平式体系结构、层次式体系结构、混合式体系结构。

1) 扁平式体系结构

扁平式体系结构具有以下特点：① 控制平面所有控制器位于同一层次地位相等，均可以做全局决策；② 所有控制器共享全局网络信息（如视图信息、设备信息）。采用扁平式体系结构的研究包括 HyperFlow[45]、Onix[46]、ONOS[47]以及 Yanc[48]等。

HyperFlow[45]是一种基于 OpenFlow 的事件驱动的逻辑集中式物理分布式的控制平面。在 HyperFlow 中，每个区域内都部署一到多台控制器，控制器之间通过消息的发布—订阅模式传输网络事件。每个控制器订阅数据信道、控制信道和自身信道。区域内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发送给对应的控制器信道。另外，每台控制器必须将自身的网络状态信息定期发布到控制信道，便于控制器的发现和故障检测。控制器之间信息的同步是通过分布式文件系统 WheelFS 完成的，网络事件在不同控制器之间以文件的更新形式来实现。

Onix[46]提供了一个分布式的可操作的控制平面，采用基于内存的 DHT 维护网络信息库中的全局网络视图并为上层应用提供基础分布式的状态原语以及通用的 API 为网络提供可编程能力。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 4 部分构成。其中，Onix 通过物理网络基础设施读写网络状态信息，通过网络连接基础设施与物理网络基础设施之间进行通信连接；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(NIB)用于维护网络全局的状态。

ONOS[47]是一种开放的网络操作系统，用于提供大规模网络下高可用的分布式控制平面。ONOS的一个主要特点是其分布式架构的横向扩展能力和容错能力。ONOS运行在多个服务器上，其中，每个服务器管理一个 OpenFlow 交换机子集。一个ONOS实例负责在交换机之间传递状态变化以及全局网络视图。当数据平面的容量增长或对控制平面的需求增加时，可以在ONOS集群中增加新的实例来分担控制平面的负载。同时，ONOS添加了一个事件响应框架，改进了数据存储和数据模型，并增加了一个缓存层用以提高事件的响应延迟。

Yanc[48]将网络的配置及状态作为一个文件系统，使用户和系统应用可以通过标准的文件 I/O 来使用任何语言编写应用。在 Yanc 中，网络应用是独立的进程，通过建立在用户空间的虚拟文件系统层，不同的应用通过网络分片可以映射到不同的网络视图。

2) 层次式体系结构

层次式体系结构具有以下特点：① 控制平面的控制器为多个层次，不同层次的控制器掌握的网络信息（如网络视图、网络设备等）不同；② 底层控制器仅具有底层物理网络视图信息，负责管理本地设备与本地流量的转发。顶层控制器为集中式单控制器体系结构掌握全局网络视图，负责跨区域流量的管控。采用这种体系结构的研究包括Kandoo[49]、Logical xBar[50]以及ElastiCon[51]等。

Kandoo[49]是一种层次式的可扩展控制平面体系结构。Kandoo采用2层控制器：底层的本地控制器是一组独立的控制器，与数据平面的交换机通信，掌握本地视图运行本地应用，负责本地流量的转发；顶层根控制器与本地控制器交互网络信息，具有全局视图，运行全局应用进行跨区域的流转发。

Logical xBar[50]是一种递归的逻辑块用于构建层次式的 SDN 体系结构。它提供了一种网络抽象能力，通过网络抽象，屏蔽底层网络内部细节，将网络抽象的拓扑提供给上层控制器来提高网络整体的可扩展性。

由于控制器与交换机的连接采用静态配置的方式，这导致控制平面无法根据不同区域的流量情况调整控制器与交换机之间的连接。为此文献[51]提出了一种弹性的分布式控制器体系结构可根据流量情况动态调整控制器资源池。首先通过监测所有控制器的负载情况，周期性地调整控制器与交换机之间的映射均衡所有控制器的负载。其次，当控制器负载超过最大阈值时，添加新的控制器到资源池并进行现有交换机的迁移。当控制器负载低于最小阈值时，减小资源池中控制器的数量并调整交换机与控制器之间的映射。最后，文献[51]还设计了一个迁移协议来进行流量的无缝迁移。

3) 混合式体系结构

混合式体系结构具有以下特点。① 控制平面由多个层次组成。该结构顶层由多个分布式通信的多控制器组成，与层次式体系结构的集中式的顶层不同。控制器之间通过分布式通信协议分发各自掌握的网络信息，进而共享全局网络信息。顶层控制器负责跨区域流量的计算与转发。② 底层控制器仅具有底层物理网络提供的网络信息，负责本地的流量计算与转发。

Orion[24]提出了一种混合层次式体系结构用于大规模网络。在该体系结构中，底层控制器掌握本地视图处理本地应用，并将自己的边界交换机组成的抽象视图发给上层控制器。为了降低地理范围带来的延迟，Orion 顶层采用多个控制器并通过分布式协议分发全局抽象视图。通过抽象网络视图，Orion可降低SDN控制平面多维路由计算复杂度以超线性增长的问题。同时，它设计了一种抽象的层次式路由计算方法用于解决层次式抽象体系结构带来的路径延展问题。

