Journal of Engineering Science | *Vol. XXV, no. 3 (2018), pp. 61 - 73*
Fascicle *Electronics and Computer Science* *ISSN 2587-3474*
Topic *Computers and Information Technology* *eISSN 2587-3482*

DOI: 10.5281/zenodo.2557306
CZU 004.7

SOFTWARE - DEFINED NETWORKS … A GENERAL SURVEY AND ANALYSIS

Ali Ameen

Technical University of Moldova, 168 Stefan cel Mare Av., MD-2004 Chisinau, Republic of Moldova
Corresponding author: Ali Ameen, email: *alisalmanhussein@yahoo.com*

Received: September, 14, 2018
Accepted: November, 22, 2018

Abstract: Technology of Software-defined networks is rapidly evolving in order to develop the networks world, hence to provide better connectivity, agility and security to defend the world of cyber information against the ever-evolving security threats and challenges. This article is totally dedicated for beginners in both the IT world generally and the SDN world specifically as well. Through software-defined networking we can enter a new era of portable network management interface simply by using a mobile app or your smart watch, which gives a great deal of smoothness for network administration, and simplifies the IT tasks, but on the other hand it loads the stack of cyber security challenges with new ones.

Keywords: *Southbound API, northbound API, openflow, Open vSwitch, ASIC, NFV.*

1. Introduction

Despite that, every aspect of technology's world was and is continuously evolving but still the world of networks is a bit late in that race, nearly since the 80s the networks have not significantly evolved, and here it comes the age of SDN solution.

We can think of it not as a new invention but as a new way to divide and manage network architecture layers and resources. In this article we'll try to give simple analysis and general survey for the whole ongoing development procedure, draw some simple comparisons for the main defining features of this new technology like its controllers and give some main pros and cons for both SDN and traditional networks.

Now as you may know the traditional network device like a switch would conduct the forwarding procedure according to the rules specified by the admin by entering a bunch of commands in the command line interface CLI of the device, which means that this device is a standalone unit that contains it's own brain, that brain is called (the Management plane) because it's possible to manage that device through its brain to make it achieve its admin-specified objectives, that part that runs the commands of the brain is called (the Data plane).

So we can say that the traditional switches have that feature in them where both the management and data planes reside and co-exist together within the same unit and that's one of the main features of the traditional networks generally.so let's draw a simple comparison between both the traditional and the suggested new SDN technology.

2. What is SDN

SDN dates back to more than 20 years ago, despite that the term SDN was coined in 2009, but the earliest ideas date back to the time of phone network.
Historically speaking, the main intellectual features of SDN are as follows:

- Active networks: which means the ability to program networks to do special tasks and calculations through its nodes and connection points like routers and that was really the first obvious attempt to make networks programmable, which gave the researchers and developers a new opportunity to deploy new ideas and embed them within that kind of innovation, because researchers, developers and network operators were frustrated by the new ideas and new innovations deployment challenges.

That project provided a significant amount of reduction in computation cost, and that attracted more interest from funding agencies.

- Control and data plane separation: which led to the development of open interfaces between control and data planes. Tackling traffic engineering problems was the main feature defining the early technological attempts to provide network programmability.

Working tech groups in the internet engineering task force (IETF) like ForCES designed open interfaces between the control and data planes, while others developed logically centralized control technologies like Routing control platform (RCP).

One of the push-toward-SDN reasons was that separation of control plane from data plane tackled significant specific network management problems, which was of a great help to network admins. So, in other words that kind of separation has some intellectual contributions for networks technology and we can summarize it with 2 main points:

  - Logically centralized network control through open interfaces connected to routers/switches.
  - Provided algorithms for creating a distributed state management for distributed network controllers.

And unlike what many would think, logically centralized route control doesn't violate the fate sharing, while some exiting routing technologies like OSPF and BGP route reflectors already violated this type of principles; on the contrary the obvious separation between the control and data loops provided developers and researchers with better ability to think of cleaner ways to achieve distributed state management for networks. [1].

Openflow API and NOS: that represents the first instance of the formal widely spread adoption of an open interface. the openflow technology took a general trend and approach adding more functions on earlier route controllers and deploying new ideas and techniques on existing switch hardware; despite that relying on exiting switch hardware support caused some limited flexibility, but on the other hand it provided instant deployability somehow.

