A SURVEY OF SOFTWARE DEFINED NETWORK: THE RISE OF FUTURE NETWORK

[1]Mehul Manani,[2]Nital Prajapati
[1]PG Scholar,[2]Asst. Professor
[1]Computer Engineering Department,
[1]Shri Sa'd Vidya Mandal Institute of Technology, Bharuch, India.

Abstract: With the advancement in network technology, network management becomes more complex and time-consuming. The high bandwidth requirement, dynamic network management, scalability, and flexibility are critical issues in traditional networks. Software Defined Networking (SDN) brings new hope for the modern complex network architectures and promises to fulfill the requirements. SDN offers several advantages like network programmability, simpler configuration management, better security, performance improvement over traditional networking. SDN achieves the performance by separating the control plane from the data plane from networking devices. Network programmability also motivates more research opportunities. However, SDN is still in its growing phase, it requires attention by academicians and technocrats. This paper aims to conduct a survey on various basic aspects of SDN which includes history, definition, and architecture. It also covers the applications and advantages of the SDN over the conventional network. Moreover, this survey paper also concludes the wide future research areas of this revolutionary network architecture for potential researchers. Overall this survey paper can show the path to the academicians and industry research and development teams to put their first step in the ocean of the SDN.

IndexTerms **– Software Defined Network, Survey of SDN, Future of Network, SDN advantages, SDN applications.**

I. INTRODUCTION

The evolutions of cloud and data center network, network virtualization have changed the workload of the current networking infrastructure. These existing networking devices suffer from flexibility and scalability. The modern network traffic is dynamic in nature and requires crucial attention while deciding the path for traffic flow. It also requires more bandwidth and more computing resources. The traditional networking devices cannot perform the dynamic operations very efficiently because the control layer is tightly coupled with the forwarding layer. This bonding limits the functionality of the network administrator. The limitations of the traditional network have blocked the innovations and advancement of network technology. The research community is in search of a new architecture that can withstand the challenges of a conventional network.

The research communities have accepted the challenges and come with the solution which is known as "Software Defined Network (SDN)". SDN promises to overcome the limitations of the SDN and improve the Quality of Service (QoS) of the network. SDN decouples the control layer from the data layer and places it in another entity. This feature enables the network administrator to handle the dynamics of the network. It also supports network programmability through northbound APIs, which allows the developer to access the portion of the network. The configuration and management of the network become easier and faster in SDN [1][2]. However, the evolution of the SDN is a long journey started from the 1990s. Feamster et al. [3] very sharply describe the evolution process of the SDN. It starts with active networks [4], where the packet not only carries data but also some executable codes. These codes are executed in end hosts and based on that the paths are decided. The rise of the open signaling also contributes to the SDN evolution. The main enabling factor for SDN is OpenFlow (OF) [5], which allows innovations in networking. It allows the interactions between the control layer and the data layer. Network virtualization and network function virtualization are also considered as supportive factors for SDN evolution [3].

SDN is layered architecture which comprises three layers namely Infrastructure layer, Control layer, and Application layer. The infrastructure layer is the physical structure of the networking devices. The control layer is the brain of the network. The controllers which handle the whole network resides in this layer. The top layer is the application layer which bundles various network performance applications. The end user can interact with this layer by developing the applications. There are different types of interfaces available for communicating between these layers. The southbound API enables communication between the control layer and the infrastructure layer. The standard for this API is OpenFlow. The northbound API allows communication between the application layer and the control layer. There is no standard API defined for northbound API. Different protocols like REST, RESTFul API, etc. are used as northbound API. In a distributed environment, the eastbound and westbound API are used to communicate between multiple controllers [1], [6], [7].

SDN is used in multiple fields like multimedia applications, wireless networks, cyber-physical systems, cloud networks, and data center networks. SDN can be leveraged to achieve optimum performance in the above-mentioned areas. The features of SDN are suitable for the current advance application domains. The needs of the modern network applications and usage area are satisfied by using SDN. However, the actual implementation of SDN faces some challenges. To achieve the maximum from the SDN paradigm, these challenges must be resolved. Even though SDN is a promising architecture, it is still incomplete. There are some research fields like standardization, implementation, and deployment related to SDN which require more attention from academicians and industrialists. The actual realization of SDN will definitely replace the traditional network and be the future of the network if properly implemented.

Following are the contributions of this paper,

- This paper is a reference guide for those new researchers who want to understand various aspects of SDN.
- The study of this survey paper motivates to select specific field of SDN research.
- From the literature review point of view, this paper tries to bridge the gap between existing research and current research trends.

The rest of the paper is structured as follows. The next section covers the history of the SDN related technologies through which it is evolved. Section 3 describes the definition of SDN. Section 4 explains the architecture of the SDN. This section elaborates various layers of SDN architecture such as Application layer, Control layer, and Infrastructure layer with their specific functions. Various communication interfaces are also explained in this section. The basic working methodology of SDN and comparative study of various SDN development tools are presented in Section 5. The following section explains the applications of SDN. The advantages of the SDN are stated in Section 7. The next Section presents the challenges and future research trends of SDN. This section also summarizes the future research directions in brief. At last, the paper is concluded in Section 9.

II. HISTORY OF SDN

Even though the term SDN is highly popular nowadays and is being viewed as the replacement of traditional networking paradigm, the concepts which are imported in SDN are not new. The evolution of SDN has not happened all of a sudden, it is the result of the continuous efforts and advancement of networking technologies. The inline concepts have been researched and implemented for many years. The following subsections review these concepts in brief.

A. Active networks

Feamster et al. [3] explain the history of the programmable network in great details. The research work covers all the aspects of intellectual history through which the SDN is evolved. The first step towards networking advancement was active networking. In the early years of the 1990s to mid years of 1990s, active networking was proposed as a clean-slate approach to the advancement of the network architecture [8].

As per active networking methodology, the packet is transmitted with executable programs in it rather than raw data. At the node end where the packet is received, the program is executed and the action (forward or drop) is performed according to the design of the data plane. This principle of active networking is similar to some intelligent behavior of the networking device which builds an environment which acts based on the contents of the packet instead of just transmitting the data bit by bit from source to destination [4]. The active network creates intelligent network controls in which the packets are the core elements that handle the node behavior. The active network concentrates on providing smart networking environment like end-point PCs which is better than limited capacity dumb switches. It is less focused on the control layer of the networking devices. The key characteristic of the active network which is used in SDN is programmability. The SDN implements on control layer programmability whereas active networking implements data plane programmability [9].

