UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA
SCHOOL OF COMPUTER SCIENCE TECHNOLOGY

REPORT: SOFTWARE DEFINED NETWORKING

INTERNET PROGRAMMING DESIGN

SUBMITTED TO
PROF. TANG YONG

BY

YEBOAH-DUAKO ELVIS
yeboahduako770@gmail.com
(202124080120)

MAY, 2022

Abstract

Software Defined Networks are the new paradigm in computer networking. They introduce simplicity in network controls and management by decoupling the control plane from the forwarding plane thereby centralizing the intelligence of the network and adding a layer of programmability. In a similar vein, Network Function Virtualization (NFV) also abstracts the functionality of network nodes and implements them as a software in virtualized environment. This is a cost saving technology as enterprises no longer need to invest in several hardware for network services. Instead, with a single powerful computing node, they can virtualize all core network functions such as routing (routers, switches), security (firewalls, IDS), and load balancers. In this report, I introduce readers to basic underlying operations and architectures of both SDN and NFV.

Table of Contents

CHAPTER ONE: INTROCTION

1.0 Overview

Computer Networking has been a key technology in modern our modern world. By the help of this powerful technology, distance is no longer a barrier to communication. Individuals can communicate and share data from miles aways within microseconds. That is the power of Computer Networking. By definition, A computer network is a group of computers that are interconnected with one another with the sole purpose of sharing data and information. These are achieved by means of networking devices that can be classified into two broad categories: Access Network Nodes and Core Network Nodes.

Access Network Nodes: These refer to the computing devices that users use to directly access resources on the network. They are mostly referred to as host devices or end devices. Examples include Personal Computers, Laptops, Phones, and in the light of Internet of Things, virtually any device that can access the internet or be accessed over the internet.

Core Network Nodes: These are devices that mediate between the end devices. Their primary role is to manage network traffic and to ensure that data originating from one end device is delivered or routed to the right recipient or end device at the other end of the network. Examples these core network nodes include Routers, Switches, Firewalls and Load balancers.

S

Fig 1.1: A network of Computers

In this report, focus is on the core network nodes and their functionalities. The role these nodes, particular the switches and routers play can be explained by the concept of planes. Planes are dimensions or logical definitions of the core network functionalities of a given network node particularly a switch or router. There are three main network planes: the ***Data plane*** which deals mainly with the movement of data traffic across the network, the ***Control plane*** which provides control and routing for the flow of traffic across the network and the ***Management plane***: that allows network administrators to manage and set necessary policies for the network.

These planes work together to ensure that, data originating from a source is sent across the network to the right recipient. Fig. 1.2 illustrates the concept of the planes.

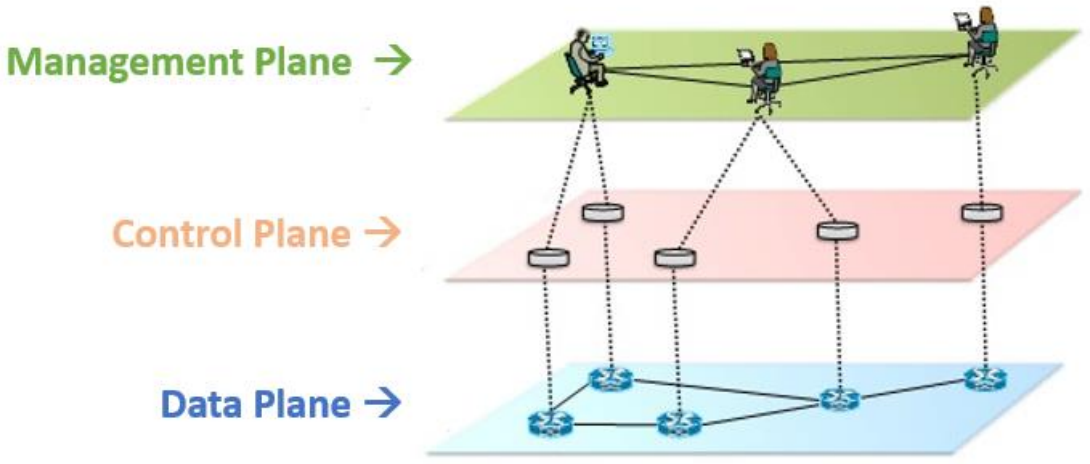


Fig 1.2: Networking Planes

1.1 The Planes in Detail

In this section, the various planes are examined in detail.

1.1.1 Data Plane (Forwarding Plane)

The data plane primarily concerns itself with the processing and delivery of packets from an ingress interface to the appropriate egress interface. It does this by inspecting the packet headers and by crosschecking its local forwarding table to make an informed decision of how to proceed with the handling of that packet. Forwarding tables are a mini database that store information on which interface of the switch or router an address (an IP address mostly) can be located. It is worth noting that, though these forwarding tables are used by the data plane, it does not create or modify this table. It relies on the Control plane for the population of the forwarding table.