The technological reason for developing the openflow was a combination of reasons that would form the ((the perfect storm)) between network administrators who face significant network management problems, vendors who are living in an ever evolving technological market competition, chipset designers who started new approaches to open their APIs and technology researchers who are looking for new ways for developing networks world.
Openflow in the beginning was adopted and tested in university campuses like Stanford's but then it spread to data centers.

Openflow itself has some intellectual contributions for the world of networks like:

- Now control and data planes can support general networks functions and devices.
- A new vision for the network operating system (NOS) layers which are slightly different from the general definition of SDN infrastructure which contained the control, data and management loops or layers; while here we see that the openflow's contribution provides the following layers for NOS:
  - Data plane in conjunction with an API
  - Network state management layer
  - A control logic layer that affects the data plane based on the network's state.

Moreover, that new vision provided new techniques for distributed state management.

Least but not last, unlike what many would have thought but the first packet of every dataflow doesn't necessarily go to the controller, so we can say that this is a myth about openflow, while the truth is that this is not up to the openflow to decide or to assign the level of granular management, also we should know that with openflow networks the controller doesn't' have to be centralized, actually the truth is that most real deployments have distributed controllers.

Also, we should recognize that openflow is not equivalent to SDN because openflow is just a particle in that compound called software-defined networks.

3. Traditional networks vs software-defined networks

The old traditional network architecture provides complexity in the management, configuration and resource allocation to some extension (figure 1).

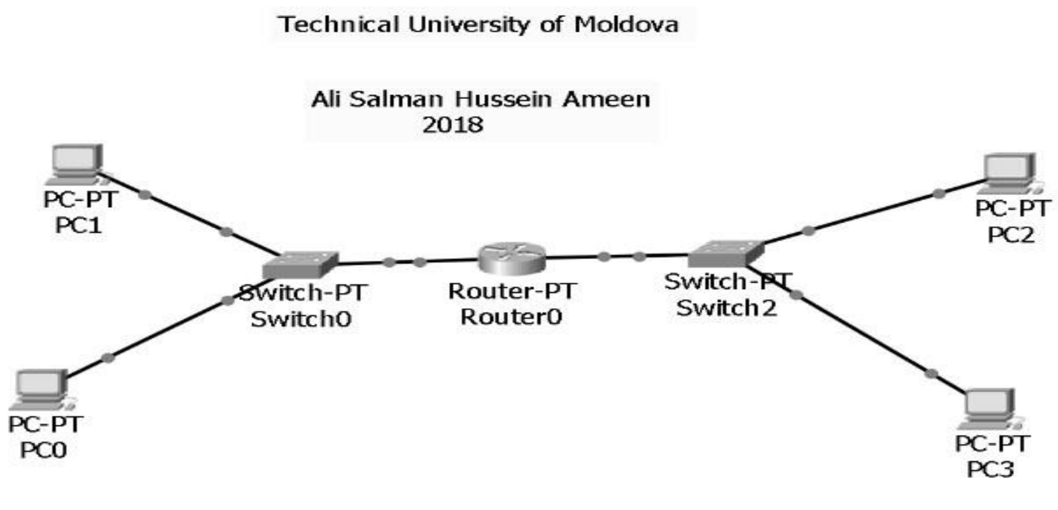

Figure 1. Traditional network.

Traditional network configuration versus SDN configuration, the traditional networking approach is of two main factors:

1) Network functionality is mainly implemented in a dedicated unit. In this case, 'dedicated appliance' means one or multiple switches, routers and/or application delivery controllers.

2) Most functionality within this apparatus is implemented in dedicated hardware. An Application Specific Integrated Circuit (or: ASIC) is often used for this purpose. Organizations are increasingly confronted with the
Limitations that accompany this hardware-centric approach, such as:

a- Traditional configuration is time-consuming and error-prone

Many steps are needed when an IT administrator needs to add or remove a single device in a traditional network. First, he will have to manually configure multiple devices (switches, routers, firewalls) on a device-by-device basis.

The next step is using device-level management tools to update numerous configuration settings, such as ACLs, VLANs and Quality of Service.