B. Open signaling

After the development efforts of active networking in the 1990s, the community related to open signaling (OPENSIG) was expanded. This approach involves telecommunication and programmability. The open programmable interfaces should be used in the development of the communication hardware in order to access the networking devices. Thus third party software providers get an indirect invitation to enter into the telecommunication software industry market[10].

The interesting utility between OPENSIG and SDN is the differentiation of the various layers in the network. OPENSIG differentiate transport, control and management layer clearly [11]. The key concept is to establish network interface for accessing network devices for network controls and then the new services are introduced by the network service providers by accessing those networking devices to manipulate the state of the network. These enable service providers to implement faster and flexible networking services [9].

C. Data and Control plane separation

At the beginning of the 2000s, the focus of the researchers was diverted towards the reliable, predictable, high performer through the network management functions. The major focus was to gain control over the traffic path of the network. Though the existing approach was working at its best, some researchers had identified the frustration of the network-operators in their inability to control the traffic path. Researcher communities have started working on the separation of the control plane from its data plane [3].

Even though researchers are able to achieve the programmability through the active network, the control over the traffic path was the limiting factor for network operators. Network architecture was so complex and needed to simplify it through the decoupling of data plane and control plane. Merwe et al. [12] propose an ATM Virtualization project which is known as Tempest to make it possible to share the physical resources between multiple virtual networks. These virtual networks are controlled by the software controller. The software controller makes the decisions regarding traffic path.

Internet Engineering Task Force (IETF) has defined the standard to provide an interface in order to make communication possible between the control plane and data plane. The project is known as Forwarding and Control Element Separation (ForCES) [13].

Ethane [14] and SANE [15] propose the control layer access and define the flow based on traffic policies. These projects are the extension to the clean slate project 4D [16] at some level and build the foundations of SDN. The OpenFlow Protocol was initially defined in Ethane which is later adopted by the ONF.

D. OpenFlow (OF)

Midst of the 2000s, network experimentation was at a higher level and researchers were successfully implementing their innovative ideas of network simplification. One such project was Ethane. The initial deployment scenario of OpenFlow (OF) was Stanford campus to actively support ongoing network research. Later OF protocol was accepted widely by many network device manufacturers [3]. OpenFlow (OF) is one of the major enabling factors of SDN which makes the communication possible

between the control plane and data plane. OpenFlow was initially proposed in Ethane and later the term SDN is coined. OF is the realization of SDN and there is a misconception that users start believing that both are same. Nowadays OF becomes the standard API for communication between the control plane and data plane. Controlling of forwarding behavior is possible through OF in SDN [17][18].

E. Network virtualization

Chowdhury and Boutaba [19] conduct a systematic survey of network virtualization and its realization and explain the basic principle of Network Virtualization. The Network virtualization enables network administrators to share the physical resources to implement multiple instances of virtual networks. Both Network Virtualization and SDN are closely connected with each other. By differentiating and forwarding flows to different network slices, SDN can also be used to achieve Network Virtualization.

F. Network Function Virtualization

European Telecommunications Standards Institute (ETSI) defines the standard for virtualization of some network functions such as Firewalls, load balancers, VPN gateways, etc. which may be connected to provide additional services to the network [20]. Using one or more virtual network devices running different software, a virtualized network function can be implemented. As with active networks, NFV focuses primarily on the programmability of the data plane. With this in view, NFV can extend SDN programmability of the data plane because of the control of the current SDN programming of the plane. Furthermore, SDN can be functioned using NFV solutions.

III. SDN DEFINITION

The term Software Defined Network(SDN) is first used in the article [21], where the author defines that data flows can be defined using software and termed this technology as "Software Defined Networking".

The Open Networking Foundation (ONF) is an open, non-profit, community-based corporation which develops, standardizes and commercialize the SDN to transform and revolutionize the carrier industry. The highly appreciated and well-adapted definition given by the ONF is as follows [22]:

"Software-Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable."

According to this definition, the SDN has two major revolutionary network implementations. First, the SDN is an architecture in which the control logic of the networking device is decoupled from the forwarding layer of the networking device. This control logic is installed to an external server. Second, the network is directly programmable through software applications. This principle enables the network administrator to define the desired organizational policies in terms of network behavior through the control layer without implementing additional middleboxes. However, these two characteristics are not newer, but together they provide uniqueness to the SDN.

Kreutz et al. [1] broadly define the SDN as a networking architecture that comprises these four backbones.

A. Control plane separation

The major innovative principle in SDN is the separation of the control plane from the data plane. The control logic is transferred from the network devices to an external entity. The network devices work as simple packet forwarding devices.

B. Flow defined forwarding decisions

The term flow with respect to SDN can be specified as a sequential packet stream between source and destination. The behavior of the networking element is limited to forward the traffic based on the instructions received from the control layer. The decision to forward the packets is flow-based rather than destination based.

C. External SDN controller

The control logic is separated and placed into the commodity server as an external entity. This entity is known as SDN Controller. The controller provides the required resources to process network traffic. It is also responsible to present a global view of the network to other network applications.

D. Programmable network

The fundamental characteristic of the SDN is network programmability. It enables developers to program network behavior through software applications. These applications are running on the top of the control layer and interact with the forwarding layer to establish desired behavior through the southbound interface.

According to the SDN concepts explained in [23], the SDN can be defined as conceptual architecture depends on three fundamental principle abstractions:

1. Forwarding

The forwarding abstraction defines the decisions to forward the traffic are decided by the control logic without revealing the hardware specification on which it runs. OpenFlow is the implementation of the forwarding abstraction.

2. Distribution

This principle centralizes the distributed applications to logical one through unique control layer. This plane has two major responsibilities. First, it decides the forwarding behavior and installs the flow rule into the forwarding devices. Second, it obtains the information regarding the status of the network devices and the interrelated links to present the global topological view to other network applications.

3. Specification

The specification abstraction principle grants various network applications to perform specified network behavior without implementing the behavior themselves. Network programming languages and network function virtualization enable the specification abstraction.