The data plane assumes responsibility for packet buffering, packet scheduling, header modification, and forwarding. If an arriving data packet's header information is found in the forwarding table it may be subject to some header field modifications and then will be forwarded without any intervention of the other two planes. However, not all packets can be handled in that way, sometimes simply because their information is not yet entered into the table, or because they belong to a control protocol that must be processed by the control plane.

The diagram below illustrates how the data plane forwards data. First an ingress port receives an incoming packet from a host device. It then inspects the header information for the destination address of the packet. It checks the forwarding table if it has a mapping between the destination address and a corresponding egress interface. If there is no such entry, it confers with the control plane for action. In some cases, the packet could be flooded or even dropped. If, however, an entry for the destination address is found, then the switch makes the necessary modifications to the header of the packet and forwards it via the appropriate egress port.

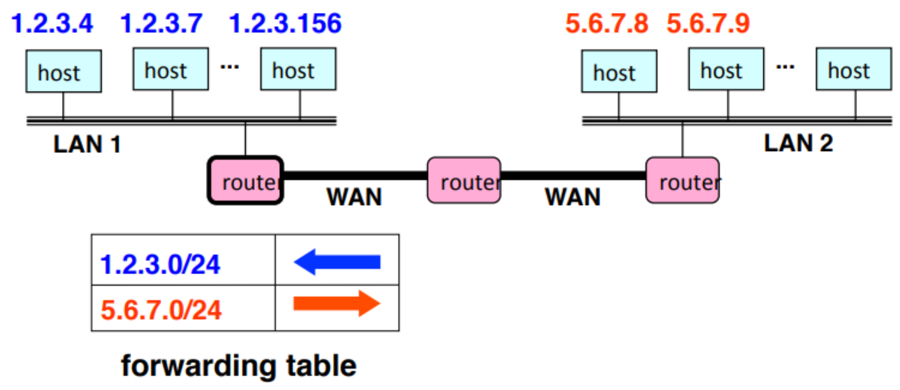


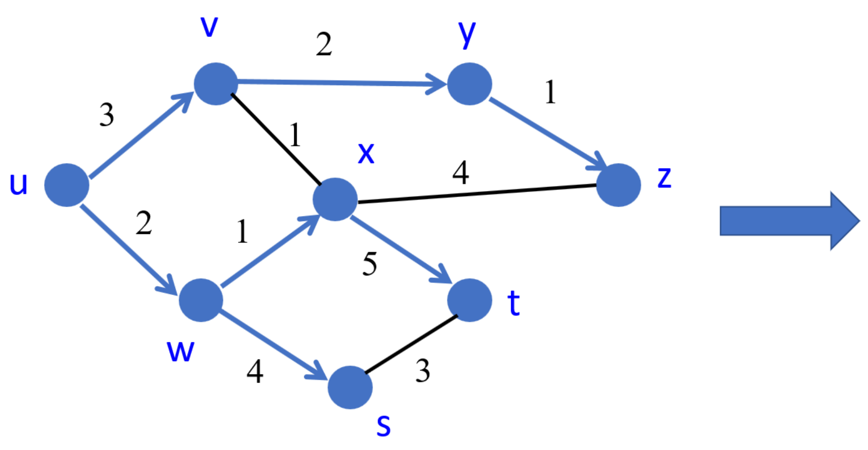
Fig 1.3: Data Plane Forwarding illustrated

1.1.2 Control Plane (Logical Plane)

The primary role of the control plane is to populate and keep the information in the forwarding table up-to-date at all times so that the data plane can independently handle as high a percentage of the traffic as possible. It does this by processing a number of different control protocols that may affect the forwarding table depending on the configuration and the type of the switch. Example of such protocol is the Open Shortest Path First (OSPF) protocol used for route discovery. These control protocols are jointly responsible for managing the active topology of the network. They are complex, and unlike the data plane that can be implemented entirely in silicon, the control plane requires special-purpose microprocessors and an accompanying software to implement. The core functions of the control plane can be summarized as follows: *Routing, Traffic Engineering* and *Failure detection and recovery.*

Algorithms for route discovery (computing paths) vary based on the various control protocol. But generally, there are two broad categories: *the Link state routing protocols (e.g., OSPF, ISIS)* and *the Distance vector Routing protocols (e.g., RIP)*