3.2.2 东西向通信接口

由于单台 SDN 控制器的处理能力有限，当采用多控制器协作方式扩展控制平面的处理能力时，需要提供东西向通信接口来传递网络拓扑及网络设备等信息。

在扁平式多控制器体系结构地设计中，需要使用东西向接口传递消息，同步网络信息。但这类研究主要侧重于体系结构的设计，通常采用开源的分布式数据库作为东西向接口分发信息。例如，Onix[46]采用DHT[52]方式存储数据，并提供灵活的原语分发允许应用程序的设计者无需重新设计分发机制[53]。ONOS[47]采用 Cassandra[54]提供的键—值存储方式分发信息。

Yu 等[55]提出 Zebra，一种用于不同 SDN 域之

间通信的框架。Zebra 有 2 个模块：异构控制器管理模块（HCM）和域关系管理模块（DRM）。HCM 从一组互相不通信的控制器收集网络信息，并生成域内的视图。DRM 从其他域收集网络信息生成全局网络视图。

Benamrane 等[56]提出并实现了一种用于分布式控制平面的通信接口 CIDC，用于在分布式的多控制器之间交换同步信息、通知信息以及服务信息。同时，该研究实现了防火墙、负载均衡器等应用来提高分布式体系结构整体的安全性和服务质量。

SDNi[57]提出一种用于多管控域的东西向接口协议，支持多个域之间的路由信息交换，并支持跨域应用的路径需求、QoS、SLA 等需求。

WE-Bridge[58]是一种在 SDN 多管控域之间的东西向信息传递机制，定义了跨 SDN 域之间传递的信息。

3.2.3 控制器放置

Heller[59]等首次提出了 SDN 中的控制器放置问题：1) 若给定一个拓扑，需要多少控制器；2) 这些控制器放在网络中的什么位置。为了回答这 2 个问题，该研究将延迟作为度量标准对控制器放置问题建模，并对 Internet2 等公开真实拓扑进行了分析，给出了控制器数量及放置位置。

Yao 等 [60]提出除了将传输延迟作为度量值，控制器的负载对于控制器的放置位置也是一个关键因素。CCPP[60]对该问题进行了定义，并针对该 NP-hard 问题给出了一个整数规划模型进行求解。

Hu 等[61]提出通过控制器放置最大化网络的可靠性。该研究定义了提高网络可靠性的度量标准，并设计了控制器放置的算法。同时，为了最大化系统的可靠性，Hu 等[61]基于 OS3E 以及 Rocketfuel 的拓扑评价，对部署控制器的数量进行研究，并对可靠性和延迟之间的进行权衡评价。

Muller 等[62]提出了一种控制器放置策略 Survivor。该策略不同于与只考虑设备到控制器之间单路径连接的情况，而是考虑了多路径连接情况来增强连接性。同时，通过增加容量发现机制主动避免控制器过载。进一步，通过备份控制器列表和 2 种启发式算法来提高故障网络的收敛速度。

Rath 等[63]提出一种非零和博弈控制器最优放置机制，支持在每个 SDN 控制器中可以分布式使用。该机制可以确保控制器的最佳数量及其与 SDN 交换机之间的映射，从而节省成本并提高 QoS。

Lange 等[64]认为控制器放置是一种多种度量混合考虑的过程，从延迟、容错到负载均衡，提出了一种基于帕累托最优的控制器放置架构，可以根据不同的度量值为用户提供帕累托最优放置位置。

3.3 管控可扩展

在 SDN 中，对应于传统网络的自治系统定义 SDN 管控域的概念：SDN 中的管控域指的是由一个管理者进行管控的网络。SDN 管控域既可以仅包含一个 AS，也可以是在多个 AS 上的覆盖网（overlay）。

3.3.1 单管控域单自治域

单管控域单自治域指在互联网中一个自治系统由同一个管理者对所有 IP 网络和路由器进行管理，对互联网执行共同的路由策略。

在多控制器体系结构中介绍的 HyperFlow[45]、Onix[46]、ONOS[47]、Yanc[48]、Kandoo[49]、Logical xBar[50]、ElastiCon[51]、Orion[24]等工作，均可用于 SDN 单管控域单自治域，对同一个管控域内的网络设备进行管理，统一进行路由决策，并根据统一的转发决策对管控域内部的交换机下发流表，进行流量转发。

3.3.2 单管控域多自治域

单管控域多自治域指互联网中的多个自治系统由一个管理者对所有 IP 网络和路由器进行管理，对互联网执行共同的路由策略。有一些研究关注软件定义的广域网（WAN），由一个管理者管理以覆盖网形式跨越多个自治域的网络。

Google 提出了私有 WAN 网络 B4[65]。B4 具有以下特点：部署在站点上有大量带宽需求；平均带宽利用率最大化的弹性需求；通过对边缘服务器和网络的全面控制权，在边缘进行流速限制和测量。这些特性支撑了 B4 这种基于 OpenFlow 的软件定义的体系结构，使其可以简单地控制采用商业芯片的交换机。通过实际部署，B4 集中式的流量工程服务使链路利用率提高接近 100%。

SWAN[66]通过集中式的控制使 WAN 数据中心网络承载更多的流量。SWAN 在提高了利用率的同时可以满足策略目标，如优先级服务以及在同优先级服务之间的公平性。