IV. SDN ARCHITECTURE

The SDN architecture is a layered architecture which defines three layers. The layers in the SDN are Application Layer, Control Layer and Infrastructure Layer. The architectural view of SDN is shown in figure 1. As shown in a figure these layers interact with each other through different interfaces. These interfaces are known as southbound interfaces, northbound interfaces, eastbound interfaces and westbound interfaces [1], [2], [6], [9], [22], [24]–[29].

As depicted in figure 1, SDN is a composition of different layers. These layers are designed to perform specific tasks. The following subsection introduces each layer from the bottom-up approach with its defined functions. Here the core concepts and features are explained.

A. Infrastructure Layer

An infrastructure layer of SDN contains a set of various networking elements. These elements are routers, switches, and middlebox appliances. This layer is similar to the infrastructure layer of the traditional network. But the main difference is that the control logic is now removed from the networking elements of this layer and placed somewhere else in the network. The networking devices just act like dumb forwarding devices without any control over forwarding decisions.

The functions of switching devices in the infrastructure layer include[30]:

- The devices are responsible for collecting network status, storing them in local devices and forward them to the controllers. The network status may include network information such as network topology, traffic statistics, and network usages.
- These devices are responsible to forward the packet according to the rule provided by the controllers.

B. Control Layer

In traditional networking, the networks have been managed and configured using the lower level; device-specific instruction set and closed proprietary NOSs. But SDN promised to provide easy network management by solving networking problems using logically centralized control. The control layer is the main layer of SDN architecture which consists of a logically centralized controller. This controller is either located on one server or more than one server. The controller is the brain of the SDN which generate the network specifications based on the network policies defined by the network management. There are various controllers like OpenDayLight, ONOS, Floodlight, Pox, Ryu, etc. which are used by academics and industries to implement SDN functionalities in their environment. Table 1 presented in this paper shows the comparative analysis of some widely used SDN controllers based on specified parameters.

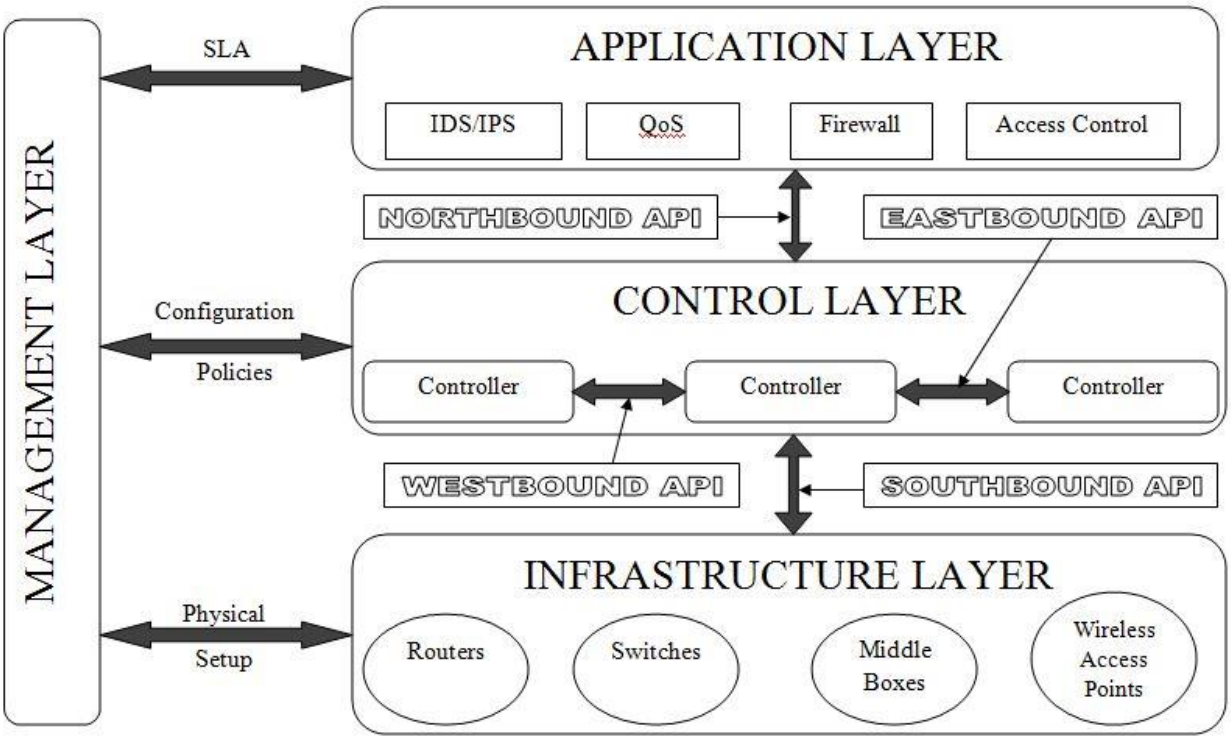

Figure 1 SDN architecture

Controllers are in their simplest form provide device reachability at the layer-2 level and routing information at a layer-3 level to networking devices that require this information to make perfect forwarding decisions.

The core functions provided by the SDN controller are:

- Topology management: The controller handles the global view of the network. It manages all the events of devices leaving and joining the network and regularly updates the information. It provides this information as an abstraction to the applications that run on the top of the control layer.
- Traffic statistics: The controller is responsible to manage all the traffic statistics. It gets the statistics from the OpenFlow devices as and when required and sends it to the applications that manage it.

- Path selection: This task includes the selection of the shortest path from the available many paths to forward the traffic to perform the packet forwarding in a timely manner.
- Security mechanism: The controller is responsible to provide security functions to the network by implementing the access control rules. The network must be secure in order to function well. The controller itself is a single point of failure. So it also ensures the security of itself to provide services to the network.

C. Application Layer

The top layer of SDN is known as the application layer contains various applications defined as per the network policies and user requirements. This layer resides above the control layer. It can easily access the global network abstract view through the controllers. This layer is responsible for implementing various management level policies into the network. These policies are developed by the management of an organization. Various policies such as network access control, firewall, security policies are implemented in the form of applications. These applications interact with the control layer of the network. They lead the controller on how to make decisions about the traffic received. Based on the rules decided in these applications, the controller will install the flow rules in the switches. This layer allows many researchers to experiment with the network according to their requirements. It encourages innovations in networking. The functions of the Application Layer are:

- The Application Layer defines various network related applications based on the policies decided by an organization.
- It allows the network administrators to experiment with the network in order to make it more secure.

