IJDCST @ May-June-2021, Issue- V-7, I-1, SW-14
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Openflow Protocol in Software Defined Networks

V. Rajesh[1], S. Sindhura[2]
[1]Assistant Professor, CSE Department, PVP Siddhartha Institute of Technology
[2]Assistant Professor, CSE Department, KLEF

***Abstract*—The Internet has led to the creation of a digital society, where (almost) everything is connected and is accessible from anywhere. However, despite their widespread adoption, traditional IP networks are complex and very hard to manage. It is both difficult to configure the network according to predefined policies, and to reconfigure it to respond to faults, load and changes. To make matters even more difficult, current networks are also vertically integrated: the control and data planes are bundled together. Software-Defined Networking (SDN) is an emerging paradigm that promises to change this state of affairs, by breaking vertical integration, separating the network's control logic from the underlying routers and switches, promoting (logical) centralization of network control, and introducing the ability to program the network. The separation of concerns introduced between the definition of network policies, their implementation in switching hardware, and the forwarding of traffic, is key to the desired flexibility: by breaking the network control problem into tractable pieces, SDN makes it easier to create and introduce new abstractions in networking, simplifying network management and facilitating network evolution. In this paper we present a comprehensive survey on SDN. We start by introducing the motivation for SDN, explain its main concepts and how it differs from traditional networking, its roots, and the standardization activities regarding this novel paradigm. Next, we present the key building blocks of an SDN infrastructure using a bottom-up, layered approach. We provide an in-depth analysis of the hardware infrastructure, southbound and northbound APIs, network virtualization layers, network operating systems (SDN controllers), network programming languages, and network applications. We also look at cross-layer problems such as debugging and troubleshooting. In an effort to anticipate the future evolution of this new paradigm, we discuss the main ongoing research efforts and challenges of SDN. In particular, we address the design of switches and control platforms – with a focus on aspects such as resiliency, scalability, performance, security and dependability – as well as new opportunities for carrier transport networks and cloud providers. Last but not least, we analyze the position of SDN as a key enabler of a software-defined environment. . OpenFlow enables a central controller to remotely provision the underlying data plane device forwarding tables in a common,scalable way, and eliminates the vendor-specific, proprietary nature of legacy networking equipment. Specifically, OpenFlow enables automation through a centralized software controller that eliminates the need to program devices and interfaces for every network service request.**

***Index Terms*—Software-defined networking,OpenFlow, netwo -rk virtualization,networkoperatingsystems ,programmable net worksnetworkhypervisor,programminglanguages,flowbased networking, scalability , dependability, carrier-grade networks,software-defined environments.**

1. INTRODUCTION

The distributed control and transport network protocols running inside the routers and switches are the key technologiesthat allow information, in the form of digital packets, totravel around the world. Despite their widespread adoption,traditional IP networks are complex and hard to manage.To express the desired high-level network policies, networkoperators need to configure each individual network deviceseparately using low-level and often vendor-specific commands.In addition to the configuration complexity, networkenvironments have to endure the dynamics of faults andadapt to load changes. Automatic reconfiguration andresponsemechanisms are virtually non-existent in current IP networks.Enforcing the required policies in such a dynamic environmentis therefore highly challenging.To make it even more complicated, current networks arealso vertically integrated. The control plane (that decides howto handle network traffic) and the data plane (that forwardstraffic according to the decisions made by the control plane)are bundled inside the networking devices, reducing flexibilityand hindering innovation and evolution of the networkinginfrastructure. The transition from IPv4 to IPv6, started morethan a decade ago and still largely incomplete, bears witnessto this challenge, while in fact IPv6 represented merely aprotocol update. Due to the inertia of current IP networks,a new routing protocol can take 5 to 10 years to be fullydesigned, evaluated and deployed. Likewise, a clean-slate approach to change the Internet architecture (e.g., replacingIP), is regarded as a daunting task – simply not feasible inpractice. Ultimately, this situation has inflated thecapital and operational expenses of running an IP network.Software-Defined Networking (SDN) is an emergingnetworking paradigm that gives hope to change the limitationsof current network infrastructures. First, it breaksthe vertical integration by separating the network's controllogic (the control plane)
from the underlying routers andswitches that forward the traffic (the data plane). Second,with the separation of the control and data planes, networkswitches become simple forwarding devices and the controllogic is implemented in a logically centralized controller (or network operating system).

