National Conference on Recent Advances in Computer Science and IT (NCRACIT)
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 4 | Issue 1 | ISSN: 2456-3307

# Software-Defined Networking: A Survey

**Sakshini Hangloo[1], Sudesh Kumar[2]**

[1]M.Tech, Dept. Of computer science, Shri Mata Vaishno Devi University, J&K, India
[2]PhD scholar, Dept. Of computer science, Shri Mata Vaishno Devi University, J&K, India
sakshini.hangloo@gmail.com[1]

## ABSTRACT

Emerging trends in information technology like cloud computing, mobile computing, big data etc are posing new challenges to future Internet, as it requires higher accessibility, high bandwidth, and dynamic management. On the other hand, traditional approaches cannot fully utilize the capability of physical network infrastructure. Software-defined networking (SDN) is one of the most promising solutions for future Internet. SDN is characterized by its two distinguished features (1) decoupling the control plane from the data plane, and (2) providing programmability for network application development. As a result, SDN is positioned to provide better performance, and higher flexibility to accommodate innovative network designs. This paper surveys the programmable networks with main focus on SDN. The SDN architecture and the OpenFlow standard are discussed.

**Keywords:** Software-defined networking, SDN, network virtualization, OpenFlow, programmable Networks, data plane, control plane

## I. INTRODUCTION

The today's network is growing very rapidly with a large number of users, applications, sensors adding the volume to it but the current technology is becoming insufficient to cater the huge traffic generated by them. Existing static networks are ill-suited for the dynamic needs of today's and future's environment. And hence, there is an increasing need of new networking infrastructure that will give high performance, and are energy efficiency and reliable. Meeting such requirements with the existing network devices is impossible as their capabilities are limited. Additionally, to implement network-wide policies and to support any new service, the network administrators have to configure thousands of network devices and protocols making it almost impossible to apply a consistent set of security, and other policies. And as an overhead, these devices have their control and forwarding planes coupled together and the internals differ from vendor to vendor.

A lack of open standard limits the ability of network operators to modify the networks as per requirement of individual environments. Hence, there is a need for an architecture that decouples the forwarding and control planes of the network devices to dynamically link forwarding and control elements.

## II. SDN ARCHITECTURE

Software-Defined Networking (SDN) is a concept that has the potential to change the networks and

the way they are designed, build, and operated. SDN [3,4] has emerged as the network architecture where the control plane is decoupled from the forwarding plane enabling the network control to become directly programmable[7]. Its concept was initially proposed by Nicira Networks based on their earlier development at UCB, Stanford, CMU, Princeton [12]. The present network architecture has many limitations which can be resolved with the SDN architecture, such as inability to optimize network for WAN and Data Centre, to generate more revenue and reduce expenses[1]. SDN can control and manage network's behaviour dynamically through software via open interfaces. It is different from the traditional networks in the way that it does not rely on proprietary defined interfaces.

The OpenFlow architecture typically includes the following 3 important components [7],[11],[26].

1) Switches: OpenFlow defines an open source protocol to monitor and change the flow tables in switches/routers.

An OpenFlow switch has major three components: the flow tables, communication channels and the OpenFlow protocols, a) The flow tables consists of an action field associated with respective flow entry, b) the communication channel provides a link for the transmission of commands and packets between controller and switches, c) the OpenFlow protocol enables the controller to communicate with any router or switch.

2) Flow-entries: Each flow-entry includes an action for that flow item. The OpenFlow switches support the following actions: (a) sending the packets to the respective ports, (b) encapsulating the packets and sending to a controller, and (c) dropping the packets.

3) Controllers: A controller can update, add, or delete flow entries from the flow table on behalf of the user's testing. A static controller is usually a simple software unit running on a system to statically establish a path between a group of network devices during a scientific experiment.