In addition to these layers, the SDN architecture also contains various bridges to interconnect these layers. As shown in the figure below, the control layer interacts with the infrastructure layer via southbound interfaces. The application layer communicates with the control layer using northbound interfaces. If the controller is distributed on more than one system, then the interaction between those servers can be made possible using eastbound and westbound interfaces.

These interfaces are the key abstractions of the SDN ecosystem. The conceptual aspects of these interfaces are described below.

1. Southbound Interfaces

The southbound interfaces are the bridges between the control layer and the infrastructure layer. The controller interacts with the data plane/ forwarding devices using southbound APIs. These APIs play an important role in separating control and data plane functionality. OpenFlow is the most widely accepted open southbound interface for SDN. It provides the standards to implement OpenFlow-enabled forwarding devices, and for the communication channel between the data plane and control plane. The detail description regarding OpenFlow is covered in the subsequent section. The OpenFlow is not the only player in the race. There are some other southbound APIs are also available in the market. The proposals for the southbound APIs include ForCES, Open vSwitch Database (OVSDB), POF, OpFlex, OpenState, Revised OpenFlow Library (ROFL), Hardware Abstraction Layer (HAL) and Programmable Abstraction of Datapath (PAD) [31].

2. Northbound Interfaces

The northbound interfaces connect the application layer to the control layer. Northbound interfaces are the software, not the hardware system. There is no standard northbound APIs defined for the SDN. Various controllers use their own northbound APIs. Various network-enabled applications can make use of northbound APIs to request services from the network via the controller. Here is the list of some northbound APIs: RestFul API, SefNet, SDMN API, pyretic, NetKAT, etc. These northbound APIs can be developed by network operators, service providers or researchers. They express the high-level behavior of the network such as advanced path computation, loop avoidance mechanism, etc. to the controller.

3. Eastbound and Westbound Interfaces

Eastbound and westbound APIs are special cases of interfaces required by distributed controllers. When there are multiple controllers implemented in the network, they must communicate with each other for stable network performance. The functions of these interfaces include import/export data between controllers, algorithms for data consistency models, and monitoring/ notification capabilities. Hence these APIs are key components in distributed controllers. So it is essential to have standard east/westbound APIs to provide common compatibility and interoperability between different controllers. SDNi is used as east/westbound API.

4. Management Layer

The management layer interacts with all the layers of SDN architecture. As shown in figure 1, it sets the Service Level Agreements (SLAs) with the Application Layer. These SLAs are used to define the level of access to the network, the trust policies of the third-party applications. Control layer sets the configuration policies defined by the management layer. Various policies are implemented in the SDN controller that handles the network configurations. Physical infrastructure management and its configuration setup are also done by this management layer.

V. WORKING METHODOLOGY OF SDN

SDN is a layered architecture in which the control layer is decoupled from the forwarding layer and placed it on a separate entity. This architecture change brings the difference in the working methodology of the SDN based network from the traditional network. The basic working method of an SDN based network is explained below.

Step 1: Whenever a host connected to OpenFlow switches want to send data to the other host, it first sends the packet to the switch. The switch has no flows installed in it except the default one which instructs it to send any unknown packet to the controller.

Step 2: The controller is the brain of the network which takes the decisions on further processing of the packet. Based on the policy defined, it installs the flows in the networking devices. These flows tell the device to forward the packet, to drop the packet. Based on the global view of the network, the device is instructed to forward the legitimate traffic to the specific device.

Step 3: The above step is performed in the network until the destination is not found and the packet is delivered to the destination host.

SDN network can be created in a virtual environment also. The virtual environment helps us to understand the basic functionality of SDN. There are advantages of the virtual environment like flexibility, security, apart from the production environment. In this paper, some famous emulators/simulators are analyzed which support the OpenFlow Network. The table 2 below shows the brief summary of these network emulators.

VI. APPLICATIONS OF SDN

The central controller based architecture of SDN facilitates the compilation of network information and makes it easier to adapt and manage network changes in dynamic networking. Therefore SDN technologies can be used to achieve different goals in various areas.

A. Cloud and Data-center networks

The cloud and the data centers are key areas where SDN implementation has created the differences. One of the major key points of the cloud network is that the user can obtain enough resources as per the requirements. So cloud management is the most concerned area. SDN is the expected solution to make the cloud and data center configuration simpler and flexible.

Baker et.al [32] proposed the SDN based cloud data center implementation using OpenFlow. Because of the centralized control plane, it is possible to configure the cloud data centers easier and faster.

Jain et al. [33] in B4 presents the benefits of the OpenFlow enabled network to connect Google's data centers. The ample bandwidth requirements, average bandwidth utilization, and control over rate limitation and measurement can be satisfied using OpenFlow enabled SDN. The results show that Google has achieved nearly 100% link utilization. Google briefly presents the benefits of SDN: First simple configuration, management, and optimization of the network. Second, optimal cyber resources utilization can be achieved. Third, network behavior can be predicted. Hence major industry leaders are now applying the SDN architecture to their data centers.

B. Network virtualization

Network virtualization is one of the hot topics of modern networks which allow users to share physical limited resources to obtain optimal resource utilization among multiple virtual networks. Because of the central controller, SDN is considered as the solution to the problems of the network virtualization.

J. Matias et.al [34] proposed a solution called the EHU OF Enabled Facility (EHU-OEF), to handle multiple virtual networks and sharing physical resources using OF and layer 2. Layer 2 prefix based network virtualization is selected to implement the EHU-OEF. Easier configuration management and flexibility support for changing header fields enable users to separate each virtual machine. Thus network virtualization management becomes easy and reliable using SDN.

