Software-Defined Networking - A Survey

Nandini Krupa Krishnamurthy
Clemson University
nandink@clemson.edu

***Abstract*—SDN has revolutionised networking domain by introducing a new of interacting between devices and has imparted to the architecture a better control. It has improved performance, increased reliability and programability and also have given good scalability. Legacy networks are way too complex and are integrated vertically. SDN overcomes this by using decoupling. SDN is unique in the sense that it decouples data planes from the control plane. In this survey paper, we start with introduction on SDN, then move to talk on how its architecture is different, its advantages and techniques used to implement SDN. We also throw light challenges associated with SDN and blooming researches in this field.**

I. Introduction

In this current world, purely defined by data, everything is nothing more than a piece of data. A persons name, age, email id, date of birth, address, or be it anything, is data. Everyday users of internet generate around 2.5 Quintilian bytes of data. We are in a trend of computers using high bandwidths, rich multimedia from a variety of resources and with a demand for high computational speed, with an oceanic flow of data. Added to this are the tsunami of mobile devices and users of social networks which have seen an exponential curve of growth in this era of data sciences. All these data are mostly stored of cloud which according to the raising demands must be robust and scalable. These are very minor examples of creation of demand for an increase in network infrastructure to boost the existing capabilities of computers. Additionally, since most of these data is stored in cloud, it is very much essential to have efficient access to these [1].

Traditional systems are not capable to meet the growing demands of the present world networks. The needs which these traditional systems cannot fulfill are explosion of mobile devices data, server-client traffic optimization and usage of cloud [2]. In other words, these systems cannot cope up with demands such as fluctuating network traffic designs, the storage of technology bloom, increase in the usage of cloud amenities, and the requirement of a better bandwidth for more data. These are inconsistent to meet the growing demands, are not scalable and their performance may not be optimal. To meet these requirements an instant solution would be to improve the network frame work by improving their capabilities rather than building it from scratch. Again, these are not problem free. One, to meet such huge demands, networks must be huge. Simplest of simple networks may contains various thousand devices. Two, these networks contain various dissimilar devices, from various resources adding to its complexity. Three, maintenance cost for these huge networks is high [1]. Thus, increase in complexity demands a simple framework which is supposedly easy to manage and also meets future demands.

Out of the various architectures available, the simplest and the most scalable one available is Software-Defined Networking (SDN). Probably SDN is best defined by ONF Open Network Foundation which is - Software-Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable [1].

Therefore, SDN has two basic properties decoupling of control plane and data planes and direct programability. Though this is not rare in the existing network architecture, what makes SDN different is its programibility using decoupling. It is also built upon simple networking devices. When we see its architecture as defined in Fig 1,we see that there is a control and data planes are separated thus denoting that controlling network flow can be carried out without affecting data flows.Before diving deep, let us first look into SDN architecture.

A. SDN Architecture

As illustrated in the Fig 1 below, SDN architecture consists of three layers the bottom-most is the infrastructure layer, the middle one is the control layer and on top lies the application layer [4].

The bottom-most is the infrastructure layer, also called data plane. Its basic functionality is to set up data path for the network. Essentially it consists of network elements whose basic job is to collect, monitor and forward local data. Due to this, it is also called as forwarding plane.

On top of this layer lies the Control layer, also called control plane. The main task of this layer is to program and control the forwarding plane. The forwarding plane delivers the data to the control plane which in turn uses this information to for network routing and function. There are some interfaces running between these two layers. These are called as south-bound interfaces and their task is to mainly carry out the communication between control and data layers. OpenFlow is the most widely used south-bound interface. These interfaces mainly consist of network components such as switches and routers.

The topmost layer is the application layer. This layer, as shown in the figure basically comprises of network application that deals with the introduction of new network features as and when necessary. These network features may include security, some networking schemes or some other feature aiding in the maintenance of the architecture.In addition to this, application

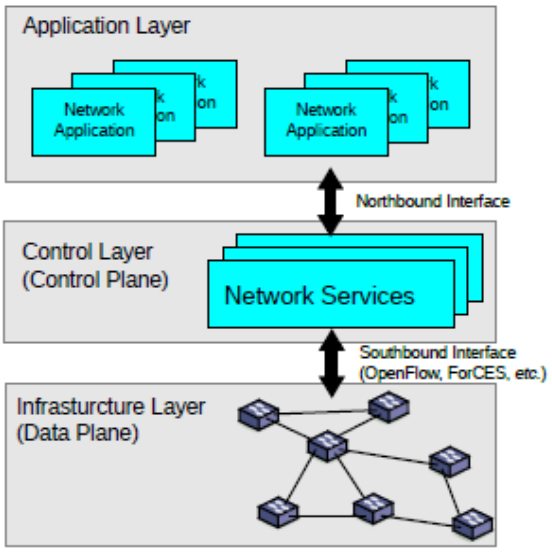

Fig. 1. SDN Architecture clearly showing the three layers and interfaces [4]

layer also carries out the function of assisting the control layers in its tasks buy providing guidance. The application layer and the control layer communicate using interfaces called north-bound interface. There is no well defines API for these interfaces and control layer provides its own API to applications.