This configuration approach makes it that much more complex for an administrator to deploy a consistent set of policies. As a result, organizations are more likely to encounter security breaches, non-compliance with implications. Conclusion: the highly administrative 'hassle' that is traditional configuration interferes with meeting business networking standards [2].

b- Multi-vendor environments require a high level of expertise

The average organization owns a variety of equipment of different vendors. To successfully complete a configuration, an administrator will therefore need extensive knowledge of all present device types.

c- Traditional architectures complicate network segmentation

A development further complicating networking matters, is the connectivity evolution that is currently occurring. In addition to tablets, PCs and smart phones, other devices such as alarm systems and security cameras will soon be linked to the internet. The predicted explosion of smart devices is accompanied by a new challenge for organizations: how to incorporate all these devices of different vendors within their network in a safe and structured manner.

Many traditional networks place all types of devices in the same 'zone'. In case of a compromised device, this design risks giving external parties access to the entire network. This can be hackers exploiting the internet connection of smart devices or vendors who can remotely log onto 'their' devices. In both cases, there is no apparent reason for giving them access to all network components. However, the 'administrative hassle' described earlier makes network segmentation a complex process and quickly leads to network clutter [3]. In conclusion, to overcome these and other traditional networking limitations, the time has come to introduce a new perspective on network management [2].

And now let's talk about embracing the change of Software Defined Networking.

Networking Software Defined Networking (SDN) is rapidly becoming the new buzzword in the networking business. Expectations are that this emerging technology will play an important role in overcoming the limitations associated with traditional networking. But what exactly is SDN?

Even though a universally agreed upon definition for SDN has not yet been formulated, 'decoupling hardware from software' is often mentioned when this topic comes up. This concerns the two network device planes,
i.e.:

1) The brain is represented by the plane that determines where to send traffic (control plane)

2) The muscles, which are represented by the plane that executes these decisions and forwards traffic (data plane)
Decoupling these two planes involves leaving the data plane with network hardware and moving the control plane into a software layer. By abstracting the network from the hardware, policies no longer have to be executed on the hardware itself. Instead, the use of a centralized software application functioning as the control plane makes network virtualization possible. This process is similar to server virtualization:

a- Server virtualization

The process of creating various VMs (virtual machines) and decoupling them from physical servers.

b- Network virtualization

The process of creating virtual networks which are decoupled from physical network components.

4. Why do we need SDN?

- Cost saving: when building and managing the infrastructure devices for the used network. So, in order to start anything new, we should know what's in it for us because it could be less in the expenditures, hence it will be useless to refurbish the network and update it. Also virtualization plays a major role in cost saving.
- More configuration accuracy, consistency and flexibility through Programmability : SDN technology provides control and central management and programmable network which gives flexibility in the design, application, management, security improvement ,time, speed ,automation ,reliability ,problem solution and error overriding ,which suits largely the big needs and future continuously changeable services.
- Security: despite that the centralization of the controller could make new security threats show up but for the current time we could say the SDN provides a great deal of security in the foreseen future of networks especially against the ever-evolving security threats , in this design we now have only one device to configure it that will act as both the mind and the firewall at the same time, it will control the flow tables that exist in the switches via the ONF-made openflow protocol , which will restrict the routes for unwanted data packets, and it will also be easier to monitor the networks status through one device or software(In case of controller virtualization) than monitoring multiple devices which will be a hideous procedure because we will have more network vulnerabilities since every single device will be a potential attack point in case of traditional networks.
- Optimizing Data Flow: A second expected business benefit of the SDN approach, is the optimization of data flows. Instead of having a single path from the source of communication flow to its destination, a SDN controller is able to identify multiple paths per flow. Furthermore, this approach allows the flow's traffic to be split across multiple nodes. Network performance and scalability is enhanced by optimizing the network path for a particular data flow based on the source and destination nodes.
- Growing technology needs: increasing data stacks and soaring internet speed also new types of data stacks like clouds , growing number of internet users all over the world , is making a huge pressure on networks and they are getting too slow to deal with that.

5. Main components of SDN

There are many areas and field to focus on during the research about software-defined networking due to the variety and diversity of SDN's components, but the main ones are:

5.1 Controllers

Since the main feature of software-defined networking is the ability to manage it centrally using controllers, then it's essential to talk about them as well. Despite that SDN is an approach to unify vendor proprietary network devices and despite that many efforts were done in that matter to prevent Data centers and network enterprises from being locked in to a single source, but still there are many types of controller products that emerged on the surface, there are hardware-based and software based controllers but since SDN emphasizes the usage of virtualization, so it's preferred to use software-based controllers some of the main and most prominent controllers that you can use even for the purpose of research and test-bed environment creation:

1. Open Daylight
2. CISCO's Application Policy Infrastructure Controller (APIC)
3. Floodlight
4. ONOS
5. RYU
6. HP Virtual Application Networks (VAN) SDN Controller

Now since there are many products mentioned so we'll try to give a simple comparison between their main features by providing concise definition for each one of them.