Table 2. Summary of SDN Development Tools

Tool	Language	Feature	OF Support	Platform Supported	Open Source
Mininet [35]	Python	The emulator that supports multiple virtual OF switches, end hosts and controllers on single PC, Real-time production environment support.	1.4	Ubuntu	Yes
NS-3 [36]	C++	Network simulator for discrete event-based network	1.3	Linux	Yes
OMNeT++ [37]	C++	Modular, component based network simulator	1.0	Linux, MAC, Windows	No
EstiNet[38]	Ruby & C	Commercial network emulator.	1.0	Linux	No
Trema [39]	Ruby	High level OF library supported network emulator to create OF network	1.3	Linux	Yes
Mirage[40]	Ocaml	Multiplatform OpenFlow supported network emulator	1.3	Linux, MAC	No
GENI [41]	Multilingual	GENI is open source project testbed which provides a virtual laboratory to develop large scale network	1.3	Linux	No

C. Security

Security is the key concern of network management in order to provide uninterrupted services. SDN provides a robust approach to detect security vulnerabilities in the network. The central controller and its global network view enable to detect security attacks at an early stage and prevent more damages to the network. SDN can be leveraged to prevent DoS, DDoS attacks, ARP Spoofing attacks, LAN attack, etc. Xiong Liu et.al in [42] proposed a security-based policy to implement the security solutions for the network.

D. Network management

The central logical controller makes it easier to implement various network policies and simplify network management. From the network administration point of view, SDN architecture backs up the flexibility and scalability of the network. Kim and

Feamster [43] specifies the problems of network management and propose the framework Procera, to make network management easier and flexible using SDN.

E. Wireless and Mobile

Luo and Quek [44] demonstrated that the SDN can also be applied to the Wireless Sensor Network(WSN). WSN with SDN leverages the advantages of SDN like flexibility, maximum resource utilization, simple network management, etc. SDN based wireless network helps to design various network policies and empowers the network controller to implement those policies effectively.

F. ICN based on SDN

Many researchers have declared in past years that modern internet architectural design cannot stand up to users ' evolving and future business needs. New architectures were implemented on the grounds of this claim. One of these architectures is the information-centered network. In ICN, a unique name of the information, independent of locations, applications, storage, and primitive distribution, is used. Multiple transmission techniques are implemented to fetch named information, like name-based routing, name-based resolution, and so on [45],[46]. To achieve the advantages of ICN, currently implemented network devices require radical changes, which raise difficulties to implement ICN. Incorporating ICN and SDN can motivate technological innovations to grow effectively and meet the challenges of today's networks with ease. Backing mobility and cloud computing in today's networks are two widely debated subjects due to IP protocol inconsistency. Based on the network view of controllers, SDN can be utilized to decide the VM migration policy. It can also be used to configure the network devices automatically after VM migration [47].

G. Multimedia and QoS

The modern architecture of the modern internet is based on peer-to-peer communication control which enables the best service efforts for data transmission. But for multimedia services, the approach is not suitable. Multimedia services like video streaming, video conferencing, on-demand video, Web TV, etc. necessitate resilient network resources and withstand particular delay amount, jitter and error-rate. To achieve these QOS, optimal path among all the available network paths need to be selected [48][49].

According to the requirements of flow statistics and global network view, it is possible to choose different paths for different traffic requirements using SDN. Moreover, the OpenFlow protocol itself provides some QoS features like queuing, transmission rate control, etc. Therefore multimedia QOS leverages the SDN.

From the above literature review, it is observed that SDN is a versatile network architecture. It can be applied to a wide area of networking applications. Many existing applications have limited benefits which can be maximized by integrating SDN with them. Table 3 presents a summary of the applications of the SDN.

VII. ADVANTAGES OF SDN

SDN is an emerging architecture of the future internet. Many major industries have adopted the SDN architecture to meet the global requirement of flexible, scalable, robust and resilient network. SDN offers following benefits of the logically centralized controller over traditional networking [1], [7], [28], [47], [50]–[52].

- SDN has reduced complexity by removing the control plane from network device.
- SDN is less error-prone when modifying network policies through high-level applications.
- The automatic reaction of the controller to the drastic changes occurred in the network helps to overcome the issues created due to the network changes.
- SDN has a global view of the network which helps to develop more customized applications, services, and network functions.
- SDN provides optimal resource utilization.
- SDN empowers high-speed data transmission through a centralized controller.
- SDN supports network programmability.
- Network management process becomes easier in SDN.
- SDN support multi-tenancy in a cloud-based network.
- Virtualization of network resources can be easily managed using SDN.
- SDN solutions reduce the overall costs incurred due to network management.
- For security centric network management, SDN provides robust and resilient solutions.

Table 3 Summary of Applications of SDN

Application Domain	Description
Network Virtualization	SDN can help to overcome the limitations of network virtualization, allow easier management of resource utilization through a central controller.
Security	Central controller and separation of the control plane from the data plane overcome the security problems of traditional networking.

Network Management	It is more convenient to install various network policies and configuration through centralized control.
Wireless Network	SDN can help to improve the performance of wireless networks. It provides flexibility, optimum resource utilization, simple network management.
ICN based on SDN	Many problems of ICN like name-based routing, name based resolution can be solved by integrating ICN and SDN.
Multimedia and QoS	Modern digitalization demands performance and QoS. Multimedia applications can be served with higher performance using SDN management.
Cloud and Data Center Networks	Cloud networks and data centers are the key areas for which the transformation of SDN was initiated. SDN helps to manage large and dynamic data centers and cloud networks easier and faster.

VIII. Challenges and Future Research Direction Of SDN

Even though the SDN promises to overcome the limitations of traditional networking and provides better and flexible solutions, SDN itself is not complete. There are certain areas which require the attention of the researchers in order to bridge the gap of the current SDN and optimal SDN. The qualitative and quantitative approaches are required to evaluate the efficiency and performance of the SDN. Hence SDN poses some challenges to the researchers that gain the attraction of the researchers [survey paper citations]. In this paper, some of those important challenges and research areas have been identified and presented briefly in the following section.

A. The programmability of data plane

Farhady et al. [9] state that the research community has not paid enough attention to the programmability of the data plane while it is technically feasible. The initial SDN architecture focused mainly on the control layer programmability. However, data and control plane programmability both requires equal attention to empower network services and applications. Data plane programmability enables intelligent behavior of network devices. This eventually helps in reducing the load of the controller. It also enhances the security of the network.

B. User-driven control of Network

Network administration can be made simpler using SDN APIs. These APIs can also be used by end-users to achieve on-demand services. The examples are intrusion detection system [53] at the end user machine, bandwidth on demand requester to improve the performance of the network [54]. To achieve the on-demand services from end users, API should exist between the end user application and control plane of the network. However, there are some issues in adapting the user-defined APIs to achieve user-defined control. The very first challenge is to maintain the trust between the network and the user-defined API. Security is the second issue which must be considered while handling the portion of the network to the users. So the above said challenges should be looked after to achieve the user-driven control and this could be potential research direction.