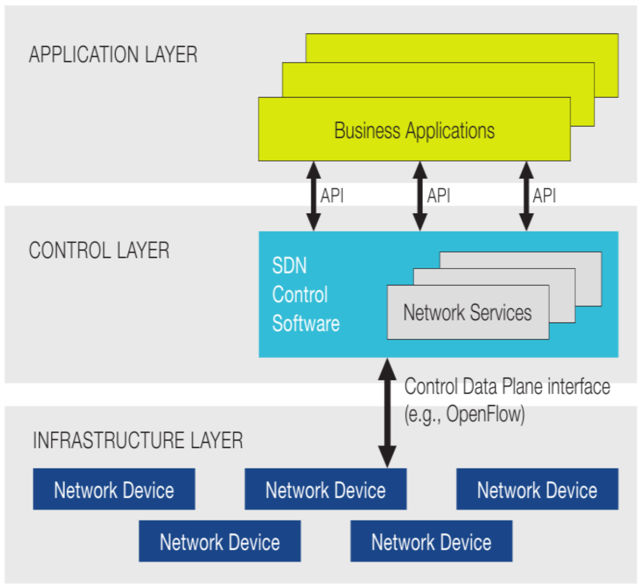

Figure 1:SDN System Architecture

A simplifiedview of this architecture is shown in Figure 1. It is importantto emphasize that a logically centralized programmatic modeldoes not postulate a physically centralized system. In fact,the need to guarantee adequate levels of performance, scalability, reliability would include such a solution. Instead , production level SDN networkdesigns resort to physically distributed control planes.

The separation of the control plane and the data planecan be realized by means of a well-defined programminginterface between the switches and the SDN controller. The controller exercises direct control over the state in the dataplane elements via this well definedapplication programming interface (API), as depicted in Figure 1. The most notable example of such an API is OpenFlow.AnOpenFlow switch has one or more tables of packet-handling rules (flowtable). Each rule matches a subset of the traffic and performs certain actions(dropping,forwarding,modifying etc.) on the traffic. Depending on the rules installed by a controller application, an OpenFlow switch can – instructed by the controller – behave like a router, switch, firewall, or perform other roles (e.g., load balancer, traffic shaper, and in general those of a middle box).

An important consequence of the software-defined networking principles is the separation of concerns introduced between the definition of network policies, their implementation in switching hardware, and the forwarding of traffic. This separation is key to the desired flexibility, breaking then network control problem into tractable pieces, and making it easier to create and introduce new abstractions in networkingsimplifying network management and facilitating network evolution and innovation. AlthoughSDN and OpenFlow started as academic experiments they gained significant traction in the industry over the past few years. Most vendors of commercial switches now include support of the OpenFlow API in their equipment.SDN momentum was strong enough make Google,Facebook,Yahoo,Microsoft, Deutsche Telekom fund OpenNetworkFoundation(ONF) with the main goalof promotion and adoption of SDN through open standards development. As the initial concerns with SDN scalabilitywere addressed – in particular the myth that logicalcentralization implied a physically centralized controller, anissue we will return to later on – SDN ideas have maturedand evolved from an academic exercise to a commercialsuccess. Google, for example, has deployed a software-definednetwork to interconnect its data centers across the globe.This production network has been in deployment for 3 years,helping the company to improve operational efficiency and significantly reduce costs.VMware's network virtualizationPlatform. As a final example, theworld's largest IT companies (from carriers and equipmentmanufacturers to cloud providers and financial-services companies)have recently joined SDN consortia such as the ONFand the OpenDaylight initiative another indication of theimportance of SDN from an industrial perspective.

II. What is Software-Defined Networking?

The Open Networking Foundation (ONF) is the group that is most associated with the development and standardization of SDN. According to the ONF. "Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow protocol is a foundational element for building SDN solutions."