5.1.1 The OpenDaylight Project

Which is the open source project and the result of the collaborative work hosted by The Linux Foundation. The aim of the project is to promote software-defined networking (SDN) alongside with network functions virtualization (NFV). OpenDaylight software is written in the Java programming language [3].

It supports technology such as OpenFlow, which we will discuss later in this article. The first version of the OpenDaylight project, named Hydrogen, was released in February 2014 [3]. A source code repository includes contributed source code initially seeded from Big Switch Networks, NEC and Cisco. There are now over 1000 cumulative contributors from various organizations as well as unassociated individuals. There is a dedicated OpenDaylight wiki, and several mailing lists are available. All these resources are aimed at developers wishing to contribute to the project, as well as other researchers interested in learning about specific sub-projects.

5.1.2 CISCO's Application Policy Infrastructure Controller (APIC)

Cisco application centric infrastructure (ACI) uses a controller called the Application Policy Infrastructure Controller (APIC). APIC the implementation of APIC is as a cluster of three boxes for the purposes and benefits of fault-tolerance and scalability.

The primary goal of the controller cluster is to provide policy resolution mechanisms and policy authority. a complete failure or disconnection of all elements in a cluster will not result in any loss of existing data center functionality because the system isn’t directly involved in data plane forwarding, or enforcing policies on individual network flows.

5.1.3 Floodlight

The Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. It is supported by a community of developers including a number of engineers from Big Switch Networks [4].
In addition, it worths mentioning that since Floodlight is Apache-licensed, that lets you use Floodlight for almost any purpose. Also, it works with physical- and virtual- switches that speak the OpenFlow protocol, plus Floodlight is developed by an open community of developers [5]. also worths mentioning that Floodlight is drop dead simple to build and run and least but not last Floodlight is actively tested and improved by a community of professional developers.

5.1.4 Onos

It stands for open network operating system, which is also a whole project and open source community hosted by Linux as well, “ONOS is the only SDN controller platform that supports the transition from legacy “brown field” networks to SDN “green field” networks. This enables exciting new capabilities, and disruptive deployment and operational cost points for network operators” [6].

5.1.5 Ryu

Ryu is a component-based software defined networking framework. “Ryu provides software components with well defined API that make it easy for developers to create new network management and control applications” [7]. “Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, OF-config, etc. About OpenFlow, Ryu supports fully 1.0, 1.2, 1.3, 1.4, 1.5 and Nicira Extensions” [7]. All of the code is freely available under the Apache 2.0 license. Ryu means "flow" in Japanese. Ryu is pronounced "ree-yooh".

5.1.6 HP Virtual Application Networks (VAN) SDN Controller

According to HP’s controller technical paper, HP VAN SDN controller is the central building block of HP SDN network and creates a platform for application innovation; HP Virtualized Application Networks SDN Controller provides centralized control and automation for your SDN network. “The HP SDN Controller controls policy and forwarding decisions, which are communicated to the OpenFlow-enabled switches in the data center or campus network”. [8] “A variety of HP and third-party SDN applications can leverage the SDN controller to automatically deliver the necessary business and network service levels” [8].

The architecture of software-defined networking separates the network control plane from the forwarding hardware. Control—which basically represents the brains of the network: is centralized, while forwarding: remains distributed. With the separation of the control plane from the forwarding plane, SDN provides the possibility for the network status and capabilities to be exposed or attached directly to the layer of business service, in order to enable the business systems to request services from the network right away. Thus, SDN applications provide a higher direction of level application to the controller. Moreover, freed from the control function, the forwarding plane can then provide optimized packet processing at very high speeds.