The SDN architecture consists of three major parts: application, control plane, and data plane (Fig. 1). The application label uses the decoupled nature of SDN to achieve specific goals, such as a security mechanism [9], a network measurement solution [10] etc. The Applications communicate with a SDN controller at the control plane via the northbound interface to enforce their policies in the data plane without directly interacting with the data plane. The interface between the control and data plane is supported by southbound APIs, there the SDN controller uses these APIs to communicate with the network devices in the data plane. The control plane manipulates forwarding devices using the SDN controller to achieve the specific goal of the target application. The controller uses the southbound interface to connect to the data plane. The data plane handles the actual packets based on the configurations that are manipulated by the controller.

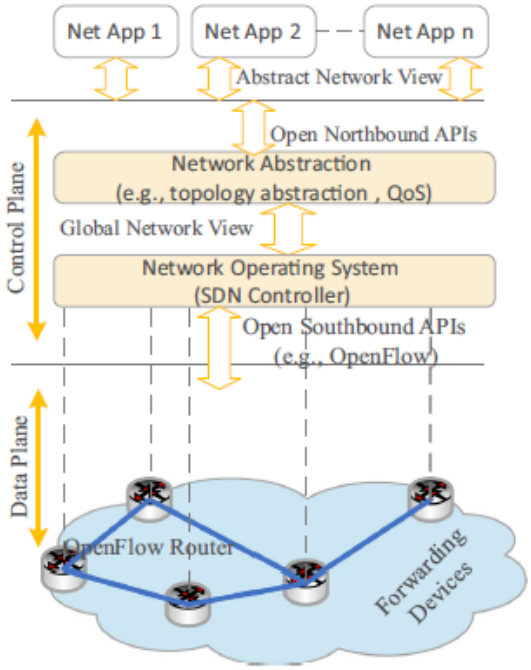


**Figure 1.** SDN architecture

The SDN architecture is:

**1) Directly programmable:** Network control is directly programmable because it is decoupled from forwarding functions. Also, the programs now do not depend on proprietary software so network managers manage, control, configure, secure, and optimize network resources quickly through dynamic SDN programs.

**2) Agile:** The network administrators can dynamically adjust network-wide traffic flow to meet the changing needs

**3) Managed centrally:** The control plane (network intelligence) is logically centralized in software-based SDN controllers that maintain a global view of network. These controllers appear to applications as a single switch. Consequently, the devices themselves no longer need to understand and process thousands of protocol standards but merely accept instructions from the SDN controllers.

**4) Open standards-based vendor-neutral:** When implemented using open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers rather than vendor-specific protocols. A concrete realization of the SDN approach is OpenFlow (OF) [5,6].

## III. OPENFLOW STANDARDS

There are a number of protocol standards that exist on the use of SDN in real applications. One of the most popular protocol standards is the OpenFlow protocol. OpenFlow is a protocol that helps in the implementation of the SDN concept in both hardware and software. One of the most important features of OpenFlow is that researchers can utilize the existing hardware to design new protocols and analyze their performance. Now it is becoming part of commercially available routers and switches as well.

OpenFlow was proposed by Stanford as a standard SDN protocol. Regarding testbeds of OpenFlow, many designs have been proposed for OpenFlow protocols. They use open source codes to control the SDN controllers and switches. OpenVSwitch (OVS) [33] is one of the most popular software-driven OpenFlow switch. Its kernel is written in Linux 3.3 and its firmware including Pica8 [34] and Indigo [35] is also available.

OpenFlow is flow-oriented protocol. In SDN, there is a controller that manages the switches for traffic control. The controller communicates with the OpenFlow switch and manages the switch through the OpenFlow protocol. An OpenFlow switch can have multiple flow tables, and an OpenFlow channel. Each flow table contains flow entries and communicates with the controller. The group table can configure the flow entries. OpenFlow switches connect to each other via the OpenFlow ports.