DISCO[67]提出了一个分布式的控制平面来处理基于 WAN 网络的覆盖网络的异构性。DISCO 中各 SDN 域均有各自的控制器，控制器间具有轻量

级和易管理的控制器间通道。该通道使代理可以共享全局网络信息，进而支持端到端的网络服务。进一步，DISCO 可以动态适应异构网络拓扑，同时又可以弹性地容错和预防攻击。

Tempus[68]提出了在数据中心间的一种在线时间规划机制。该机制可对长期连续传输进行分组，在时间和空间上进行调度的调优。该机制可以为高优先级的流量提供零延迟和零流失服务。同时，在未超过网络容量长期运行的流量请求可以在截止时间之前完成。进一步，Tempus 还可以为未来的网络需求预留出一定的资源。

由于数据中心之间的广域网承载了一部分重要的对截止时间有要求的流量，DNA[69]通过一种基于截止时间的网络抽象来作为接口提供给用户，允许用户在请求中显式地指明其将要传输的数据量以及截止时间，进而灵活地分配资源，解决这个问题。

Jin 等[70]提出了一个新型流量管理系统 Owan，通过联合控制光层和网络层用来优化广域网上块数据的流量传输。由于光设备具有动态调整连接的特性，Owan 可以动态地改变网络层拓扑，进而根据流量的需求联合调优。进一步，该研究提出了一个高效的算法，根据流量的需求动态联合优化光电路设置、路由和速率分配。Owan 采用商业光交电硬件搭建了原型系统，实验结果显示 Owan 的平均传输速度比已有研究提高了 4.45 倍。

3.3.3　多管控域多自治域

多管控域多自治域指在互联网中多个自治系统由多个管理者对各自的 IP 网络和路由器进行分布式管理，各管控域之间通过协作交换路由信息并进行数据转发。

SDNi[57]提出一种用于多管控域的东西向接口协议，支持多个域之间的路由信息交换，并支持跨域应用的路径需求、QoS、SLA 等需求。

WE-Bridge[58]提出了一种在 SDN 多管控域之间的对等机制，并在 CERNET、Internet2、CSTNET 以及 SURFnet 之间进行了跨国测试。

SDX[71]提出了一种可扩展的软件定义的互联网交换节点 IXP，通过软件定义的 IXP 将 SDN 的特性引入现有基于 BGP 的域间路由。SDX 将基于特定应用的对等机制引入域间网络，并提供可编程抽象来支持域间应用。

由于 SDX[71]的设计不具备大规模可扩展性，iSDX 提出了一个大规模可扩展的用于工业界的 SDX 体系结构[72]。通过并行计算策略以及将分组转发地址编址到 MAC 域，将策略的编译时间以及转发表容量降低了 2 个数量级。

SDI 提出了一个分布式的多域 SDN 机制来支持细粒度的域间路由机制。SDI 在多个对等网之间宣告 IP 五元组信息，并且通过减少冗余流表的方式提高系统的可扩展性[73]。

SDN-IP[74]提出了一种增量部署 SDN 的方法，并使 SDN 可以和 IP 网络无缝连接。SDN-IP 通过运行在控制器 NOS 上的 BGP 路由模块同步 BGP 路由更新信息，并将它们存储在本地路由信息表（RIB）中。然后，利用主动流安装模块为通过 BGP 学到的路由信息计算域间流量在本地的路径。

相关文献分类总结如表 1 所示。

4　软件定义网络可扩展性评价

本节从性能可扩展评价、规模可扩展评价 2 个方面介绍了 SDN 可扩展性评价的研究进展，同时对相关文献进行了分类总结，如表 2 所示。

表 1　　**SDN 可扩展问题研究进展**

扩展性	相关研究	相关方法
性能可扩展	控制平面	Maestro[25]、NOX-MT[27]、Beacon[28]、Rosemary[29]、Renart[30]、Ivashchenko[31]、Zhao[32]、Fernandez[34]
	数据平面	SDCs[37]、Ferkouss[38]、Memon[39]、Luo[40]、Stephens[41]、Li[42]、Braun[43]、Agarwal[44]
规模可扩展	多控制器体系结构	HyperFlow[45]、Onix[46]、ONOS[47]、Yanc[48]、Kandoo[49]、Logical xBar[50]、ElastiCon[51]、Orion[24]
	东西向通信接口	Yu[55]、Benamrane[56]、SDNi[57]、WE-Bridge[58]
管控可扩展	控制器放置	Heller[59]、CCPP[60]、Hu[61]、Survivor[62]、Rath[63]、POCO[64]
	单管控域单自治域	HyperFlow[45]、Onix[46]、ONOS[47]、Yanc[48]、Kandoo[49]、Logical xBar[50]、ElastiCon[51]、Orion[24]
	单管控域多自治域	B4[65]、SWAN[66]、DISCO[67]、Tempus[68]、DNA[69]、Owan[70]
	多管控域多自治域	SDNi[57]、WE-Bridge[58]、SDX[71]、iSDX[72]、SDI[73]、SDN-IP[74]