5.2 Switches

Switches are the second main element in the software-defined network architecture, they represent the forwarding or data plane and they have different algorithms and tools in use to forward packets in and out of their ports, according to their support to openflow protocol they are of two types pure and hybrid; where pure switches are just forwarding devices that cannot support the network legacy features, in other words they can't be used with traditional networks ; on the other hand hybrid network switches which represent the biggest percentage of the switches in the market, they support both the traditional network architecture and also the openflow SDN protocol. There two types of SDN switches:

- Software-based switches
- Hardware-based switches

5.2.1 Software-based switches

"Software-based switches are more frequently used in creating test environments to run an OpenFlow test-bed or for research purposes to test and develop OpenFlow-based network application" [9].

Nowadays, there are some OpenFlow switches which can be used, and some of them are have a brief description attached to them, including the OpenFlow standard version that they support and the implementation language, as follows:

1. **Open vSwitch:** Open vSwitch can be defined as a multilayer and production quality virtual switch that is licensed under the Apache 2.0 license. It's has the design that enables automation of network via programmatic extension and in the same time it's still supporting legacy and standard management interfaces and protocols as well (like OpenFlow, 802.1ag, RSPAN, sFlow, NetFlow, IPFIX , command-line interface (CLI), LACP, OVSDB, and so on).
2. **Indigo:** "Indigo the first generation is an open source OpenFlow implementation that could be installed on physical switches and uses the hardware features of Ethernet switch ASICs to run OpenFlow at line rates." **[9]** It is originally based on the OpenFlow reference implementation of the Stanford University. Indigo Virtual Switch (IVS), based on the Indigo Framework was introduced, which is an open source virtual switch for Linux compatible with the KVM hypervisor and that uses the Open vSwitch kernel module for packet forwarding.

   It is a high-performance, lightweight vSwitch built from the ground up to support the OpenFlow protocol. It is designed to enable high-scale network virtualization applications and supports distribution across multiple physical servers using an OpenFlow enabled controller, similar to VMware's vNetwork, Cisco's Nexus or Open vSwitch.
3. **Pantou (OpenWRT):** Pantou (OpenWRT) has the ability to turn a commercial access point/ wireless router to an OpenFlow-enabled switch. OpenFlow can be implemented and installed as an application on top of OpenWRT. Pantou is based on the BackFire OpenWRT release (Linux 2.6.32). The OpenFlow module is based on the Stanford university reference implementation (userspace). it supports generic Broadcom (BRCM43xx Wi-Fi) and some LinkSys models and TP-LINK (WR1043ND) access points with Broadcom (BRCM47xx/953XX) and Atheros (AR71xx) chipsets.

4. LINC: "LINC is a pure OpenFlow software switch. It is implemented in operating system's userspace as an Erlang node" [10] and with Such approach it does not become the most efficient one, nevertheless it allows quick development and testing of new OpenFlow features and gives a great deal of flexibility. LINC is an open source project led by Flow Forwarding effort and is an Apache 2 license implementation based on OpenFlow 1.2/1.3.1. LINC is architected to use generally available commodity x86 hardware and runs on a variety of platforms such as Linux, Solaris, Windows, macOS, and FreeBSD, where the Erlang runtime is available because it is written in Erlang. The benefit of the x86-based platform is that LINC can take advantage of the availability of lots of CPU cores and memory and scale gracefully to increase and decrease compute resources. This is critical when many logical switches are instantiated on a single OpenFlow capable switch.

5. Xorplus: provided by Pica8 and supported by the open community, it is also a switching software that provides no-cost Layer 2/Layer 3 protocol stacks, enabling the community to innovate. It supports protocols such as PIM-SM, IGMP, IGMP snooping, VRRP, IPFIX, and SNMP. In addition, it powers Pica8's current quickly line of switches.

6. Of13softswitch: it is a project supported by Ericsson Innovation Center that is located in Brazil and CPqD handles the maintenance in technical collaboration with Ericsson Research.

Of13softswitch is compatible with OpenFlow 1.3 user-space software switch implementation based on the TrafficLab 1.1 softswitch of Ericsson. The latest version of this software switch includes the switch implementation by (ofdatapath), the secure channel for connecting the switch to the controller by (ofprotocol), a library for conversion from/to OpenFlow 1.3 (oflib), and a configuration tool which is (dpctl).

7. Cisco's Nexus 1000V: it is a highly-secure, multitenant-services provider virtualization intelligence, supported by the VMware hypervisor beyond VMware vSphere Release 6.0 as well as on other major hypervisors.