The Link state routing protocol computes routes using the Dijkstra's Algorithm. Nodes first flood the entire topology with their known paths and the associated costs. Upon reception of peer routing details, each node or router computes the shortest path to other nodes using the Dijkstra's algorithm. This algorithm is not discussed in detail but fig. 1.4 gives a fair pictorial view of how the algorithm is used. The table shows the forwarding table of node A. based on cost for each link, the control plane computes the table and it can be seen that, to move from node ***u*** to node ***s***, there are two intermediate paths, the path via node ***v*** and the path via node ***w***. However, the total cost involved by taking path through node ***v*** is either 12 (*u-v-x-t-s*) or 9 (*u-v-x-w-s*). And the cost through w is 6 (*u-w-s*) or 11 (*u-w-x-t-s*). the shortest path is the path with the lowest cost (6 for *u-w-s).* Hence, the control plane will compute the next hope to node ***s*** as node ***w***



Dest.	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

Fig 1.4: Link State routing protocol

1.1.3 Management Plane

The management plane aids network administrators to configure and monitor the switch or router. With the help of network management system tools, admins interact with the management plane to extract information from or modify data in the control and data planes as deemed appropriate. For instance, through the management plane, an admin can decide which routing protocol (OSPF, RIP, ISIS, etc.) is to be used at the control plane to populate the forwarding tables.

Another key functionality the management plane provides is Traffic engineering which primarily includes the setting of weights so as to manually influence the direction of flow of traffic within a network. These weights can be set to be either inversely proportional to the link capacity or say directly proportional to the propagation delay or relative to network-wide optimization based on traffic. However way traffic engineering is achieved, the general objectives are the same: To minimize transmission delay, to minimize traffic congestion and to maximize the use of available bandwidth, etc. Other popular functions of the management plane include the setting policies including security features such as Access Control List (ACL) configurations.

1.1.4 Interoperation of the Planes

Though the planes each have their designated roles and seem to operate independent of the other, it is worth noting that none of them is truly independent. For instance, the data plane relies heavily on the control to provide it with updated forwarding table to ensure its successful operation. The control plane needs the management plane to give it a sense of direction as to which protocols to implement so as to provide a reliable service tailored to meet user and industry demands. The planes work hand in hand and the figure below shows the interrelationship between the planes and their operation.

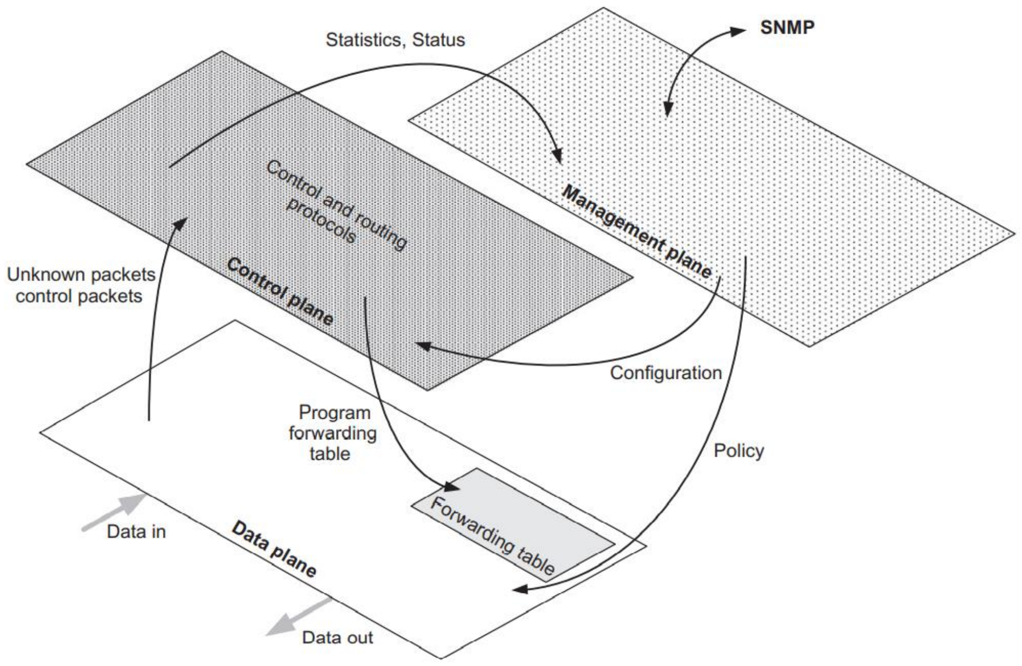
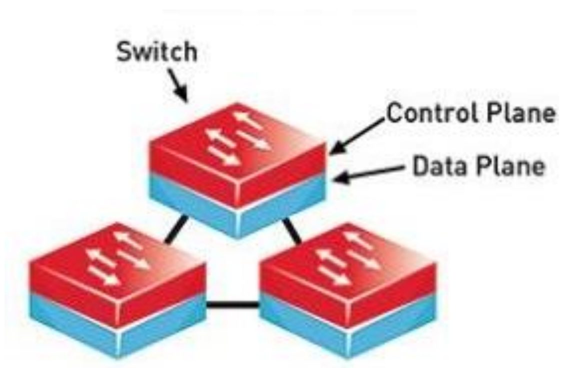