Initially the data flow path of the OpenFlow switch has an empty routing table with some fields such as source IP address, destination IP address, MAC address, QoS type, etc.. This table contains several packet fields such as an action field which contains the code for different network operations, such as packet forwarding, dropping or reception, etc. This table can be populated based on the incoming data packets. When a packet from a new flow is received which has no matching entry in the data flow table, it is forwarded to the controller to be processed.

The controller takes the packet handling decisions, for example, whether a packet is to be dropped, or a new entry is to be added into the data flow table on how to deal with this and similar packets received in the future.

SDN has the capability to program multiple switches simultaneously; but still suffers from conventional complexities such as dropping packets, delaying of the control packets etc. Current platforms for SDN such as NOX and Beacon enable programming, but it is still hard to program them in a low level languages. With OpenFlow protocols becoming more standard in industry, SDN is becoming easier to implement. The control plane generates the routing table and the data plane utilizes those table to determine where the packets should be routed [32]. OpenFlow and SDN allow data centers and researchers to easily abstract and manage the large and complex networks.

## IV. FUNDAMENTAL CHARACTERISTICS OF SDN

Software Defined Networking is characterized by five fundamental traits: plane separation, a simplified device, centralized control, openness, and network virtualization [21]

### A. Plane Separation

The very key characteristic of SDN is the decoupling of the forwarding plane and the control plane. The Forwarding plane contains the forwarding tables and the logic for dealing with incoming packets based on MAC address and IP address.

The forwarding plane manages the arriving packets by forwarding, dropping, consuming or replicating it. For forwarding, the device determines the correct output port by looking up in the address table. A packet may also be dropped due to buffer overflow. Some packets that require processing by the control plane are consumed and passed to the appropriate plane. Finally, in the case of multicasting the incoming packet must be replicated before dispensing the copies from various output ports. The main logic that is used to control the forwarding plane resides in the control plane.

The control plane determines how the forwarding tables in the data plane be configured. In a traditional network each device has its own control plane who's primary task of is to run routing protocols so that all the distributed forwarding tables on the devices throughout the network stay synchronized. This synchronization is needed to prevent loops. In SDN, the control planes of all the switching devices are moved onto a centralized controller.

### B. Simplified Device and Centralized Controller

Keeping in mind the idea of separation of forwarding and control planes, the other characteristic is the simplification of devices. Here in SDN, instead of running thousand lines of code of complicated control plane software, that software is removed from the device and placed in a centralized controller. The device is allowed to behave autonomously by a centralized system on which management and control software run.

The controller provides the instructions to these simplified devices, when needed, in order to allow them make faster decisions about how to deal with the arriving packets.

### C. Openness

A basic characteristic of Open SDN is that its interface should remain well documented, standard, and not proprietary. Individuals can take advantage of this capability in order to test new ideas, resulting in better and faster technological advancement in the functioning of networks.

By exploiting the power of the open source development community should greatly accelerate innovation in SDN.

In addition to facilitating experimentation and research, open interfaces permit devices from different vendors to interoperate. This produces a competitive environment and therefore reduces the cost of network equipment for the consumers.

### D. Network Virtualization

The idea of virtualization is to create a higher-level abstraction that runs on top of the actual physical instance being abstracted. With the help of network virtualization, the network administrator is able to create, expand and contract a network anytime and anywhere as per the requirements [21]. Network Virtualization enables coexistence of multiple network instances on a shared physical infrastructure, thus, NV can be used to run an SDN solution. Also, Network virtualization in SDN is a good way to provide different users with infrastructure sharing capabilities because as the network grows so are the needs of the users [11][25].

## V. ADVANTAGES OF SDN

Major advantages of SDNs include [11],[13]–[17], [18]–[20].

### 1) Intelligence with Speed

SDN is intelligent enough to efficiently distribute the workload via powerful control plane resulting in high speed transmissions and making more efficient use of the resources.

### 2) Network Management Made Easy

The administrators have a centralized control over the network and can change the network characteristics as per the demand of environment. This enables administrators to modify the network configurations with ease.