The separation of the network and data planes helped in implementing network abstraction where we just have to program the SDN network controller and all the logic will go down to individual nodes through southbound APIs and protocols. Here, controller pushes the instruction from controller and virtually control the network. All applications work on the top of the software layer but sends all the instructions down to the components. This helps in adapting to changing business requirements very fast as all individual components dont need to be physically programmed but just the network controller. This saves a lot of time as well as abstracts the network view. Since OpenFlow is the most popular standard, we will be discussing that below along with the three planes that constitute the SDN.

B. Advantages of using of SDN

There are various uses of using SDN. Some of them are defined as follows [2] [4] [5]:

1) **Central control:** SDN is centrally controlled. This means that SDN control software can control interfaces by any seller rather than differing in the management by a group of vendors.
2) **OpenFlow advantage:** SDN using OpenFlow reduces automation tasks because it is flexible enough to create tools for automation of many manual tasks being done.
3) **Less complexity:** As this can be automated, complexity of networks can be reduced. Operational costs may also be lessened due to this.
4) **Greater performance:** SDN does not need to configure network every time a device is added or removed from the network. This increases its robustness, reliability, optimum working environment, dynamic nature and security.
5) **Consistency:** SDN allows the changed to be implemented at the granular level. This helps in maintaining consistency.
6) **Adaptability:** Since SDN is dynamic and adaptable to better user needs it gives a better user experience.

II. Main Techniques

SDN is considered to have better architecture because it possess optimal routing. This optimal routing is the main reason behind its unique design and performance. These methods reduces the complexity of SDN thus increasing its performance and reliability. The architecture SDN uses for RouteFlow is as shown in Fig 2.

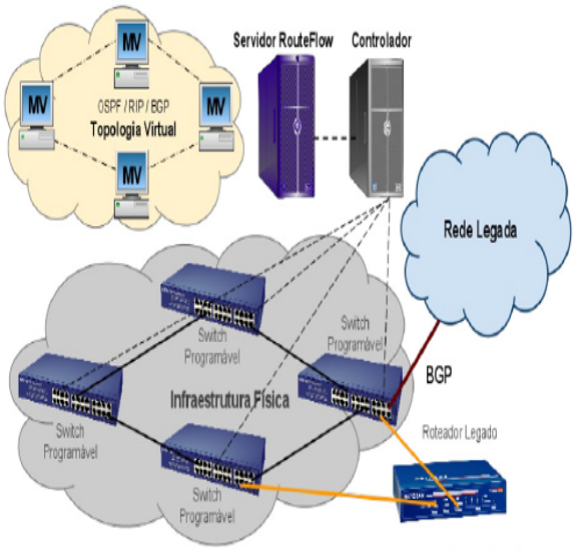

Fig. 2. Architecture of RouteFlow for SDN [6]

The route packets take is simplified by the unique concept of SDN where control and forward are separated. This section concentrates on various techniques used in SDN for routing.

A. Routing Control Platform

In Routing Control Platform or RCP, routing is carried out as an independent unit. RCP formulates and connects inter-domain and intra-domain RCP routes. The three factors behind the principle of RCP are [6]:

- Networks consistent view of state.
- Controlled routing protocol interaction

- Expressive and flexible policies

The implementation of RCP in SDN follows Intelligent Route Service Control point or IRSCP which already existed before the SDN came into the picture. RCP can be explained in simple terms. iBGP Route Reflector (RR) gathers information and shared with the border routers. Best route is calculated after this. The entire information needed for routing resides in Control plane. RCP also possess a local registry which has information on inter-domain routing.

B. The Softrouter

In Softrouter, Forward Elements (FE) and Controlling Elements(CE) are separated. CE and FE form a Network Element (NE) by setting up a port. CE and FE may be many places away but are connected by a centralized server. ForCEs Is the concept in communication between these elements.