C. Platform independence

Innovations and productions should not depend on specific hardware or software platforms. Current SDN is still in its early stage and OF switches are dependent on specific hardware and software. This restricts the openness of the SDN. Future requirements of SDN demand the true flavor of open SDN. Hence it will be a good research direction in which SDN can be transformed into platform-independent architecture where commodity hardware can be utilized to design and develop independent SDN data plane.

D. Standardization of Controller and APIs

Even though the OpenFlow is widely accepted standard for the southbound API and the de facto realization of the SDN, it is not the only player in the race. IETF has also published the SDN framework [55]. As per the literature [20], NFV is also promoted as a strong contender for SDN by ETSI Industry Specification Group. A comprehensive study of all the SDN implementations should be considered to make standard implementation. Control plane also lacks implementation standards. Many controllers with similar approaches are available but none of them are complete.

E. Implementation of Control Plane

SDN separates the control plane from the data plane in the networking devices. At the initial stage, it looks ideal to replace the control plane and place it to the external entity. But it also removes the inbuilt routing protocols from the forwarding devices. People may hesitate to adopt the idealistic SDN approach over traditional networking approach because the replacement of the conventional network devices may result in chaos. Some issues still exist in OpenFlow design. OpenFlow is still in its transition phase. Newer versions of OpenFlow protocols are evolved continuously and backward compatibility should be maintained [56].

F. Deployment of SDN in network

To adopt and deploy the SDN at large scale, additional research is required by researchers. There are many areas like wireless sensor network [57], wireless mesh network [58], cyber-physical system [59], carrier networks[60] in which SDN can be integrated to leverage the benefits of SDN. These areas are potential research areas that can explore the new heights of technology in networking.

G. Smart flow rule management

SDN motivation is to remove the controller function from the networking switches and make those switches only to forward the traffic based on the flow installed in the flow table. The idea is to make the network processing function simpler and easier. The networking devices simply send traffic to the upper layer when they do not find any related flow in the flow table. The controller will decide what operation should be performed on a packet or will send some status messages back. The current OF specification does not specify how to manage those data. Moreover, when the network traffic is too heavy and the flow tables have complicated flow rules installed, it is advisable that the switches should perform the operations based on some intellectual self-learning. So controller and network switches may implement some intelligent flow rule management. This could be a potential research direction in the SDN [7].

H. Controller scalability

Many large networks have adopted SDN, but the network controller installed in a single entity may be the choke point for the network in heavy load situations. In large data centers, the network is not limited to a small region but expanded in large areas. Both scenarios need logical central controller should be installed in multiple servers. The distributed servers may join the race to acquire the computing resources. To handle the coordination between multiple controllers needs attention. Moreover, optimal resource utilization strategy should be implemented in order to share the resources among the controllers [7].

I. Language flexibility

To successfully achieve the objectives of the SDN, SDN programming language is capable of handling dynamic flow rules. An event change occurs in the network is also adapted by SDN programming language in order to change the defined policy according to the network change. A good programming language is required to implement policy changes in networking devices [61].

J. QoS implementation

Quality of Services is necessary for better network performance. Many applications like multimedia services, load balancing services require benchmark QoS to implement them satisfied level. Multimedia services use video encoding standards which can be utilized to assign priorities to different layers of video data. Policy-based flow rule controlling could be integrated into OpenFlow to support such multimedia services. Additional integration of flow rules with various encoding standards needs active research too [62].

In order to distribute the heavy load of the network among multiple network devices, optimal load balancing capabilities are necessary for the Content Distribution Network. OpenFlow makes it possible to perform load balancing in each of the flow paths. The research is required to standardize the steps in OF controller to coordinate the path and load balancer selection.

K. Robust wireless integration

Even though OpenFlow has been successfully coordinated with the wired local area network, very fewer studies have been conducted its integration with wireless networks. Even with these existing studies, OpenFlow integration with wireless networks suffers from two issues. First, the management between channel access and data forwarding is not standardized. Second, scheduling algorithms are also required to manage collision free control and data traffic in wireless links. Therefore, wireless networks need additional research in order to integrate the SDN with it [44], [57], [58].

L. Security of SDN

SDN initially designed to prevent security attacks and provide robust security to the network. But it has also opened doors for attackers by introducing new SDN specific attack vectors. This attack vector comprises attacks on the control plane, OF switches, links between OF switches and control plane, un-trusted third-party applications, etc. Moreover, SDN is in its childhood phase, therefore there are no robust solutions exist for securing the SDN based network from the attackers who target these new vulnerabilities. Many researchers have tried to provide some of the solutions of possible attacks in SDN but satisfactory work has not been proposed yet. Therefore the security researchers can dive deep into this field and can come up with new ideas for the resilient network [63]–[69].

Though SDN is an emerging and growing concept of the networking world, it also suffers from its own limitations and challenges. These challenges and limitations cannot be ignored in order to achieve the true benefits of the SDN. To overcome these challenges the researchers should pay attention to find out solutions. There are some additional research directions are also discussed in this paper. Table 4 presents the summary of research trends of SDN and its brief description.

Table 4 Summary of Research Trends in SDN

Research Trends	Description
The programmability of data plane	The data plane programmability introduces the intelligent behavior of the network and reduces the load of the controller. Thus it improves the performance of the network.
Platform independence	To achieve true open source flavor of SDN, it should be researched to make it platform independent.
User-driven control of Network	Network management becomes easier with user-defined APIs. These APIs can have multiple issues of security and integration. The research should be carried out to design a standard framework for API control.