### 3) Multi-Tenancy

The concept of the SDN can be expanded across multiple partitions of the networks such as the data centres and data clouds where there is a need to deploy their applications in virtual machines (VMs) across several sites. Existing architectures do not support joint intra-tenant or inter-tenant network control ability but SDN can support cross-tenant data centre optimization.

### 4) Virtual Application Networks

Virtual application networks make use the virtualization of network resources to hide the low-level physical details from the user applications and allow the users to reconfigure the network tasks easily.

## VI. CHALLENGES

These challenges continue to be relevant today in SDN. We list a few of them here [21]:

### 1) Latency

In SDN we have a centralized controller and the networking element requests policy directions from it, resulting in number of decisions that will suffer significantly in round-trip latency. The way and the extent this latency affects the operation of the network is undetermined. Furthermore, it is also not known whether the traditional servers on which the controller runs will be able to service these requests at sufficient speed so as to have minimal or no impact on network operation.

### 2) Security

Having a centralized controller means that the attacker has to focus on that one point of failure, and hence can lead to the modification rules/policies in the network devices, unauthorized access to the network, data leakage and deny a legitimate user to access the available resources (DoS) [22].

Therefore, it is important to consider some extra steps to protect both the centralized controller and the communication channels between the controller and the network devices.

### 3) Scale

Having a centralized controller means that responsibility for the topological organization of the network, determination of optimal paths, and the controller must handle device reconfiguration. But as more and more network devices are added to the network, a question of the ability of a single controller to handle all those devices arises.

It is difficult to know the solution when the number of network devices outgrows the capacity of the controller to handle them. If we attempt to scale the network by adding more controllers, how will they communicate, and who will control the coordination among the controllers?

### 4) High availability (HA)

The centralized controller is a single point of failure for the network, and if this fails the whole network will stop working. Obviously, this is not a preferable scenario so there is a need for redundancy schemes in various areas.

Firstly, there must be redundant controllers such that processing power is still available even in the event of failure of a single controller.

Also, the actual data needed by the set of controllers should be mirrored such that the controllers can manage the devices in a consistent way.

Furthermore, the communications links to the various controllers need to be redundant so as to ensure that there is always a working communications link between a switch and at least one controller.

## VII. EVOLUTION OF SDN ARCHITECTURE

SDN supports both centralized and distributed controller models. Both the models have different infrastructure elements and requirements to consider. This section describes the SDN models along with their advantages and disadvantages. Also, the hybrid SDN model is described which combines the benefits of both the approaches.

### A. The Centralized SDN Model

In centralized SDN architecture a single centralized controller manages and supervises the entire network. The network intelligence is centralized inside a single decision point. Since a single centralized controller is used to program the entire network, so it must have a global view of the loads on each switch across the routing path. Also, it must keep a track of which flow inside which router is presenting a bottleneck on which link.

The controller communicates with OpenFlow switches to collect network statistics from the network devices, and sends this data to the management plane. The management plane is software that consists of a database module and analytic algorithms that detects the switch overloads and predicts the future loads that may occur in the network.

Although the centralized control plane has an advantage of a single point of management and better control over the network, it incurs several limitations, (a) the controller needs to update OpenFlow switches more frequently than traditional ones. Thus, the topology discovery produces higher overloads because all ports must be scanned linearly. This increases the response time and may impose a higher overload for large-scale networks, (b) in the centralized model, the initial packet of every new flow in the system must first be forwarded to the controller for inspection. The controller determines the future path for the flow hop-by-hop. Thus, when a new flow is to be programmed, the controller needs to contact all the switches in the path, which is a scalability challenge for large networks, (c) The centralized controller represents a single point of failure which makes the network highly vulnerable to intrusions and attacks, (d) SDN networks are becoming more complex since they are designed to support different communication services and provide diverse functionalities such as intrusion detection, firewall, network virtualization, and load balancing. These services need to coordinate their activities in the control plane to achieve complicated control objectives and maintain a global view of the entire network. However, it is hard to fully coordinate the control actions and keep the consistency of network states among distributed functions.