We define an SDN as a network architecture with five pillars:

- ➢ .***Directly programmable:***Network control is directly programmable because it is decoupled from forwarding functions.
- ➢ ***Agile****:*Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- ➢ ***Centrally managed****:*Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- ➢ ***Programmatically configured***: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- ➢ ***Openstandards based vendor neutral****:*When implemented through open standards, SDN simplifies network design and operation because

instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

The strong coupling betweencontrol and data planes has made it difficult to add newfunctionality to traditional networks, a fact illustrated inFigure 6 The coupling of the control and data planes(andits physical embedding in the network elements) makes thedevelopment and deployment of new networking features(e.g., routing algorithms) very hard since it would imply amodification of the control plane of all network devices –through the installation of new firmware and ,in some cases hardware upgrades.Hence,theNewnetworkingfeaturesarecommonly introduced via expensive, specializedand hard-toconfigureequipment(akamiddle boxes) such as load balancers,intrusion detection system(IDS).
andfirewalls, among others.These middleboxes need to be placed strategically in thenetwork, making it even harder to later change the networktopology,configuration,andfuncti-onality.In contrast, SDN decouples the control plane from thenetwork devices and becomes an external entity: the network
operatingsystem or SDN controller. This approach has severaladvantages:

- ❖ All applications can take advantage of the same networkinformation (the global network view), leading (arguably) to more consistent and effective policy decisions while re-using control plane software modules.

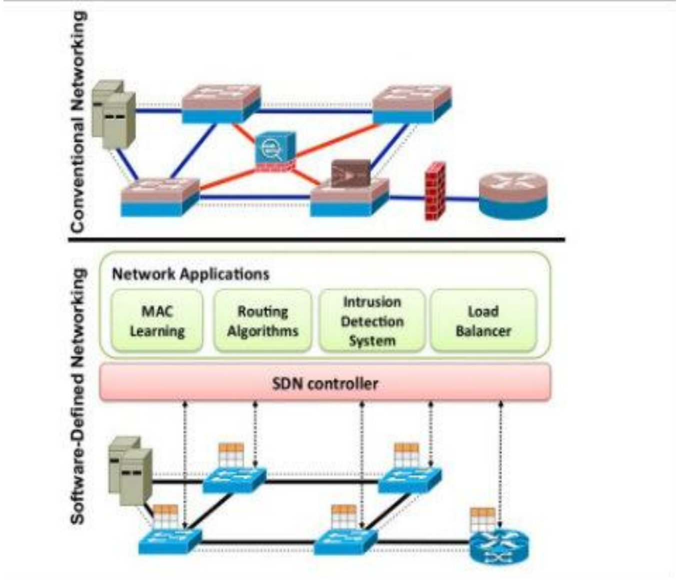

Fig 2: Traditionalnetworking versus SoftwareDefined Networking (SDN)

- ❖ Theseapplications can take actions(,reconfigure forwardingdevices) from any part of the network. There is therefore no need to devise a precise strategy about the location of the new functionality.
- ❖ The integration of different applications becomes more Straightforward. For instance, load balancing and routing applications can be combined sequentially, with load balancing decisions having precedence over routing policies.

A.Terminology

To identify the different elements of an SDN as unequivocallyas possible, we now present the essential terminologyused throughout this work.

Forwarding Devices (FD): Hardware- or software-based dataplanedevices that perform a set of elementary operations. TheForwardingdevices have well-defined instruction sets (e.g.,Flow rules) used to take actions on the incoming packets(e.g., forward to specific ports, drop, forward to the controller,rewrite some header). These instructions are defined by southbound interfaces(e.g.,OpenFlow ,For CES Protocol-Oblivious Forwarding (POF)) and are installed in theforwarding devices by the SDN controllers implementing the
southbound protocols.

Data Plane (DP):Forwarding devices are interconnected throughwireless radio channels or wired cables. The net-workinfrastructure comprises the interconnected forwardingdevices, which represent the data plane.

Southbound Interface (SI): The instruction set of the forward-ingdevices is defined by the southbound API, which is part of the southbound interface. Furthermore, the SI also defines the communication protocol between forwarding devices andcontrol plane elements. This protocol formalizes the way thecontrol and data plane elements interact.

Control Plane (CP**):** Forwarding devices are programmed bycontrol plane elements through well-defined SI embodiments.The control plane can therefore be seen as the “network brain”All control logic rests in the applicationsand controllers, which form the control plane.

Northbound Interface (NI): The network operating system canoffer an API to application developers. This API represents anorthbound interface, i.e., a common interface for developingapplications. Typically, a northbound interface abstracts thelow level instruction sets used by southbound interfaces toprogram forwarding devices.