表 2 SDN 可扩展性评价相关方法

扩展性	评价	相关方法
性能可扩展	定性评价	Shalimov[75]、Zhao[32]、Shah[76]、Rotsos[77]、Bianco[78]
	定量评价	Jarschel[79, 80]、Azodolmolky[81]、Fu[82]、Yao[83]、Osgouei[84]、Xiong[85]
规模可扩展	多控制器体系结构	Hu[86]、Karakus[87]

4.1 性能可扩展评价

4.1.1 定性评价

一些研究通过测试对 SDN 控制器及 OpenFlow 交换机的扩展性进行定性评价。

Shalimov 等[75]对多种 SDN 开源控制器（NOX、POX、Beacon、Floodlight、MuL、Maestro、Ryu）的各项效率指标（性能、可扩展性、可靠性、安全性）进行了综合分析。

Zhao 等[32]在同样的条件下对多款主流控制器的最新版本（POX、Ryu、Nox、Floodlight、Beacon、OpenDaylight、ONOS）的吞吐率和延时进行了测试。

Shah 等[76]提出了一种体系结构，验证 SDN 控制器的可扩展性。基于 3 种不同的度量指标（线程可扩展性、交换机可扩展以及延迟）验证了 4 种开源控制器的性能。进一步，根据测试结果，归纳了设计可扩展 SDN 控制器的 3 条准则：1) 用于高吞吐率的控制器设计应该使用静态交换机分区以及分组缓存；2) 延迟敏感的控制器设计应该采用负载可调节的分组缓存以及任务缓存来减小每个分组的延迟；3) 为了进一步降低延时，控制器应该单独发送每个控制信息。

Rotsos 等[77]提出了一种用于评估 OpenFlow 交换机的开放框架——OFLOPS，开发者可以利用 OFLOPS 模拟具体的场景并了解交换机实现对于性能的影响。利用 OFLOPS 测试了 OpenFlow 的软件实现 OpenvSwitch 以及若干硬件平台的 OpenFlow 实现，结果表明不同实现在流更新速率和流量检测能力上有很大差异。

Bianco 等[78]分析了 OpenFlow 在 Linux 主机上的实现，比较了 OpenFlow 交换、二层交换以及三层 IP 路由的性能，并分析了在不同的流量模式下，轻负载和重负载情况下的转发吞吐率和分组延迟。同时，该研究还考虑了转发表的大小以及 OpenFlow 表类型对于性能的影响。

4.1.2 定量评价

一些研究通过建模对 SDN 的可扩展性进行定量分析。

Jarschel 等[79]提出了一个 OpenFlow 网络架构下的基本模型用于分析转发速率和拥塞概率。该模型捕捉了分组通过控制器处理和只通过交换机处理的延迟作为对比，测算了控制器在高负载情况下的分组丢失率。模型仿真的结果表明分组的逗留时间主要取决于 OpenFlow 控制器的处理速率，从而表明控制器性能对于安装新流的重要性。

Jarschel 等[80]对 OpenFlow 硬件交换机进行了测量，并建立了一个带反馈的 M/M/1−S 排队系统来对具有一台控制器和一台 OpenFlow 交换机的 SDN 系统进行建模。这个模型可以用来估计分组在系统内的丢失率。

Azodolmolky 等[81]利用网络演算补充排队论对一台 SDN 控制器和一台交换机进行了建模。其中，网络演算主要用于各项参数边界和最坏情况的分析，排队论主要用于分析交换机的延迟以及队列长度的界限。

Fu 等[82]基于排队论提出了一种灵活的可休多控制器模型。该可休多控制器模型允许部分控制器在轻负载条件下进入休眠状态，以节省系统成本。同时，通过排队分析获得了系统的各种性能度量。

Yao 等[83]假设数据平面交换机到控制器的流建立请求为批量到达过程，采用排队模型 $M^k/M/1$ 对该系统进行建模，并采用排队论分析了控制器的性能，得到了流的平均服务时间。

Osgouei 等[84]发现在已有的 SDN 虚拟化方案中虚拟化导致了性能降低问题（如额外的分组转发延迟）。因此，该研究提出了一种针对虚拟化 SDN 网络切片的网络演算分析模型，用于分析虚拟化 SDN 网络切片的各项参数的上限值。

Xiong 等[85]采用 $M^X/M/1$ 模型对交换机的到达流进行建模，采用 M/G/1 模型对 SDN 控制器进行建模，通过开源测试工具 cbench 对控制器的延迟以及分组的逗留时间进行了验证。

4.2 规模可扩展评价

Hu 等[86]采用响应时间作为 SDN 控制平面可扩展体系结构的度量指标，对 3 种典型的 SDN 控制

平面体系结构（集中式体系结构、去中心化体系结构以及层次式体系结构）进行了评价和建模。进一步，将去中心化体系结构分为 2 种：去中心化本地视图以及去中心化全局视图。通过数值仿真实验，表明层次式体系结构和去中心化本地视图体系结构的可扩展性基本相同。以上 2 种结构的可扩展性好于去中心化全局视图体系结构。集中式体系结构的可扩展性最差。