### B. The Distributed Sdn Model

The distributed SDN model focuses on eliminating the single point of failure and enabling scale up by sharing the load among distributed controllers.

Distributed SDN control planes are designed to be more responsive to handle local network events in data centers. In particular, for multi-domain SDNs with a large variety of network technologies, the distributed SDN model is easily able to adapt to the network requirements. Additionally, a distributed controller is more responsive and can react faster and efficiently while handling global events.

Authors in [27] introduces two-layers of hierarchical distributed controllers: (i) bottom-layer consists of a group of locally non-connected distributed controllers each managing one or more switches without any global knowledge of the network, and (ii) the top-layer consists of a logically centralized root controller that manages the network-wide state. In addition, a cluster-based distributed model is proposed by authors in [28] where a master controller is selected based on the load in the network so that if the load increases, the master node can be switched to a less loaded one. Also, authors in [29] introduce a SDN Controller Cluster (SCC) that is composed of multiple controller instances interconnected over East–West interfaces. Further, in [30] the authors describe a controller

placement problem to decide the optimal number of controllers needed and their placement in the SDN network.

There are several key challenges faced by this architecture that must be addressed in the future SDN to improve scalability and robustness of networks.

(a) The above approaches require a consistent global view in all controllers. The mapping between control planes and forwarding planes must be programmed instead of the present static configuration, which can result in uneven distribution of load among the controllers.
(b) Finding an optimal number of distributed controllers that ensure linear scale up of the SDN network is hard.
(c) Such approaches mostly make use of local algorithms to develop coordination protocols in which each controller needs to respond only when an events take place in its local neighbourhood. Thus, there is a need to synchronize the local and the distributed events to provide a global view of the network.

### C. The hybrid SDN control Architecture [2]

To tackle the limitations in each of the approaches described above, hybrid SDN architectures are being taken into account. However, a critical challenge arises when determining how much of network abstraction modules can be centralized and efficiently designed to support logically centralized control tasks, and at the same time provide physically distributed protocols. Consequently, to take the advantages of both the centralized and the distributed architectures, a hybrid control plane is required to achieve such coordination.

The hybrid SDN model is influenced from by the benefits of the simple control of managing specific data flows as in the centralized model with the scalability and flexibility of the distributed model. It requires various components to coordinate the communication between SDN controllers. The network administrators will require standard interfaces, and policies to manipulate and interact with the control planes in distributed environments [31]. The hybrid SDN model may be useful in providing answers to (a) what state belongs in distributed protocols, (b) what state must be local to the switches, and (c) what state should be centralized. It can boost the network performance by facilitating efficient resource utilization because it will be easier to program each aspect of the network at the application level.

Furthermore, the hybrid SDN model could provide management policies to solve security issues, enable network Optimization, and state synchronization in the case of control plane overload. Also, hybrid SDN model allows up gradation of the existing infrastructure without the need to change the overall system.

## VIII. CONCLUSION

Software Defined Networks as a rising technology is bring modernization into the networking with decoupling of control plane and the data plane, and removing proprietary in the network architecture to open and programmable network. SDN is becoming increasingly popular due to the interesting features it presents that unlock innovation in how we design and organize networks. Due to various advantage of this architecture, many enterprises are shifting from the traditional network architecture to new SDN architecture. But still, there are some important challenges that need to be solved before realizing successful SDN with security being one of the main issues that threatens the future of SDN technology.

## IX.REFERENCES


[1] Machine Learning Based Intrusion Detection System for Software Defined Networks

[2] Software-Defined Networking: Challenges and research opportunities for Future Internet

[3] N. McKeown, Software-defined networking, INFOCOM keynote talk, April 2009, Rio de Janeiro, Brazil.