Management Plane (MP**):**The management plane is the set of applicationsthat leverage the functions offered by the NI to implement networkcontrol and operationlogic.Thisincludes applicationssuchas routing firewalls ,load balancers
monitoring, and so forth.Essentially, a management applica tiondefines the policies, which are ultimately translated to southbound-specific instructions that program the behavior ofthe forwarding devices.

B. Alternative and Broadening Definitions:

The definition of SDN will likely continue tobroaden, drivenby theindus -try business-oriented viewson SDN– irrespectiveof thedecouplingof the control plane. In this survey, wefocus on the original,"canonical" SDN definition based on the forementionedkey pillars and the conceptoflayered abstractions.However,for the sake ofcompleteness and clarity, weacknowledge alternative SDN definitions ,including:*Control Plane / Broker SDN*: A networking approachthatretains existing distributed control planes but offers newAPIs that allow applicationsto interact (bidirectionally) withthe network. An SDN controller –often called orchestration platform– acts as a broker between the applications and thenetwork elements. This approacheffectively presents controlplane data to the application and allows a certain degree ofnetwork programmability by means of "plug-ins" between the orchestrator function andnetwork protocols. This API-driven approach corresponds to a hybrid model of SDN, since it enablesthe broker to manipulate and directly interact with the control planes of devices such as routers and switches.Examplesofthis view on SDN include recent standardizationefforts at IETF -and the design philosophybehind theOpenDaylight project that goes beyondtheOpenFlow split control mode.

III. OPENFLOW PROTOCOL:

The OpenFlow protocol is the most commonly used protocol for the southbound interface of SDN, which separates the data plane from the control plane. The white paper about OpenFlow points out the advantages of a flexibly configurable forwarding plane. OpenFlow was initially proposed by Stanford University, and it is now standardized by the ONF . In the following, we first give an overview of the structure of OpenFlow and then describe the features supported by the different specifications.

Overview:

The OpenFlow architecture consists of three basic concepts. (1) The network is built up by OpenFlow-compliant switches that compose the data plane; (2) the control plane consists of one or more OpenFlow controllers; (3) a secure control channel connects the switches with the control plane. In the following, we discuss OpenFlow switches and controllers and the interactions among them. An OpenFlow-compliant switch is a basic forwarding device that forwards packets according to its flow table. This table holds a set of flow table entries, each of which consists of match fields, counters and instructions. Flow table entries are also called flow rules or flow entries. The "header fields" in a flow table entry describe to which packets this entry is applicable. They consist of a wildcard-capable match over specified header fields of packets. To allow fast packet forwarding with OpenFlow, the switch requires ternary content addressable memory (TCAM) that allows the fast lookup of wildcard matches. The header fields can match different protocols depending on the OpenFlow specification, e.g., Ethernet, IPv4, IPv6 or MPLS. The "counters" are reserved for collecting statistics about flows. They store the number of received packets bytes, as well as the duration of the flow. The "actions" specify how packets of that flow are handled. Common actions are "forward", "drop","modifyfield", etc.

HeaderFields	Counters	Actions

Figure 3:Flow table entry for openflow

A.Openflow:

A software program, called the controller, is responsible for populating and manipulating the flow tables of the switches. By insertion, modification and removal of flow entries,thecontroller can modify the behavior of theswitches with regard to forwarding. The OpenFlow specification defines the protocol that enables the controller to instruct the switches. To that end, the controller uses a secure control channel. Three classes of communication exist in the OpenFlow protocol: controller-to-switch, asynchronous and symmetric communication .The controller-to-switch communication is responsible for feature detection, configuration, programming the switch and information retrieval. Asynchronous communication is initiated by the OpenFlow-compliant switch without any solicitation from the controller. It is used to inform the controller about packet arrivals, state changes at the switch and errors. Finally, symmetric messages are sent without solicitation from either side, i.e., the switch or the controller are free to initiate the communication without solicitation from the other side. Examples for symmetric communication are hello or echo messages that can be used to identify whether the control channel is still live and available.