Karakus 等[87]提出了一种度量指标用于评估 4 种体系结构的可扩展性。该研究用于可扩展性度量的指标定义为工作负载（W）与开销（O）之比，其中，工作负载（W）指通过数据平面进入网络的流的数量；开销（O）指控制平面中控制器处理的消息数量。基于提出的可扩展度量指标，对比了文献[86]中的 4 种体系结构，其结论为层次式体系结构的可扩展性最好；去中心化本地视图的体系结构的可扩展性次之；去中心化全局视图的体系结构的可扩展性再次之；集中式体系结构的可扩展性最差。

4.3　其他系统的可扩展评价方法

文献[88~90]介绍了分布式系统的可扩展评价方法。这些文献主要以计算资源的各项性能指标、负载情况、性能（如吞吐率、延迟等）几个方面对分布式系统的可扩展性进行评价。

文献[91~94]针对云计算系统的特点，从系统的性能（如响应时间和吞吐率）、计算能力、工作负载、带宽、存储容量、对不同用户数量提供服务的能力、QoS 及 SLA 等级等方面对云计算系统的可扩展性进行建模及测评。

软件定义网可扩展性的评价方法与分布式系统和云计算系统基本一致，主要从定性分析和定量分析 2 方面进行。但是，在进行软件定义网可扩展性评价时，主要侧重于 SDN 自身特点带来的性能及规模可扩展方面的问题进行评价。

5　可扩展性未来研究方向展望

首先，尽管现在提出了多种多控制器体系结构，但是控制器之间水平方向以及垂直方向通信接口的标准化尚有待进一步研究。

其次，现有 SDN 网络在实际中部署得不多，如何在现有互联网中进行增量部署，是一个值得研究的问题。

再次，由于 TCAM 价格昂贵、容量有限、其内部结构也并不是为 SDN 专门设计，因此，在 TCAM 之外寻找一种简单而高效的转发硬件也是未来可能的方向之一。

最后，关于 SDN 可扩展性评价方面的研究尚不是很充分，目前国际对这方面的研究还处于初步阶段，更多的度量指标与评价模型有待进一步研究与提出。

6　结束语

本文针对软件定义网路可扩展性进行了综述，从软件定义网络自身特征出发讨论了导致软件定义网络可扩展问题的 3 个主要原因。同时，本文从性能可扩展、地理位置可扩展、管控可扩展介绍了 SDN 可扩展问题的相关研究进展。进一步，介绍了软件定义网络可扩展性评价的现有研究。最后，对 SDN 可扩展性的未来研究方向进行了展望。

参考文献：

[1] THOMAS D N, XENGRAY K G. 软件定义网络：SDN 与 OpenFlow 解析[M]. 毕军译. 北京：人民邮电出版社, 2014.
THOMAS D N, XENGRAY K G. Software definition network: SDN and OpenFlow resolution[M]. Translation by BI J. Beijing: Posts & Telecom Press, 2014

[2] MERWE J V, ROONEY S, LESLIE I, et al. The tempest-a practical framework for network programmability[J]. IEEE Network, 1998, 12(3): 20–28.

[3] FEAMSTER N, REXFORD J, ZEGURA E. The road to SDN: an intellectual history of programmable networks[J]. ACM SIGCOMM Computer Communication Review, 2014, 14(2):87-98.

[4] YANG L, DANTU R T. ANDERSON, et al. Forwarding and control element separation (ForCES) framework[J]. Heise Ieitschriften Verlag, 2004.

[5] CAESAR M, CALDWELL D, FEAMSTER N, et al. Design and implementation of a routing control platform[C]//Symposium on Networked Systems Design & Implementation (NSDI'05). 2005: 15-28.

[6] GREENBERG A, HJALMTYSSON G, MALTZ D A, et al. A clean-slate 4D approach to network control and management[J]. ACM SIGCOMM Computer Communication Review, 2005, 35(5): 41-54.

[7] CASADO M, FREEDMAN M J, PETTIT J, et al. Ethane: taking control of the enterprise[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 1-12.

[8] CAMPBELL A T, KATZELA I, MIKI K, et al. Open signaling for ATM, Internet and mobile networks[J]. ACM SIGCOMM Computer Communication Review, 1999, 29(1): 97-108.

[9] ANGIN O, CAMPBELL A, KOUNAVIS M, et al. The mobiware toolkit: programmable support for adaptive mobile networking[J]. IEEE Personal Communications, 1998, 5(4):32-43.

[10] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al.Open-flow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[11] 唐宏, 刘汉江, 陈前锋, 等. OpenDaylight 应用指南[M]. 北京：人民

邮电出版社, 2016.
TANG H, LIU H J, CHEN Q F, et al. OpenDaylight application guide[J]. Beijing: Posts & Telecom Press, 2016.

[12] GORJA P, KURAPATI R. Extending open vSwitch to L4-L7 service aware OpenFlow switch[C]//Advance Computing Conference. 2014: 343-347.

[13] RICHARDSON L, RUBY S. RESTful Web services[M]. Beijing: Publishing House of Electronics Industry, 2008.

[14] FOSTER N, HARRISON R, FREEDMAN M J, et al. Frenetic: a network programming language[C]//ACM SIGPLAN International Conference on Functional Programming. Tokyo, Japan, 2011: 279-291.