Standardization of Controller and APIs	Many alternatives for controllers and interfaces are available in the market. The standard should be defined for those controllers and APIs to make SDN architecture more robust.
Implementation of Control Plane	Separation of control plane needs drastic changes in the existing network. It requires more careful investigation before implementing the SDN in an existing network.
Deployment of SDN in network	SDN can be integrated with the wireless sensor network, wireless mesh network, cyber-physical systems, etc to empower the functionality of this network.
Smart flow rule management	Intelligent behavior can be achieved in the SDN infrastructure layer which will help in improving the performance and QoS of the network.
Controller scalability	To accommodate and manage large scale network, scalability of the controller is also a research direction.
Language flexibility	SDN programming languages should be able to handle huge dynamic flow rules. The standard for SDN programming languages should also be developed.
QoS implementation	QoS is always a critical concern for any network, SDN is no exception. QoS implementation is complex for a dynamic network like SDN. It requires additional investigation to make it compatible with the dynamic network.
Robust wireless integration	SDN is successfully integrated with wired network and proves the benefits, but it needs some more attention while integrating with the wireless network. It is not as simple as a wired network.
Security of SDN	SDN introduces more security-related threats. These threats are specific to SDN. So it requires new approaches to handle those attacks.

IX. CONCLUSION

SDN is gaining popularity amongst the network administrator and researchers because of its benefits over the conventional network. It has an interesting set of features that allows innovations in managing and handling the networks. Some of the outstanding features of the SDN includes dynamic programmability, security, separation of data and control plane. However, SDN is in its infancy period. It has critical challenges which need to be resolved before actually implementing it.

As a part of our SDN journey in this review paper, we first elaborate the technologies like active network, open signaling, separation of data plane and control plane, OpenFlow, network virtualization, and network function virtualization, which eventually helped in SDN architecture evolution. The roots have been analyzed from where the SDN has emerged. Next, the definition and architecture of the SDN are explained with great details. Therein, the components of SDN like infrastructure layer, control layer, application layer, northbound API, southbound API, and east-west bound APIs are described along with their functionalities. The comparative analysis of various SDN controllers and SDN development tools are also presented in this review paper. Afterward, the advantages of SDN are presented. Various use cases and applications of SDN like multimedia services, data center and cloud network services, wireless network, services are discussed. At the end of this paper, we have tried to analyze the current and future trends of the SDN networks. From the literature review, it is concluded that the SDN can be leveraged if properly maintained and integrated with advanced networking technologies. These could be good research directions for potential researchers. The research trends such as wireless networks, ICN based on SDN, network virtualization, mobile networks and security field of SDN are properly analyzed. If properly researched and developed, SDN will certainly replace the traditional network.

Table 1 Comparative Analysis of SDN Controllers

Controller	Architecture	Modularity	Programming Language	Developed by	Open Source	OF Version	Multi-threaded	GUI support	Northbound API	Southbound API	Platform Support	Open Stack Support	Developed Year	Application Domain	Features
Pox [70]	Centralized	Low	Python	Nicira	Yes	1.0	No	Python QT4	ad-hoc API	OF	Linux, Mac, Windows	No	2013	Campus WAN	Open source, general purpose controller licensed with Apache Public License
Nox[71]	Centralized	Low	Python, C++	Nicira	Yes	1.3	No	Python QT4	ad-hoc API	OF	Linux	No	2008	Campus WAN	First OpenFlow controller with General Public License (GPL)
Onos[72]	Distributed	High	Java	Linux Foundation	Yes	1.5	Yes	Web Based	RESTFul API	OF, NETCONF	Linux, Mac, Windows	No	2014	WAN, Transportation, Datacenters	Scalable, high performance, resilient open source network OS
Maestro [73]	Centralized	Average	Java	Rice University	Yes	1.0	Yes	Console	ad-hoc API	OF	Linux, Mac, Windows	No	2010	Research	Open source modular network controller supports multicore processes and parallelism.
OpenDay Light [74]	Distributed	High	Java	Linux Foundation	Yes	1.3	Yes	Web Based	REST, RESTCONF	OF, NETCONF, YANG	Linux, Mac, Windows	Yes	2013	Datacenters	Eclipse Public License based network OS widely used for commercial applications
Ryu [75]	Centralized	Average	Python	NTT Labs	Yes	1.5	Yes	Console	ad-hoc API	OF, NETCONF	Linux	Yes	2011	Campus WAN	Component-based SDN framework, logically centralized, opensource controller
Floodlight [76]	Centralized	Average	Java	Big Switch Networks	Yes	1.3	Yes	Web Based	RESTFul API	OF	Linux, Mac, Windows	No	2012	Campus WAN, Research	Open source controller which supports OF and non-OF network handles multiple virtual and physical switches

REFERENCES

[1] D. Kreutz, F. M. V Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," *Secur. Commun. networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[3] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, 2014.

[4] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 2, pp. 5–17, 1996.

[5] N. Mckeown, T. Anderson, L. Peterson, J. Rexford, S. Shenker, and S. Louis, "OpenFlow: Enabling Innovation in Campus Networks," vol. 38, no. 2, pp. 69–74, 2008.

[6] Open Networking Foundation, "SDN Architecture Overview," *Onf*, no. 1, pp. 1–5, 2013.

[7] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[8] K. Calvert, "Reflections on network architecture: an active networking perspective," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 27–30, 2006.

[9] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Comput. Networks*, vol. 81, pp. 79–95, 2015.

[10] A. T. Campbell, I. Katzela, K. Miki, and J. Vicente, "Open signaling for ATM, internet and mobile networks (OPENSIG'98)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 1, pp. 97–108, 1999.

[11] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, 1999.

[12] J. E. der Merwe, S. Rooney, L. Leslie, and S. Crosby, "The Tempest-a practical framework for network programmability," *IEEE Netw.*, vol. 12, no. 3, pp. 20–28, 1998.

[13] A. Doria *et al.*, "Forwarding and control element separation (ForCES) protocol specification," 2010.

[14] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, 2007, vol. 37, no. 4, pp. 1–12.

[15] M. Casado *et al.*, "SANE: A Protection Architecture for Enterprise Networks.," in *USENIX Security Symposium*, 2006, vol. 49, p. 50.

[16] A. Greenberg *et al.*, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, 2005.

[17] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[18] A. Nygren *et al.*, "Openflow switch specification version 1.5. 1," *Open Netw. Found. Tech. Rep.*, 2015.

[19] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[20] M. Chiosi *et al.*, "Network functions virtualisation introductory white paper," in *SDN and OpenFlow World Congress*, 2012.

[21] K. Greene, "TR10: Software-Defined Networking." 2009.

[22] O.N.F., "Software-defined networking: The new norm for networks," *ONF White Pap.*, vol. 2, pp. 2–6, 2012.