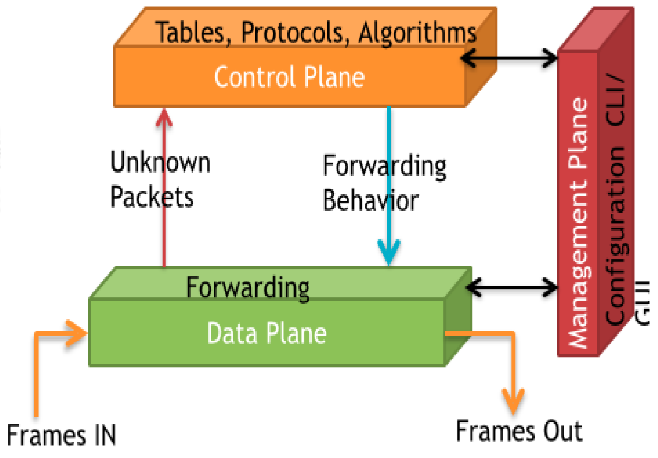
Fig 1.5: Interoperation of the network planes

CHAPTER TWO: TRADITIONAL NETWORKING APPROACH

2.0 Overview

The network planes described in chapter one are the underlying principles behind the successful implementation of the various networks across the world. In the traditional networking architecture, each switching device has all the three planes borne within it and hence a dedicated intelligence system resides in every switching device for its operation. Early network device developers and standards-creators wanted each device to perform in an autonomous and independent manner, to the greatest extent possible. This was because networks were generally small and fixed, with large shared domains. Also, the goal was to simplify rudimentary management tasks and to make the networks as plug and play as possible. Their relatively static configuration needs were performed manually. And to achieve coordination between the devices, collective decisions were made through the collaborative exchange of data between the switches.





(a) (b)

Fig 2.1: (a) Traditional Network Switch. (b) Coordination between planes in the traditional network nodes

As shown in fig 2.1(a), each individual switch has its own control plane and data plane (as well as the management plane). (b) describes the working operation for the switch. When data frames or packets are received, the data plane checks its local forwarding table to determine which egress port to send the packet to. When there is no entry, the data plane consults the control plane which details of the unknown packet and the control plane after processing the information, updates the forwarding table stating the forwarding behaviour that should be applied to the packet. The data plane then transmits the frames out through the appropriate egress port. In special cases where default routes are configured, the data plane does not consult the control plan but rather forwards the data along the default path.

2.1 Limitations of the Traditional Network Approach

Though the traditional networking approach works just fine and is able to deliver efficient networking services, there are a couple of constraints that are worth noting and addressing especially as networks keep expanding.

1. *Network Management is tedious*: As network expands to include several core network nodes, network management can become frustrating for network administrators. This is because whenever upgrades, maintenance or configuration changes need to be deployed on the network, admins are tasked to configure each device manually and worst of it all, to carry out the same commands on tens or hundreds of network nodes. How boring it gets. And if unlucky, a single bug is detected right after implementation, admins will have to rollback or update the settings on each device. This is not only tedious, but it consumes lots of time.

2. *Networks are rigid:* The switches and nodes that are shipped by vendors carry application specific integrated circuits (ASIC) and vendor-specific software that define the functionalities of the switch. This makes the network rigid in the sense that new

functionalities cannot be added to a switch. And to make up for this, admins will have to procure newer devices whenever they need to implement policies or features that are not supported by their current switches. Buying the latest switch does not necessarily guarantee rigid-free networks since technology keeps growing at an alarming rate. What is latest today could become obsolete tomorrow.

3. *Limited Network View:* The network nodes or switches have individual intelligence that makes them capable of operating independently of the other. As such, nodes compute their own forwarding tables by coordinating with each other to have a fair idea of the paths accessible via each node and the associated costs involved. In this process, large amount of data is constantly being shared between the nodes which first of all consumes computing resources and bandwidth. Secondly, the view of each node is limited by the information received by the peering node. The information may not always be the true reflection of the paths available as they are influenced by local routing policies. Hence, routing information computed may not be the best to efficiently route data. Also, traffic congestion is most likely to occur due to this limited network-wide view.

4. *Cost Intensive Implementation:* In the traditional network setting, devices are made for specific purposes. For instance, routers are needed for network-wide data routing and forwarding. Switches are needed for local routing and forwarding. To implement security, firewalls, Intrusion Detection Systems, etc. need to be procured. To optimize the network, load balancers and the likes are also required. Almost all functionalities needed to stabilize the network are implemented in different nodes. This makes our data centers flooded with several equipment which are not cost friendly.