[15] VOELLMY A, WANG J, YANG Y R, et al. Maple: simplifying SDN programming using algorithmic policies[C]//SIGCOMM 2013. 2013: 87-98.

[16] SEDDIKI M S, SHAHBAZ M, DONOVAN S, et al. Flow QoS: QoS for the rest of us[C]//ACM Workshop Hot Topics in Software Defined Networks. 2014: 207-208.

[17] BENSON T, ANAND A, AKELLA A, et al. MicroTE: fine grained traffic engineering for data centers[C]//Conference on Emerging Networking Experiments and Technologies (CONEXT'11). 2011.

[18] NEUMAN C B. Scale in distributed systems[M]//Readings in Distributed Computing Systems. IEEE Computer Society Press, 1994.

[19] CURTIS A R, MOGUL J C, TOURRILHES J, et al. Devoflow: scaling flow management for high-performance networks[C]//ACM SIGCOMM'11. 2011: 254-265.

[20] BENSON T, AKELLA A, MALTZ D A. Network traffic characteristics of data centers in the wild[C]//ACM SIGCOMM Conference on Internet Measurement (IMC'10). Melbourne, Australia, 2010: 267-280.

[21] HE K, KHALID J, JACOBSON A G, et al. Measuring control plane latency in SDN-enabled switches[C]//ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR'15). Santa Clara, California, 2015.

[22] GUAN X, CHOI B Y, SONG S. Reliability and scalability issues in software defined network frameworks[C]//Research and Educational Experiment Workshop (GREE). 2013.

[23] TAVAKOLI A, CASADO M, KOPONEN T, et al. Applying NOX to the datacenter[C]//ACM Workshop on Hot Topics in Networks (HotNets'09). 2009.

[24] FU Y H, BI J, CHEN Z, et al. A hybrid hierarchical control plane for flow-based large-scale software-defined networks[J]. IEEE Transactions on Network and Service Management, 2015, 12(2): 117-131.

[25] CAI Z, COX A L, EUGENENG T S. Maestro: balancing fairness, latency and throughput in the OpenFlow control plane[R]. Rice University Technical Report, TR11-07, 2011.

[26] GUDE N, KOPONEN T, PETTIT J, et al. NOX: towards an operating system for networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105-110.

[27] TOOTOONCHIAN A, GORBUNOV S, GANJALI Y, et al. On controller performance in software defined networks[C]//USENIX Workshop on Hot Topics in Management of Internet (Hot-ICE'12). 2012.

[28] ERICKSON D. The beacon OpenFlow controller[C]//ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13). 2013: 13-18.

[29] SHIN S, SONG Y, LEE T, et al. Rosemary: a robust, secure, and high performance network operating system[C]//ACM Conference on Computer and Communications Security. 2014: 78-89.

[30] RENART E G, ZHANG E G, NATH B. Towards a GPU SDN controller[C]//Workshop on Software-Defined Networking and Network Function Virtualization for Flexible Network Management, 2014.

[31] IVASHCHENKO P, SHALIMOV A, SMELIANSKY R. High performance in-kernel SDN/OpenFlow controller[C]//USENIX Open Networking Summit Research Track, 2014.

[32] ZHAO Y, IANNONE L, RIGUIDEL M. On the performance of SDN controllers: a reality check[C]//Network Function Virtualization and Software Defined Network, 2015.

[33] MEDVED J, VARGA R, TKACIK A, et al. OpenDaylight: Towards a model-driven SDN controller architecture[C]//World of Wireless, Mobile and Multimedia Networks. 2014: 1-6.

[34] FERNANDEZ M. Evaluating OpenFlow controller paradigms[C]// International Conference on Advanced Information Networking and Applications, 2013.

[35] YU M, REXFORD J, FREEDMAN M J, et al. Scalable flow-based networking with DIFANE[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 351-362.

[36] 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.

[37] MOGUL J C, CONGDON P. Hey, you darned counters! Get off my ASIC![C]//ACM Workshop on Hot Topics in Software Defined Networks. Helsinki, Finland, 2012: 25-30.

[38] FERKOUSS O E, SNAIKI I, MOUNAOUAR O, et al. A 100Gig network processor platform for OpenFlow[C]//International Conference on Network and Services Management. International Federation for Information Processing. 2011:1-4.

[39] MEMON G, VARVELLO M, LAUFER R, et al. FlashFlow: a GPU-based fully programmable openflow switch[R]. University of Oregon, Technic Report, 2013.

[40] LUO Y, CASCON P, MURRAY E, et al. Accelerating OpenFlow switching with network processors[C]//Symposium on Architectures for Networking and Communications Systems. 2009: 70-71.

[41] STEPHENS B. Designing scalable networks for future large datacenters[D]. Rice University, 2012.

[42] LI Y, ZHANG D, HUANG K, et al. A memory-efficient parallel routing lookup model with fast updates[J]. Computer Communications, 2014, 38(1): 60-71.

[43] BRAUN W, MENTH M. Wildcard compression of inter-domain routing tables for OpenFlow-based software-defined networking[C]// European Workshop on Software Defined Networks (EWSDN). 2014.

[44] AGARWAL K, DIXON C, ROZNER E, et al. Shadow MACs: scalable label-switching for commodity Ethernet[C]//ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'14). 2014: 157-162.