C. RF IP Routing: IP routing services over RouteFlow based SDN

RouteFlow is made up of three elements:

- RouteFlow Controller (RC)
- RouteFlow Server (RS)
- Virtual network Environment (VE)

The server and the controller act together using protocol called as RouteFlow protocol. RouteFlow Client and Routing Engine are the components of VE. Routing Engine provides Forwarding Information Base (FIB). This is received by RouteFlow client which in turn is responsible for converts these to OpenFlow tuples. These are then sent to RS which establishes the routing topology using the information provided in them. This information is further shared with RC which takes the necessary action required. VE is connected to RC with a Virtual Switch. NoSql was introduced in this architecture to aid in inter-process communication. RouteFlow also accomplishes logical splitting, multiplexing and aggregation. RouteFlow is the basic architecture in SDN.

D. VRS: Virtual Routers as a Service

Virtual Router System or VRS is a technique in which the forwarding elements are distributed. Its topology consists of a Core Node (CN) which is connected to Customer Edge Gateways (CEG) which are, in turn, connected using Intermediate Nodes (INs). A Virtual Router (VR) interacts with this topology using Point-Of-Presence (POP). OpenFlow is used to program Forwarding Engines. The decisions needed for routing are carried out in the CN. There is also the existence of a Virtual Machine (VM) which that builds up upon the rules needed to carry out forwarding. This uses another special instance called as Path Management Controller (PMC). As the name defines, PMC calculates the route with the minimum cost. And as expected, cost of VRS is directly proportional to the number of CEG nodes. VRS comprises of two actions [6]:

- Core node location selection
- Optimal forwarding path allocation

E. RFCP: RoutFlow Routing Control Platform over SDN

RFCP is a hybrid model between RCP and RouteFlow. This has another extra layer for formulating routes. It also possess a data platform which stores RFCP Core state information, Network View and Network Information Base (NIB). The southbound communication between the components is carried out by OpenFlow protocol control message. RFCP has three components, namely, RouteFlow-Client, RouteFlow-Server and RouteFlow-Proxy. To form a hybrid network, this architecture is linked to traditional one by using Route Reflecter (RR). To aid this integration, we have RR connected with Provider Edge (PE) using iBGP which communicated with RFCP-Controller. The gateway her is the RouteFlow-controller. After RouteFlow-controller gathers information on routing, RouteFlow-Server defines the routing path. An additional inclusion is a discovery controller which automatically designs RFCP.

Topology controller monitors for any new switch occurs . Its configuration contains switch ID and port number which is used to create a virtual machine with same information resembling an actual network. Information about all these is stored in the form of configuration files in a database. RPC client takes up the information from these files and passes this on to RPC Server which uses these information in the virtual network it just created.

F. Raas:Routing as a Services

Raas is based on OpenFlow architecture. It uses a centralized control plane which shows a global view and this in turn, helps in making the routing decisions. The routing block has following components:

- Link Discovery module
- Topology Manager
- Virtual Routing

G. CAR-Cloud Assisted Routing

CAR is another example of an architecture which uses Service. It essentially follows two principles:

- CPU Principle
- Memory Principle

CPU principle is used when routing functions are heavily computational and also have to be shifted to cloud. For this control plane is kept closer to cloud. In memory principle, data plane is closer to the cloud. This type of architecture is made up of two types of routers hardware routers which holds incomplete FIB and software routers which holds complete FIB. The architecture primarily talks about shifting heavy computation to cloud. The basic idea lies in deciding on what percent of this computation should be sifted on to cloud.

III. Issues and Problems

Though the concept of SDN increases scalability, reliability, flexibility along with managing network devices centrally and automating them, there are various issues associated with SDN [7] [8]. Below are listed some of them.

1) SDN is based on decoupling of control plane and data plane. But they may also include a middle layer called management plane which usually looks after managing with fault, account and security. In large networks, with the ever growing managing devices, there is a need to automate the network management.
2) Besides automation as said in point above, it also needs to work with latest trends. But the management in SDN is not at all given importance in the available SDN networks and these still use traditional systems.
3) There is still no clarity on if the management would be centralized or distributed.
4) SDN is totally dependent on controller. Hence this has to be monitored constantly and made to work according to the instant needs. It should be able to add/delete controller functionalities as and when required. But usually this is not the case in the existing SDNs.
5) The main issue with SDN lies in security. Since this uses a virtual system, it becomes very essential to use a safe environment. Added to this, SDN has shared memory, hardware and system components making it essential to have high security.
6) Another important aspect is the presence of varied types of components and switches which challenge interoperability.
7) Most of the SDN networks has virtual environments which are not unified.
8) SDN framework has a property programability. This is usually used to make the tasks simpler. But so is not the case as the ideas on how to layer the architecture have not be properly demarcated so far.
9) There are a variety of technologies used in the forming an application of SDN. This obstructs from creating a simple model which can demonstrate all models.
10) SDN Programming language is the ideal choice to have SDN applications implemented. But this is just not the case.