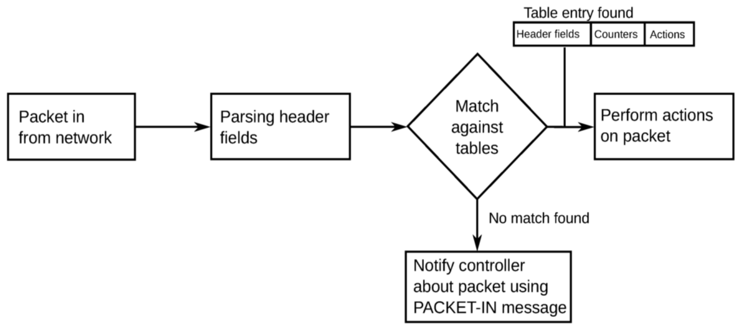

Figure 4: Basic packet forwarding with OpenFlow in a switch

The basic packet forwarding mechanism with OpenFlow is illustrated in Figure 4. When a switch receives a packet, it parses the packet header, which is matched against the flow table. If a flow table entry is found where the header field wildcard matches the header, the entry is considered. If several such entries are found, packets are matched based on prioritization, i.e., the most specific entry or the wildcard with the highest priority is selected. Then, the switch updates the counters of that flow table entry. Finally, the switch performs the actions specified by the flow table entry on the packet, e.g., the switch forwards the packet to a port. Otherwise, if no flow table entry matches the packet header, the switch generally notifies its controller about the packet, which is buffered when the switch is capable of buffering. To that end, it encapsulates either the unbuffered packet or the first bytes of the buffered packet using a PACKET-IN message and sends it to the controller; it is common to encapsulate the packet header and the number of bytes defaults to 128. The controller that receives the PACKET-IN notification identifies the correct action for the packet and installs one or more appropriate entries in the requesting switch. Buffered packets are then forwarded according to the rules; this is triggered by setting the buffer ID in the flow insertion message or in explicit PACKET-OUT messages. Most commonly, the controller sets up the whole path for the packet in the network by modifying the flow tables of all switches on the path.

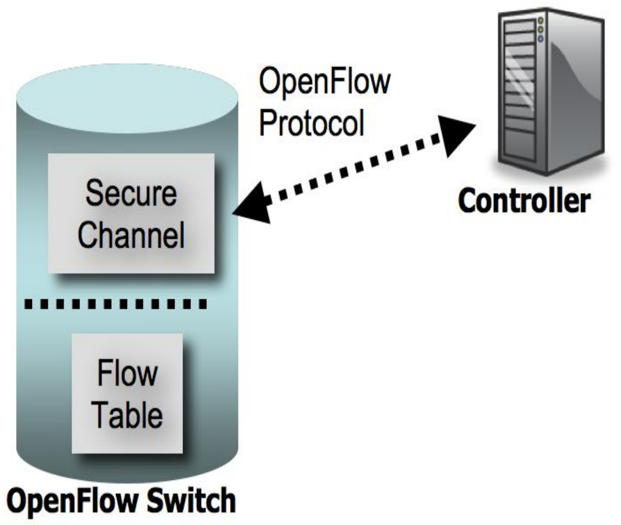

Figure 5:OpenFlow enabled SDN devices

IV .*DESIGN CHOICES FOR OPENFLOW BASED SDN*

Today, SDN is mostly used for flexible and programmable data centers. There is a need for network virtualization, energy efficiency and dynamic establishment and enforcement of network policies. An important feature is the dynamic creation of virtual networks, commonly referred to as network-as-a-service (NaaS). Even more complex requirements arise in multi-tenant data center environments. SDN can provide these features easily, due to its flexibility and programmability. Future Internet 2014, 6 320 However, SDN is also discussed in a network or Internet service provider (ISP) context. Depending on the use case, the design of SDN architectures varies a lot. In this section, we point out architectural design choices for SDN. We will discuss their implications with regard to performance, reliability and scalability of the control and data plane and refer to research on these topics.

V.*DISCUSSIONS OFOPENFLOW-BASED SDN:*

In this work, we have shown that OpenFlow-based SDN provides than conventional networking architectures, so that new features and network applications can be added to networks more easily. Researchers have analyzed OpenFlow-based SDN in various networking areas and showed improvements, even for complex networking tasks and features. We presented network applications in the fields of network security, traffic engineering, network management, middlebox networking, virtualization and inter-domain routing . All those network applications are facilitated by the SDN control plane. It provides a consistent and global view of the