[45] TOOTOONCHIAN A, GANJALI Y. HyperFlow: a distributed control plane for OpenFlow[C]//Internet Network Management on Research on Enterprise Networking, 2010.

[46] KOPONEN T, CASADO M, GUDE N, et al. Onix: a distributed control platform for large-scale production networks[C]//USENIX Conference on Operating Systems Design and Implementation. 2010: 351-364.

[47] BERDE P, GEROLA M, HART J, et al. ONOS: towards an open, distributed SDN OS[C]//ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'14). 2014: 1-6.

[48] MONACO M, MICHEL O, KELLER E. Applying operating system principles to SDN controller design[C]//ACM Workshop on Hot Topics in Networks (HotNets'13). 2013.

[49] YEGANEH S H, GANJALI Y. Kandoo: a framework for efficient and scalable offloading of control applications[C]//ACM Workshop on Hot Topics in Software Defined Networks (HotSDN'12). 2012: 19-24.

[50] MCCAULEY J, PANDA A, CASADO M, et al. Extending SDN to large-scale networks[C]//Open Network Summit. 2012: 19-24.

[51] DIXIT A, HAO F, MUKHERJEE S, et al. ElastiCon: an elastic distributed SDN controller[C]//ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '14). 2014: 17-28.

[52] GHODSI A. Distributed *k*-ary system: algorithms for distributed hash tables[D]. The Royal Institute of Technology, 2006.

[53] KREUTZ D, RAMOS F M V, VERISSIMO P E, et al. Software-defined networking: a comprehensive survey[J]. Proceedings of the IEEE, 2015, 103(1): 14-76.

[54] LAKSHMAN A, MALIK P. Cassandra: a decentralized structured storage system[J]. ACM Sigops Operating Systems Review, 2010, 44(2): 35-40.

[55] YU H, LI K, QI H, et al. Zebra: an east-west control framework for SDN controllers[C]//2015 44th International Conference on Parallel Processing (ICPP). 2015: 610-618.

[56] BENAMRANE F, MAMOUN M B, BENAINI R. An east-west interface for distributed SDN control plane: implementation and evaluation[J]. Computers & Electrical Engineering, 2017, 57: 162-175.

[57] YIN H, XIE H, TSOU T, SDNi: a message exchange protocol for software defined networks (SDNS) across multiple domains[Z]. IETF Working Documents BCP 78, 2012.

[58] LIN P, BI J, WOLFF S, et al. A westeast bridge based SDN inter-domain testbed[M]. IEEE Communications Magazine, 2015, 53(2): 190-197.

[59] HELLER B, SHERWOOD R, MCKEOWN N. The controller placement problem[C]//ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks (HotSDN'12). 2012: 7-12.

[60] YAO G, BI J, LI Y, et al. On the capacitated controller placement problem in software defined networks[J]. IEEE Communications Letters, 2014, 18(8): 1339-1342.

[61] HU Y N, WANG W D, GONG X Y, et al. Reliability-aware controller placement for software-defined networks[J]. IFIP/IEEE International Symposium on Integrated Network Management (IM), 2013, 11(2): 672-675.

[62] MULLER L F, OLIVEIRA R R, LUIZELLI M C, et al. Survivor: an enhanced controller placement strategy for improving SDN survivability[C]//Global Communications Conference (GLOBECOM'14). 2014: 8-12.

[63] RATH H K, REVOORI V, NADAF S, et al. Optimal controller placement in software defined networks (SDN) using a non-zero-sum game[C]//World of Wireless, Mobile and Multimedia Networks. 2014: 1-6.

[64] LANGE S, GEBERT S, ZINNER T, et al. Heuristic approaches to the controller placement problem in large scale SDN networks[J]. IEEE Transactions on Network and Service Management, 2015, 12(1): 4-17.

[65] JAIN S, KUMAR A, MANDAL S, et al. B4: experience with a globally-deployed software defined wan[C]//ACM SIGCOMM'13. 2013: 3-14.

[66] HONG C Y, KANDULA S, MAHAJAN R, et al. Achieving high utilization with software-driven WAN[C]//ACM SIGCOMM'13. 2013: 15-26.

[67] PHEMIUS K, BOUET M, LEGUAY J. DISCO: distributed multi-domain SDN controllers[C]//Network Operations and Management Symposium (NOMS). 2014.

[68] KANDULA S, MENACHE I, SCHWARTZ R, et al. Calendaring for wide area networks[C]//ACM SIGCOMM'14. 2014.

[69] ZHANG H, CHEN K, BAI W, et al. Guaranteeing deadlines for inter-datacenter transfers[C]//European Conference on Computer Systems (Euro Sys'15). 2015.

[70] JIN X, LI Y, WEI D, et al. Optimizing bulk transfers with software-defined optical WAN[C]//ACM SIGCOMM'16. 2016: 87-100.

[71] GUPTA A, VANBEVER L, SHAHBAZ M, et al. SDX: software defined Internet exchange[C]//ACM SIGCOMM'15. 2015: 551-562.

[72] GUPTA A, MACDAVID R, BIRKNER R, et al. An industrial-scale software defined Internet exchange point[C]//USENIX Symposium on Networked Systems Design and Implementation (NSDI '16). 2016: 16-18.