According to Cisco's website they define this type of switch that it is filled with many features like: Supporting vCloud Director and vSphere hypervisor, Extensive virtual network services built on the advanced service of insertion and routing technology by Cisco, consistency of management to facilitate the integration with the physical infrastructure, plus management of Policy and control by the networking staff instead of the server virtualization team (which means duties separation) and Exceptional policy and least but not last control features for thorough and wide networking functionality.

5.2.2 Hardware-based switches

1. Zodiac FX: The Zodiac FX is the world's smallest and most affordable OpenFlow enabled switch. It provides many of the characteristics of an enterprise grade switch; also, it is small enough to fit in the palm of your hand, making it perfect for research and education. It currently supports OpenFlow version 1.0 and version 1.3.

2. Hewlett Packard 2920: The HP 2920 Switch Series consists of four Switches which are: the HP 2920-24G and 2920-24G-PoE+ switches with 24 10/100/1000 ports and the HP 2920-48G and 2920-48G-PoE+ switches with 48 10/100/1000 ports. Each switch has four dual-personality ports for 10/100/1000 or SFP connectivity. These switches support traditional technologies coupled with OpenFlow® support for SDN. The 2920 series of switches are most suitable for high-performance networks present in enterprise networks.

3. Pica8 (P3297, P3930): Pica8's open switches series is ideal for cloud or virtualized data centers that require flexibility and adaptability. These switches can smoothly integrate with today's data center applications on classical network architectures, while in the same time allowing the exploration of new Software-Defined Networking (SDN) technologies, like OpenFlow. Pica8 white box switches run PicOSTM, an open network OS that runs standards-based Layer 2/Layer 3 protocols with industry-leading OpenFlow 1.4/Open vSwitch (OVS) 2.0 integration.
OVS runs as a process within PicOS and provides the OpenFlow interface for external programmability. ''PicOS utilizes proven high-performance hardware with a switching fabric capacity of 1.28 Tbps and options for 10 GbE copper and fiber connectivity" [11].

5.3 Openflow

OpenFlow is an example of application programming interface (API) and communications protocol that gives access to the forwarding plane (Data Loop) of a network switch or router over the network, simply speaking it helps the control plane represented by the controller to connect to the data plane represented by the switch, using an API it becomes so handy to adjust network's behavior as per the business needs, through openflow it's possible to add, delete and modify the entries of the flow table installed in the switched which contains the information about the traffic packets forwarded, deleted or discarded. OpenFlow is one of the main defining features of SDN, with this protocol it's possible to monitor network's status, provide a reasonable amount of security and change the business policies according to the network administrator's needs and requirements as well.

6. Architecture Of SDN

As mentioned before, the main network operating system's layer of software-defined networks can be categorized as management, control and data layers, but with the cooperative trend and efforts of all tech companies and information security forums and foundations comes some vendor-based subsidiary differences, so we see that most people would think that the controller would manage every aspect of network's policy without distributing the control in other words the brain and all functions will be concentrated in only one point of the network architecture but that wouldn't be totally right because that will also limit the scalability, for instance we have the Cisco example of SDN architecture, they use their own openflow protocol which is called OpFlex between the controller and the managed forwarding entities in the Cisco application centric infrastructure ACI, OpFlex was modeled with an application-centric policy. With OpFlex, the devices like the switches and the firewalls still enforce the policies in Cisco ACI. ''Which provides the centralization of policy management but with distributed control, allowing automation of the entire infrastructure without limiting scalability through a centralized control point or creating a single point of catastrophic failure'' [12].

7. Virtualization & labraroy equipment

The beauty in SDN is that it enables the use of virtualization technology in it's best and most prominent ways, that's why we see Linux and some other virtualization technology pioneers provide a great deal of support for this new of technology that could aspire for whole new era of networks management and consolidated security for networks. For that you can use virtualization technologies in order to set your own lab and test-bed