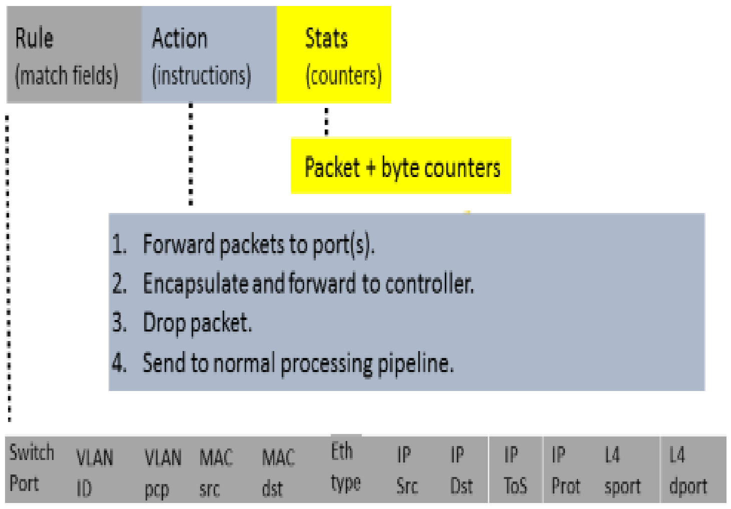

Figure6 :flow table entry

network, which enables network control algorithms to be written in a simplified fashion. The control plane is software-based and can run on powerful server hardware with lots of memory and modern CPUs, which enables computation-intensive route calculations in practice, such as traffic engineering and route optimization. Moreover, the number of control elements in OpenFlow-based SDN is usually smaller than the number of forwarding elements, which facilitates upgrades. However, OpenFlow-based SDN possibly requires data plane updates for new protocols, if the set of operations offered by an OpenFlow specification is insufficient. The

OpenFlow protocol provides more flexibility because new match fields can be defined using the OpenFlow Extensible Match (OXM). Other southbound interfaces are more flexible than OpenFlow, e.g., Forces offers a more programmable data plane. As the control server sets the flow table entries in OpenFlow switches via the OpenFlow protocol, the frequency of configuration requests by OpenFlow switches may drive the control server to its limit, Future Internet 2014, 6328 so that a bottleneck may occur. Such situations may happen in the presence of a large number of fine-grained flow table entries that occur in large networks with many switches and end-hosts or in the presence of network failures or updates when many flows need to be simultaneously rerouted. We presented various works that discuss and improve the control plane scalability of OpenFlow by leveraging hierarchical control structures or intelligent flow rule distribution. Software-based network applications enable network innovation. Nonetheless, complex software often contains many bugs, which also holds for network control algorithms. Failures in network control software can cause failures and outages in a network. Therefore, the correctness of network applications, which are applied to critical infrastructure, have to be correct and well tested before they are deployed. We discussed several approaches for the testing and verification of SDN software . In addition, the OpenFlow data plane faces scalability issues. OpenFlow switches support wildcard matches that are used to classify packets to flows. Thus, fast packet forwarding for OpenFlow requires special hardware: TCAM is used in switches for the fast lookup of wildcard matches. TCAM size is often very limited, due to the high cost. Serious scalability problems can occur when many flow table entries are needed. The number of necessary flow table entries increases for larger networks, excessive use of fine-grained matches and data plane resilience methods. We discussed several proposals in that improve the performance of the OpenFlow data plane, especially with regard to the limited number of flow table entries. This discussion shows that it is recommendable to investigate prior to deployment whether the advantages of OpenFlow-based SDN solutions outweigh its scalability concerns, which both depend on the specific use case.

References:

[1] T. Benson, A. Akella, and D. Maltz, “Unraveling the complexity ofnetwork management,” in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI’09,
Berkeley, CA, USA, 2009, pp. 335–348.

[2]B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, andS. Shenker, “Software-defined internet architecture:Decoupling architecture from infrastructure”, inProceedings of the 11thWorkshop on Hot Topics in Networks, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 43–48.

[3] "OpenFlow: Enabling Innovation in Campus Networks McKeown, T. Andershnan, G.Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turneron, H. BalakrisACMComputer Communication Review, Vol. 38, Issue 2, pp. 69-74 April 2008

[4] Richard Wang, Dana Butnariu, and Jennifer Rexford OpenFlow-based server load balancing gone wild, Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), Boston, MA, March 2011.

[5] NOX: Towards an Operating System for Networks
https://sites.google.com/site/routeflow/home
http://www.openflow.org/