2.2 The Way forward

Owing to the above-described problems associated with the traditional networking approach, there was the need for a novel approach that will resolve those issues by providing:

- ✓ Easy Network management such that network admins can effectively handle the work loads when configuring and managing their networks.

- ✓ Programmable networks to deal with the rigidity in the traditional networks. By so doing, new functionalities can be added to existing switches without needed to purchase new switches.

- ✓ Global Network view to provide efficient and effect routing and forwarding of traffic

- ✓ Virtualization for implementation of different device functionalities on a single hardware and to further cut down cost in procurement of bulk hardware.

CHAPTER THREE: ForCES

3.0 Overview

The challenges outlined in chapter two led researchers to explore and exploit new networking models that would address those issues. In 2003, the Internet Engineering task Force (IETF) pioneered in proposals that recommended the decoupling of the forwarding (data) plane from the control plane and providing a standard means of communication between them was key to addressing the traditional networking challenges. This led to the introduction of the Forwarding and Control Element Separation (ForCES) project by the IETF. The general objective of ForCES was to provide a simple hardware-based forwarding entities at the foundation of a network device, and software-based control elements above whose key responsibility was to provide coordination between the multiple network devices. The IETF in view of this defined four basic functional components for the implementation of ForCES:

Control Element: that would be responsible for providing the needed coordination between the individual devices on the network, and for generating and communicating forwarding and routing information for efficient traffic delivery.

Forwarding Element: that would be responsible for the enforcement of the forwarding and filtering rules provided by the control element. To improve system performance, this element was proposed to be typically implemented in hardware with little to no software dependence in its operation.

ForCES Protocol: This protocol was needed to ensure that there is efficient communication between the forwarding elements and the control elements.

Network Element: this is the actual network devices which consist of one or more Forwarding elements and Control elements.

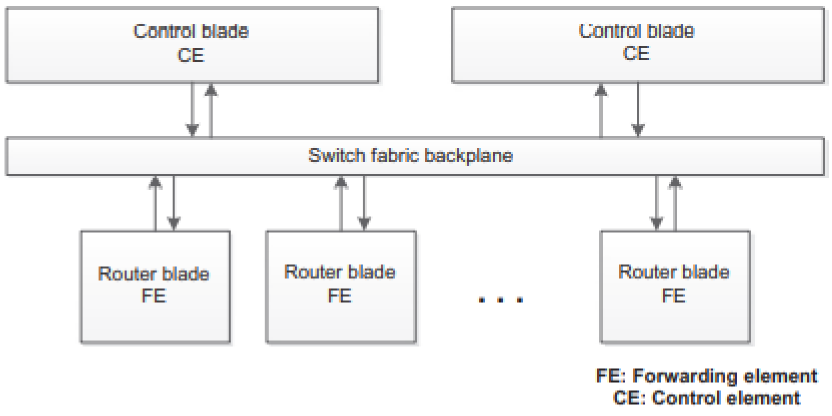


Fig 3.1: ForCES design

Based on the idea that ForCES presented, research was intensified to arrive at the best way of implementing the separation between the forwarding plane and the control plane. In the next chapter, I discuss the concept of Software-Defined Networking which operates based on the basic principles of ForCES.

CHAPTER FOUR: INTRODUCTION TO SOFTWARE DEFINED NETWORKING (SDN)

4.0 Overview

SDN is the new paradigm in the networking industry. It promises massive simplicity and higher efficiency in network management and an improved overall general performance for computer networks. It achieves these by providing a new network model that has the control plane (logic functionality) decoupled from the data plane (data forwarding). By so doing, switches are reduced to simple forwarding elements and are controlled by a centralized control element known as the controller. A single controller controls a number of forwarding elements and this makes it possible to easily manage the various network elements. With this network model, network admins are able to optimize their networks on the fly from a single source helping them to quickly respond to changes in the network usage without the need to manually reconfigure every single node in the network.