environment due to it’s no-cost utilization, ease of installation and deployment not forgetting it’s effectiveness for real world networks establishments since they could be tested in the virtual environment and troubleshot before being installed in real world.
There are many tools to use but some of my favorite ones and easiest to use that can help a beginner in this field without the need to be lost in the internet search for the best tools, are GNS3 and Cisco packet tracer but of course to run SDN simulations you will need GNS3 which is GNS3 is a Free and Open Source software under GPL v3 licensing, of course you can install it on windows but in this situation you will need a virtual machine software installed on your windows like VMware, VirtualBox, etc… then you will have to install the GNS3 virtual image in order to make the GNS3 installed on windows sync with the virtual image installed on the virtual machine, this way you can run different network simulations on GNS3 besides that you can install some controllers like the HP controller on GNS3 simulator and run it, you will also need an Ubuntu container that will host the API for the controller, you can install GNS3 for windows from the link:
https://www.gns3.com/software after downloading and installing it on windows you will need to download the virtual image of GNS3. you will need to install the VM software of course. Another great tool of a massive help for this simulation is the mininet software which is widely used to design an SDN lab by simulating networks it makes virtual network devices, it also simulates both OpenFlow switches and controllers. After we test the network and make sure it works, we will use simple ways to measure the efficiency of the network, there are many software types for that purpose, for example:

- Cbench: an openflow controller benchmarker, it's a special tool used to measure the efficiency of the OpenFlow controller by creating a set of various switches that send messages of type packet-in to the controller and monitors the response and based on the reading it measures the performance and ability of the controller.
- OFLOPS: An OpenFlow switch benchmarker, it is a tool that does the opposite role of the previously mentioned tool, it measures the efficiency of the OpenFlow switches by creating virtual controller that sends messages to the switches and monitors their responses and measures the performance of the switches based on that reading.
- OFtest: it is a tool used to test the embedded OpenFlow protocol in the switches, it supports up until the 1.2 version of that protocol.

Of course, if you have an Ubuntu OS it will be much better and easier to install mininet and some controller like Open DayLight natively.

8. Some areas of research and suggestions

Of course SDN is still in its early birth stages despite that its earliest ideas could date back to the 80s but it still poses both great security threat and challenge to all researchers and developers let alone network devices manufacturers so there are now many suggestions and research ideas emerging on the surface but since this article is dedicated solely for analysis purposes and directed to those who are still in the early phase of their research and tip toeing the main features of this subject so we can suggest for them some ideas of research and from that they can go further, and continue to the end of the road.

- So unlike controller-switch communication (which is , the southbound interface),

Currently, there is no accepted standard for the control plane communication with the management plane (which is the northbound interface) and they are more likely to be implemented on an ad-hoc basis for particular applications. So it will be a great opportunity to use this area as a field of research.

- Since SDN is related to many other network technologies, it is possible to develop that field by developing SDN like active networks, IoT, ad hoc networks, neural networks.
- Since SDN is also based and related to programming then the usage of machine learning could be of a great deal of help in that matter.
- Since the architecture of SDN could be flexible and the thought of one centralized controller is a myth so it's possible of course to deploy multiple controllers to work as redundant units or to communicate with each other while controlling separate software-defined networks and for that there's a specific protocol called the east-west communication protocol which will need some research concentration to some extension.
- According to technical paper of Open DayLight Controller, the deployment of controllers, could be in different approaches there is the vertical approach which comprises a master controller that can orchestrate the configuration of each SDN domain and controller, and there is the horizontal approach where in that type of approach controllers establish a peer to peer connection and this is also called the east-west interface, it will be plausible to suggest a hybrid approach of both (figure 2).