[4] H. Kim, N. Feamster, Improving network management with software defined networking, Communications Magazine, vol. 51 (2), IEEE, 2013, pp. 114–119.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, Openflow: enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. (2008) 69–74.
[6] ONF, The openflow 1.3.1 specification, Tech. rep. September 6, 2012.
[7] https://www.opennetworking.org
[8] Software-Defined Networking: A survey Hamid Farhady, HyunYong Lee ⇑, Akihiro Nakao
[9] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, M. Tyson, Fresco: modular composable security services for software-defined networks, in: Proceedings of Network and Distributed Security Symposium, 2013.
[10] M. Yu, L. Jose, R. Miao, Software defined traffic measurement with opensketch, in: USENIX NSDI, vol. 13, 2013.
[11] A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation
[12] S. Ortiz, "Software-defined networking: On the verge of a breakthrough?" Computer, vol. 46, no. 7, pp. 10–12, Jul. 2013.
[13] C. S. Li and W. Liao, "Software defined networks," IEEE Comm. Mag., vol. 51, no. 2, p. 113, Feb. 2013.
[14] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A retrospective on evolving SDN," in Proc. Workshop Hot Topics Softw. Defined Netw., Aug. 2012, pp. 85–90.
[15] 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 Proc. Int. Conf. Netw. Serv. Manage., Oct. 2010, pp. 330–333.
[16] H. Fei, Network Innovation Through OpenFlow and SDN: Principles and Design. New York, NY, USA: Taylor & Francis, 2014.
[17] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in Proc. ACM SIGCOMM Workshop Hot Topics Netw., New York, NY, USA, 2010, pp. 19:1–19:6.
[18] T. D. Nadeau and P. Pan, "Software driven networks problem statement," IETF Internet-Draft (Work-in-Progress), Oct. 2011, draft-nadeau-sdnproblem-statement-01.
[19] M. Yu, J. Rexford, M. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," Proc. ACM SIGCOMM Comput. Commun. Review, vol. 40, no. 4, pp. 351–362, Oct. 2010.
[20] K. Yap et al., "OpenRoads: Empowering research in mobile networks," ACMSIGCOMM Comput. Commun. Review, vol. 40, no. 1, pp. 125–126, Jan. 2010.
[21] Software Defined Networks A Comprehensive Approach
[22] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, et al., "Are we ready for SDN? Implementation challenges for software-defined networks," Communications Magazine, IEEE, vol. 51, pp. 36-43, 2013.
[23] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," presented at the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Hong Kong, China, 2013.
[24] N. Chowdhury, R. Boutaba, A survey of network virtualization, in: Elsevier Computer Networks, 2010.
[25] D. Drutskoy, "Software-Defined Network Virtualization," M.S. thesis, Princeton University, Princeton, NJ, USA, 2012. OpenFlow. [Online]. Available: http://www.openflow.org/
[26] H.Y. Soheil, G. Yashar, Kandoo: a framework for efficient and scalable offloading of control applications, 2012, pp. 19–24.
[27] M.O.S. Volkan Yazici, A.O. Ercan, Controlling a software-defined network via distributed controllers, in: 2012 NEM Summit Proceedings.
[28] Z. Cao, Z. Li, Analysis of sdn controller cluster in large-scale production networks, Tech. rep. 00, 2013.
[29] B. Heller, R. Sherwood, N. McKeown, The controller placement problem, in: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, 2012
[30] T. Nadeau, P. Pan, Framework for software defined networks, Internet-Draft draft-nadeau-sdn-framework-01, Internet-Draft, October 2011.
[31] K. Bakshi, "Considerations for software defined networking (SDN): Approaches and use cases," in Proc. IEEE Aerosp. Conf., Mar. 2013, pp. 1–9.
[32] OpenVSwitch. [Online]. Available: http://openvswitch.org/development/openflow-1-x-plan
[33] Pica8 Open Network Fabric. [Online]. Available: http://www.pica8.org/solutions/openflow.php

[34] Indigo—Open Source OpenFlow Switches. [Online]. Available: http://www.openflowhub.org/display/Indigo/