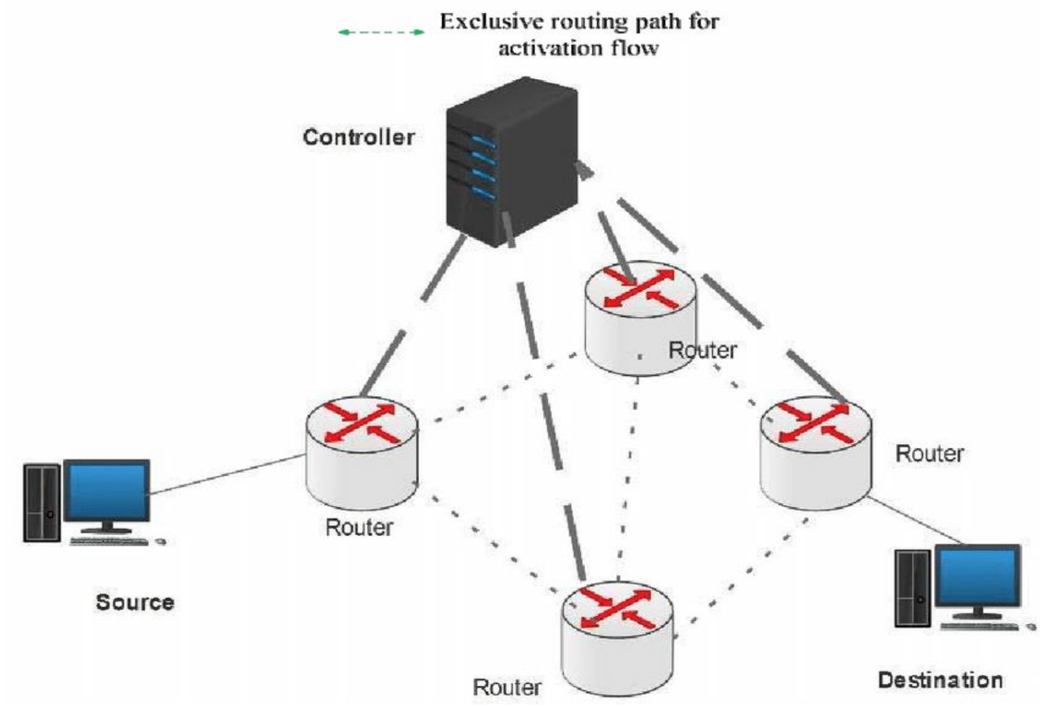


Fig 4.1: Structure of the SDN Model

4.1 SDN Architecture

SDN in addition to the traditional networking planes, introduces a new layer or plane known as the application layer. This layer provides network programmability functionalities thereby making the network more flexible and easier to automate processes. With this introduction, the management plane is implemented as a sub-plane of the application plane. Also, two main Application Programming interfaces (the Northbound and Southbound) are defined to allow communication between the layers. Fig. 4.2 shows the general architecture of the SDN model.