[23] S. Shenker, M. Casado, T. Koponen, N. McKeown, and others, "The future of networking, and the past of protocols," *Open Netw. Summit*, vol. 20, pp. 1–30, 2011.

[24] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *J. Netw. Comput. Appl.*, vol. 67, pp. 1–25, 2016.

[25] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on software-defined networking," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[26] B. N. Astuto *et al.*, "A Survey of Software-Defined Networking : Past , Present , and Future of Programmable Networks," 2014.

[27] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in Software Defined Networks: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[28] C.-S. Li and W. Liao, "Software defined networks," *IEEE Commun. Mag.*, vol. 51, no. 2, p. 113, 2013.

[29] S. Singh and R. K. Jha, "A survey on software defined networking: architecture for next generation network," *J. Netw. Syst. Manag.*, vol. 25, no. 2, pp. 321–374, 2017.

[30] "Pox," *URL https//github.com/noxrepo/pox*.

[31] "Floodlight," *Available: http://www.projectfloodlight.org*.

[32] C. Baker, A. Anjum, R. Hill, N. Bessis, and S. L. Kiani, "Improving cloud datacentre scalability, agility and performance using OpenFlow," in *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems*, 2012, pp. 20–27.

[33] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *ACM SIGCOMM Computer Communication Review*, 2013, vol. 43, no. 4, pp. 3–14.

[34] J. Matias, B. Tornero, A. Mendiola, E. Jacob, and N. Toledo, "Implementing layer 2 network virtualization using OpenFlow: Challenges and solutions," in *2012 European Workshop on Software Defined Networking*, 2012, pp. 30–35.

[35] "Mininet," 2019. [Online]. Available: http://mininet.org/overview/.

[36] "NS-3," 2019. [Online]. Available: https://www.nsnam.org/.

[37] "OMNet++," 2019. [Online]. Available: https://omnetpp.org/.

[38] "Estinet," 2019. [Online]. Available: http://www.estinet.com/ns/.

[39] "Trema," 2019. [Online]. Available: https://trema.github.io/trema/.

[40] "Mirage," 2019. [Online]. Available: https://mirage.io/.

[41] "GENI," 2019. [Online]. Available: https://www.geni.net/.

[42] X. Liu, H. Xue, X. Feng, and Y. Dai, "Design of the multi-level security network switch system which restricts covert channel," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, 2011, pp. 233–237.

[43] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, 2013.

[44] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, 2012.

[45] X. N. Nguyen, D. Saucez, and T. Turletti, "Providing CCN functionalities over OpenFlow switches," 2013.

[46] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, and L. Veltri, "Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed," *Comput. Networks*, vol. 57, no. 16, pp. 3207–3221, 2013.

[47] F. Hu, *Network innovation through OpenFlow and SDN: principles and design*. CRC Press, 2014.

[48] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, "A qos-enabled openflow environment for scalable video streaming," in *2010 IEEE Globecom Workshops*, 2010, pp. 351–356.

[49] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, 2012, pp. 1–8.

[50] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: a retrospective on evolving SDN," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 85–90.

[51] Y. Kanaumi, S. Saito, and E. Kawai, "Toward large-scale programmable networks: Lessons learned through the operation and management of a wide-area openflow-based network," in *2010 International Conference on Network and Service Management*, 2010, pp. 330–333.

[52] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, p. 19.

[53] J. Ibrahim and S. Gajin, "SDN-based intrusion detection system," *Infoteh Jahorina*, vol. 16, pp. 621–624, 2017.

[54] P. Qin, B. Dai, B. Huang, and G. Xu, "Bandwidth-aware scheduling with sdn in hadoop: A new trend for big data," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2337–2344, 2017.

[55] T. Nadeau and P. Pan, "Framework for software defined networks," *Framework*, 2011.

[56] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[57] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks (sdwn): Unbridling sdns," in *European workshop on software defined networking*, 2012, pp. 1–6.

[58] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE Netw.*, vol. 29, no. 4, pp. 24–30, 2015.

[59] E. Molina and E. Jacob, "Software-defined networking in cyber-physical systems: A survey," *Comput. Electr. Eng.*, vol. 66, pp. 407–419, 2018.

[60] L. Velasco, A. Asensio, Jl. Berral, A. Castro, and V. López, "Towards a carrier SDN: An example for elastic inter-datacenter connectivity," *Opt. Express*, vol. 22, no. 1, pp. 55–61, 2014.

[61] N. Foster *et al.*, "Languages for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 128–134, 2013.

[62] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, 2017.

[63] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, 2013.

[64] H. Zhang, Z. Cai, Q. Liu, Q. Xiao, Y. Li, and C. F. Cheang, "A survey on security-aware measurement in SDN," *Secur. Commun. Networks*, vol. 2018, 2018.

[65] L. Schehlmann, S. Abt, and H. Baier, "Blessing or curse? Revisiting security aspects of Software-Defined Networking," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014, pp. 382–387.

[66] Z. Hu, M. Wang, X. Yan, Y. Yin, and Z. Luo, "A comprehensive security architecture for SDN," in *2015 18th International Conference on Intelligence in Next Generation Networks*, 2015, pp. 30–37.

[67] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with SDN: A feasibility study," *Comput. Networks*, vol. 85, pp. 19–35, 2015.

[68] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *2013 IEEE SDN For Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.

[69] A. Demirpolat, D. Ergenç, E. Ozturk, Y. Ayar, and E. Onur, "Software-Defined Network Security," in *Enabling Technologies and Architectures for Next-Generation Networking Capabilities*, IGI Global, 2019, pp. 232–253.

[70] "Pox," 2019. [Online]. Available: https://github.com/noxrepo/pox.

[71] "Nox," 2019. [Online]. Available: https://github.com/noxrepo/nox.

[72] "ONOS," 2019. [Online]. Available: https://onosproject.org/.

[73] "Maestro," 2019. [Online]. Available: https://code.google.com/archive/p/maestro-platform/downloads.

[74] "OpenDayLight," 2019. [Online]. Available: https://www.opendaylight.org/.

[75] "Ryu," 2019. [Online]. Available: https://osrg.github.io/ryu/.

[76] "Floodlight," 2019. [Online]. Available: https://github.com/floodlight/floodlight.