IV. Future Trends

There are various ongoing researches to fulfill the promises made by SDN. This sections talks about the some of these researches and future work related to SDN. We will go through the available scope for future work here [8].

A. Layer 1: Data Plane

1) **Software Switching**: Software switching is mostly used in cloud computing, where it connects the various virtual components to the same host. One example for this is OpenvSwitch. Reports show that when used in cloud it shows 25% more efficiency than compared to when run on Linux OS. So it is essential to formulate new functionalities to increase efficiency even when run on Linux. Another aspect is to decrease the latency in switching
2) **Southbound Interface Protocol**l: Though OpenFlow is the most widely used protocol, there are other existing which are completely ignored. Open vSwitch Database Management Protocol (OVSDB) is one such protocol. OVSDB is way ahead compared to OpenFlow in terms of having good features, but when compared to OpenFlow, it is less popularly used. OVSDB also has better management capabilities. So this may be a good research when OpenFlow and OVSDB could cooperate for better networking.
3) **Network Slicing**: Slicing is a process where the forwarding components are sliced so as to accommodate more virtual networks over the same existing physical devices. FlowVisor and libNetVirt are examples of these. Using the properties of network slicing, business applications may be generated.

B. Layer 2: Control Plane

1) **Controller placement problem**: Here, the controller checks the amount of flow so as to have an optimum function. It is thus understood that switches do contact the controller as and when required to have an optimum flow for the best possible output. When a simulation was carried out on the placements of these controllers, it was seen that random placement increases the latency rate by 2 times. So the research area here is to find the optimum placement.
2) **Centralized or distributed controller**: There is still researches going on to find which type of controller is better suited centralized or distributed. Centralised controller increases latency whereas distributed controller increases complexity.
3) **Controller Services Application portability**: Every network operating system has controllers which has its own API. SDNs are specific to a network controllers. They do not have a well defined standardized API. So applications may be created which can operated on all controllers uniformly rather than each controller having its own API leading to increase in complexity and inconsistency.

C. Layer 3: Application Layer

1) **Programming Language Design**: SDN Programming language is the ideal choice to have SDN applications implemented. Various applications are written in other languages. Another research area would be a creation of features based on SDN based languages.
2) **Big Switch abstraction Rule Placement problem**: Rule Placement problem essentially simplifies SDN but using a single switch instead of using multiple switches. This is an optimization problem. One Big Switch [8] is proposed which can divide the rules based on paths and gets the aggregation of all paths based on the rules set up and thus helps in finding an optimum path.

References

[1] Wenfeng Xia, Yonggang Wen, et al Survey on Software-Defined Networking, IEEE Communication Surveys and Tutorials, Vol. 17, No. 1, First Quarter 2015 .

[2] Software-Defined Networking: The New Norm for Networks
[3] Ominike Akpovi A, Ominike Akpovi A, Osisanwo F. Y ”Introduction to Software Defined Networks (SDN)”, International Journal of Applied Information Systems (IJAIS) ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 11 No. 7, December 2016 www.ijais.org
[4] Wolfgang Braun, and Michael Menth, Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. Department of Computer Science, University of Tuebingen, Sand 13, Tuebingen 72076, Germany, May 2014.
[5] Kamal Benzekki, Abdeslam El Fergougui and Abdelbaki Elbelrhiti Elalaoui, Software-defined networking (SDN): a survey, Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1737, Feb 2017.
[6] Sahrish Khan Tayyaba, Sadaf Tanvir, Nadeem Javaid, Routing Techniques in Software Defined Networks: A Survey , COMSATS Institute of Information Technology, Islamabad,Pakistan, Conference Paper January 2016.
[7] Antonio Manzalini, Telecom Italia, Italy , Roberto Saracco, EIT ICT Labs, Ital , et al.Software-Defined Networks for Future Networks and Services, White Paper based on the IEEE Workshop SDN4FNS (Trento 11th 13th November, 2013), January 2014.
[8] Mohammad Mousa, Ayman Bahaa-Eldin, and Mohamed Sobh,Software Defined Networking Concepts and Challenges, Conference: 2016 11th International Conference on Computer Engineering & Systems (ICCES), DOI: 10.1109/ICCES.2016.7821979, December 2016.
[9] Diego Kreutz, Fernando M. V. Ramos, et al. Software-Defined Networking: A Comprehensive Survey, arXiv:1406.0440v3 [cs.NI], Oct 2014.