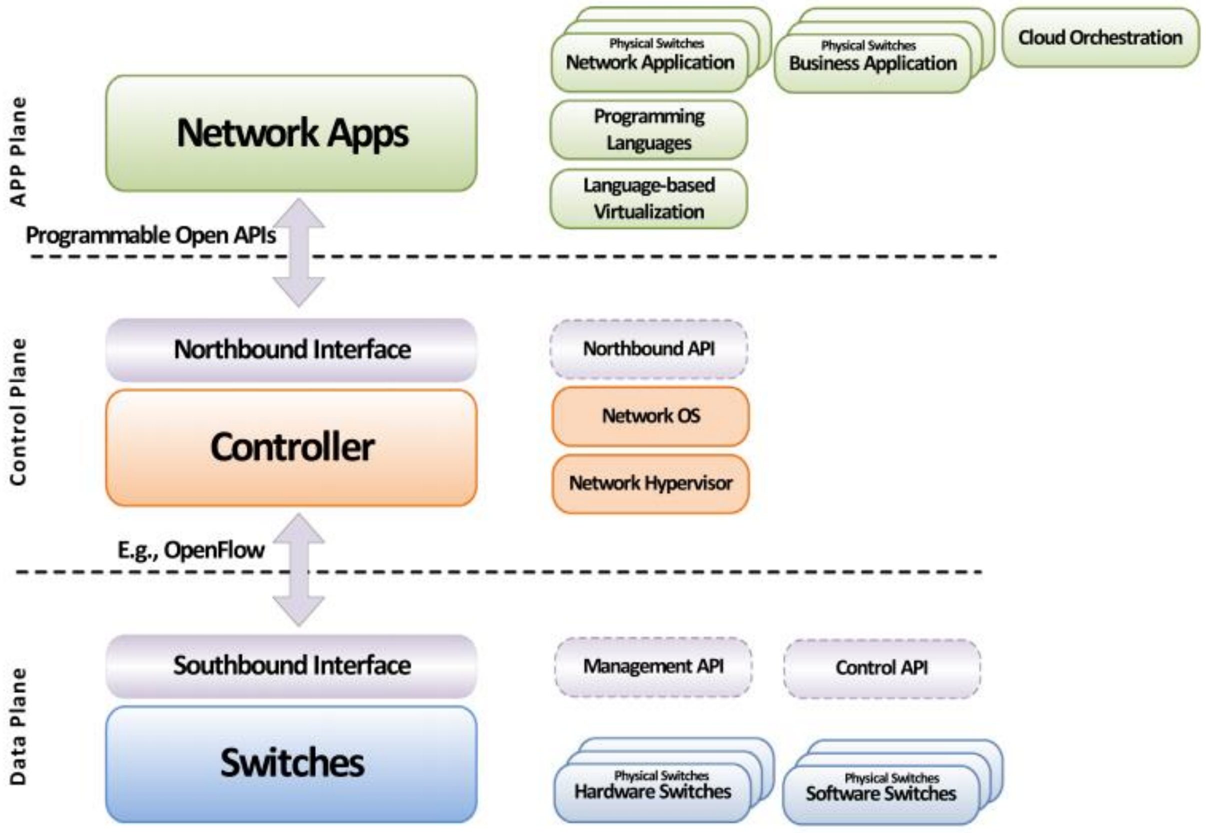


Fig 4.2: SDN Architecture

4.2 Breaking down the Architecture

In this section, the roles of each element in the SDN architecture are analyzed. The basic architecture elements in this new model include:

4.2.1 Application Programming Interfaces (APIs)

APIs are intermediary software that allow the planes to interact with each other in a meaningful manner. There are two main types of APIs in SDN:

Northbound Interface: The northbound interface is used to access the SDN controller via the application plane. It provides the link between the running network applications and the SDN controller. The applications tell the network what they need (data, storage, bandwidth, and so on) via this interface and the network can deliver those resources, or communicate what it has. Network applications that can be optimized via the northbound interface include load balancers, firewalls, other security services and cloud resources, etc.

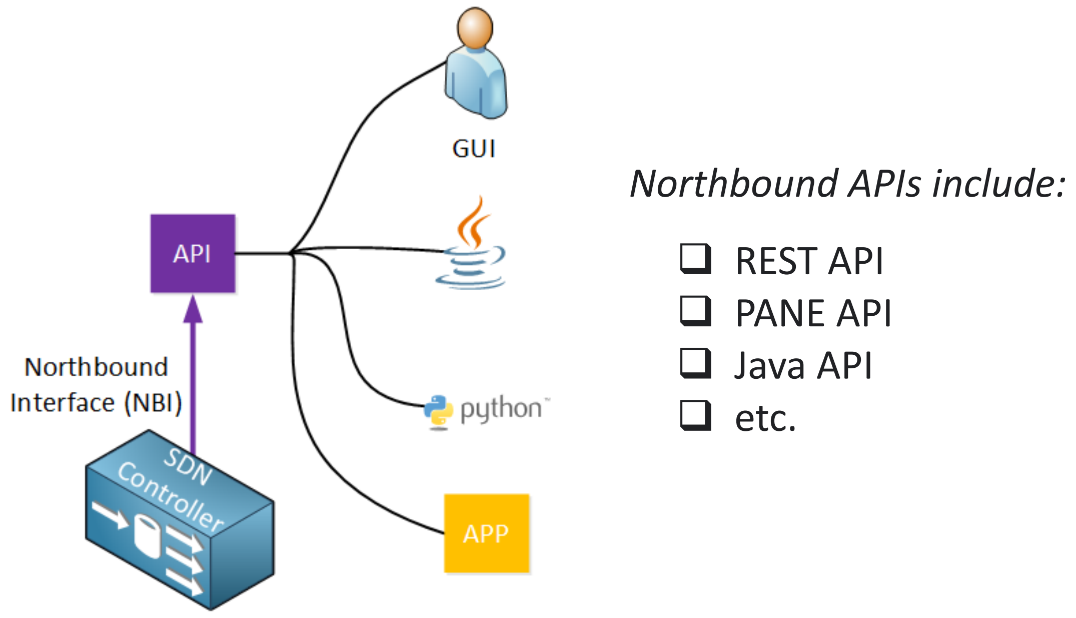


Fig 4.3: The Northbound Interface

***Southbound Interface**:* It provides the interface through which the Control plane interacts with the forwarding plane. This interface allows the controller to communicate flow rules to the switches, and the switches in turn, to query flow instructions from the controller. Several APIs exist to ensure the efficient delivery of flows. They include OpenFlow, Cisco OpFlex, OpenDaylight, etc.