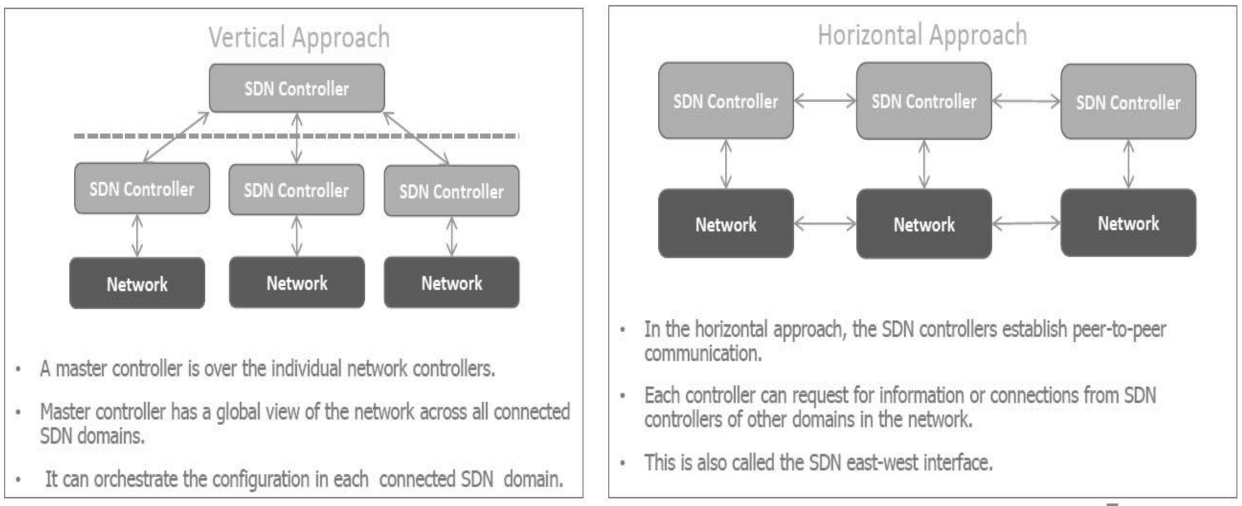
Vertical Approach
SDN Controller
SDN Controller
SDN Controller
SDN Controller
Network
Network
Network
• A master controller is over the individual network controllers.
• Master controller has a global view of the network across all connected SDN domains.
• It can orchestrate the configuration in each connected SDN domain.
Horizontal Approach
SDN Controller
SDN Controller
SDN Controller
Network
Network
Network
• In the horizontal approach, the SDN controllers establish peer-to-peer communication.
• Each controller can request for information or connections from SDN controllers of other domains in the network.
• This is also called the SDN east-west interface.

Figure 2. Vertical & horizontal approaches of SDN controller's topology.

- With the emergence of SDN technology the gap of security threats will be filled but on the other hand it poses new opportunities for new security threats and those threats could be divided to many categories like threats pointed to the management plane represented by the APIs, pointed to control plane represented by the controllers or directed to data plane (forwarding loop) represented by the forwarding switches. So it will be of a great deal of help to come up with some new algorithmic idea to consolidate the OpenFlow protocol binding the control and data planes together or a programmable interface that can protect the API itself from any kind of hacking methods like injections etc.

Because programmability is a double-edged sword SDN is exposed to more risks when it offers programmatic access to users. Imagine the case where users are forced to "trust" and depend on third party applications or standard-based solutions with the keys to the network. There is also one more situation which is where information management and control of network elements might be exploited if the isolation is note properly executed.

- Cryptocurrency is now taking the world to a new higher direction and it is coming with a soaring importance due to its mobility, ease of use and mostly due to its high security measurements especially those related to hashing which is a whole new subject but since both SDN and cryptocurrency depend on programming then it is possible to take advantage of that convergence and apply some cryptocurrency cryptography and hashing methods on the SDN field.

Conclusions

- SDN is the new trend in technology development.
- SDN is a cost, time, natural resources and energy saving solution: with SDN we can configure an enterprise network of more than 1500 switches in a push of a button according to our own requirements.
- Network technology innovations might take years to go the public real-time running, year or research, test and then deployment, for that SDN is still in its baby steps phase of life span despite that it is already installed and established by some of the major tech corporations worldwide like google, Cisco, etc… but it is still new and gives a wide horizon for both manufacturers and researchers for innovation.

References

1. https://queue.acm.org/detail.cfm?id=2560327
2. http://docplayer.net/28719982-Traditional-vs-software-defined-networking.html
3. https://en.wikipedia.org/wiki/OpenDaylight_Project
4. https://www.sdxcentral.com/projects/floodlight/
5. http://www.projectfloodlight.org/floodlight/
6. https://onosproject.org/
7. https://osrg.github.io/ryu/
8. HP Labs, HP Virtual Application Networks SDN Controller, HP technical white paper, ISSUE 2018, 6-pages.
9. Azodolmolky S., Coker O., Software Defined Networking with OpenFlow 2nd edition, Birmingham B3 2PB, UK, ISSUE 2017,344-pages.
10. https://wiki.sdn.ieee.org/display/sdn/LINC-switch
11. http://www.xentech.nl/networking/switches/pica8-p-3930.html
12. Kreutz, D., Ramos F., Verissimo P., Rothenberg C. E., Azodolmolky S., Uhlig S., Software-Defined Networking: A Comprehensive Survey, IEEE, ISSUE 2014, pages 14-76.