[73] WANG Y, BI J, LIN P, et al. SDI: a multi-domain SDN mechanism for fine-grained inter-domain routing[C]//Annals of Telecommunications, 2016.

[74] LIN P, HART J, KRISHNASWAMY U, et al. Seamless interworking of SDN and IP[C]//ACM SIGCOMM'13. 2013: 475-476.

[75] SHALIMOV A, ZUIKOV D, ZIMARINA D, Advanced study of SDN/OpenFlow controllers[C]//The 9th Central & Eastern European Software Engineering Conference. 2013: 24-25.

[76] SHAH S A, FAIZ J, FAROOQ M, et al. An architectural evaluation of SDN controllers[C]//IEEE International Conference on Communications (ICC). 2013: 3504-3508.

[77] ROTSOS C, SARRAR N, UHLIG S, et al. OFLOPS: an open framework for OpenFlow switch evaluation[C]//International Conference on Passive and Active Network Measurement. 2012: 85-95.

[78] BIANCO A, BIRKE R, GIRAUDO L, et al. OpenFlow switching: data plane performance[C]// 2010 IEEE International Conference on Communications (ICC). 2010: 1-5.

[79] JARSCHEL M, OECHSNER S, SCHLOSSER D, et al. Modeling and performance evaluation of an OpenFlow architecture[C]//The 23rd International Teletraffic Congress. 2011: 1-7.

[80] JARSCHEL M, OECHSNER S, SCHLOSSER D, et al. Modeling and performance evaluation of an OpenFlow architecture[C]//The 23rd International Teletraffic Congress. 2011: 1-7.

[81] AZODOLMOLKY S, NEJABATI R, PAZOUKI M, et al. An analytical model for software defined networking: a network calculus-based approach[C]//Global Communications Conference (GLOBECOM). 2013: 1397-1402.

[82] FU Y H, BI J, WU J P, et al. A dormant multi-controller model for software defined networking[J]. China Communications, 2014, 11(3): 45-55.

[83] YAO L, HONG P, ZHOU W. Evaluating the controller capacity in software defined networking[C]//Computer Communication and Networks (ICCCN). 2014: 1-6.

[84] OSGOUEI A G, KOOHANESTANI A K, SAIDI H, et al. Analytical performance model of virtualized SDNs using network calculus[C]//Electrical Engineering (ICEE). 2015: 770-774.

[85] XIONG B, YANG K, ZHAO J, et al. Performance evaluation of OpenFlow-based software-defined networks based on queueing model[J]. Computer Networks, 2016, 102: 172-185.

[86] HU J, LIN C, LI X Y, et al. Scalability of control planes for software defined networks: modeling and evaluation[C]//International Symposium of Quality of Service (IWQoS). 2014.

[87] KARAKUS M, DURRESI A. A scalability metric for control planes in software defined networks (SDNs)[C]//2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA). 2016: 282-289.

[88] JOGALEKAR P, WOODSIDE M. Evaluating the scalability of distributed systems[J]. IEEE Transactions on Parallel and Distributed systems, 2000, 11(6): 589-603.

[89] PASTOR L, BOSQUE J L. An efficiency and scalability model for heterogeneous clusters[C]//IEEE International Conferonce on Clusted Computing. 2001: 427.

[90] SRINIVAS A V, JANAKIRAM D. A model for characterizing the scalability of distributed systems[J]. ACM Sigops Operating Systems Review, 2005, 39(3): 64-71.

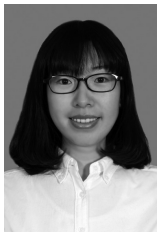
[91] COOPER B F, SILBERSTEIN A, TAM E, et al. Benchmarking cloud serving systems with YCSB[C]//The 1st ACM Symposium on Cloud Computing. 2010: 143-154.

[92] GAO J, PATTABHIRAMAN P, BAI X, et al. SaaS performance and scalability evaluation in clouds[C]//2011 IEEE 6th International Symposium on Service Oriented System Engineering (SOSE). 2011: 61-71.

[93] HWANG K, BAI X, SHI Y, et al. Cloud performance modeling with benchmark evaluation of elastic scaling strategies[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(1): 130-143.

[94] HWANG K, SHI Y, BAI X. Scale-out vs scale-up techniques for cloud performance and productivity[C]//Cloud Computing Technology and Science (CloudCom). 2014: 763-768.

作者简介：



付永红（1981-），女，黑龙江大庆人，清华大学博士生，主要研究方向为软件定义网可扩展性、网络体系结构设计、路由协议等。

毕军（1972-），男，辽宁大连人，清华大学教授、博士生导师，主要研究方向为网络空间安全、软件定义网络、网络体系结构、源地址验证、域间路由协议、NDN 网络等。

张克尧（1993-），男，安徽合肥人，清华大学硕士生，主要研究方向为 SDN 控制平面与数据平面可扩展性、域间 SDN 互连机制、SDN 数据平面可编程。

吴建平（1953-），男，山东巨野人，中国工程院院士，清华大学教授、博士生导师，主要研究方向为网络空间安全、源地址验证、IPv4 与 IPv6 过渡、网络体系结构、源地址驱动的网络等。