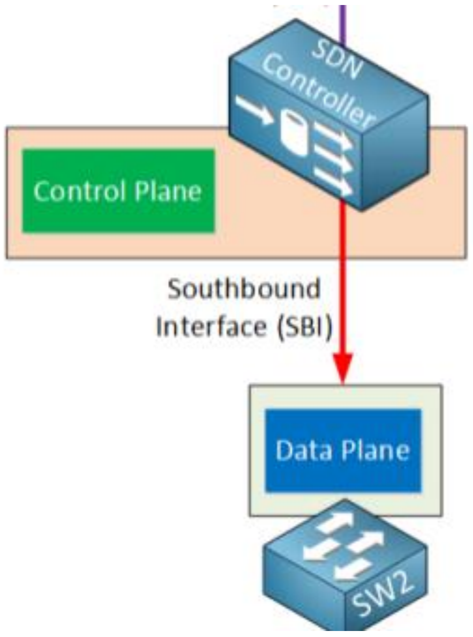


Fig. 4.4: The Southbound Interface

4.2.2 Application plane

The application plane equips the network with great programmability and automation functionalities. By so doing, network admins are given high level of control over their networks. Through the northbound interface, scripts and programs can be written to run on the controller to dictate the anticipated behavior of the network and to further optimize the networking functions. Moreover, the management plane is further enhanced and made easier with a rich Graphical User Interface (GUI) or Command Line Interface (CLI) that are designed to effectively assist in the monitoring, management and configuration of the various network parameters as deemed fit.

In addition, the applications can build an abstracted view of the network by collecting information from the controller for decision-making purposes. These applications could include networking management, analytics, or business applications used to run large data centers. For example, an analytics application might be built to recognize suspicious network activity for security purposes.

Finally, where a traditional network would use a specialized device, such as a firewall or load balancer, a software-defined network replaces the appliance with an application that uses the controller to manage data plane behavior in a similar fashion as would the specialized hardware.

4.2.3 Control Plane

The SDN control plane, just like the control plane in the traditional model, functions as the logic center for the network. Only with enhanced features and capabilities to make it more efficient. In the SDN model, the control element is powered with a Network Operating system, network hypervisors and control programs that aid the controller in making informed decisions on the network-wide mode of operation.

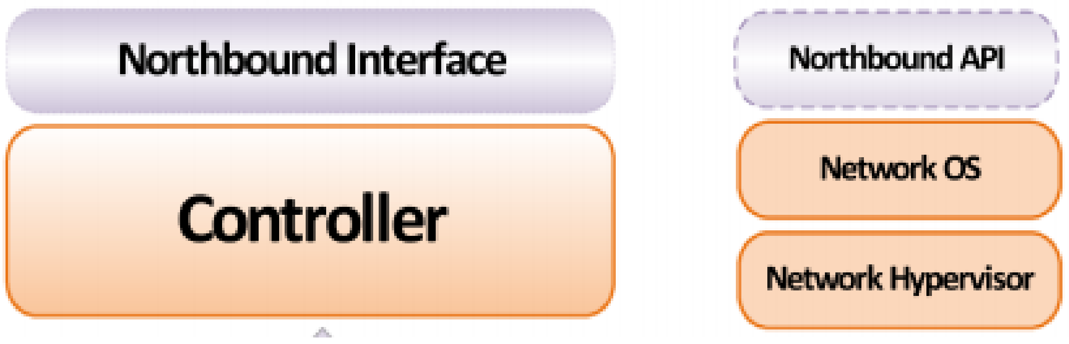


Fig. 4.5: The Control plane elements

The Controller (Control element) in an SDN is a dedicated device or server. It serves as the powerhouse and brain behind the computation of flow rules, and the operation of the network. It is the logical entity that receives instructions or requirements from the SDN application layer and relays them to the networking components. The controller also extracts information about the network from the hardware devices and communicates back to the SDN applications with an abstract view of the network, including statistics and events about what is happening. Also, it performs the computation of flow rules for each forwarding element within its domain and updates their flow tables intermittently.

The Network operating System (NOS) is a special purpose distributed system that provides the platform on which the controller works. It creates a consistent, up-to-date network view and makes provision for the various APIs to allow flow transfers across the controller and the other planes. Examples of NOS in the industry include *NOX, ONIX, OpenDaylight*, *Beacon,* etc. The NOS uses forward abstraction to obtain the state information from the forwarding elements and to also give control directives to the forwarding elements.

Control Program are applications that run on top of the NOS. They take the global network view (graph/database) as input and based on the written scripts, they compute and configure each network device by generating switch specific Flows which are then forwarded to the corresponding flow switches for their operations.

The Network Hypervisors enable virtualization on the SDN controller. Resources such as CPU, memory and storage can be maximized and made available for use by the various virtual instances or VMs that run on the controller.

4.2.4 Data Plane (Forwarding Plane)

The data plane is also referred to as the infrastructure layer in SDN. Theis is because it houses the forwarding elements (Flow Switches). These switches contain flow tables which are populated by the SDN controller through the Southbound interface. Based on the Flow tables, the data plane is able to achieve data forwarding. Data forwarding done in this regard is referred to as Flow-based forwarding and not destination-based forwarding as in the traditional networks. When there are no flows corresponding to the header information contained in a received packet, the Controller is notified to make available a flow direction for that packet. After which an action is taken based on the received flow the controller sends. The most prominent Southbound API used is the OpenFlow. Open Flow would be discussed in detail in this report in follow up chapters.

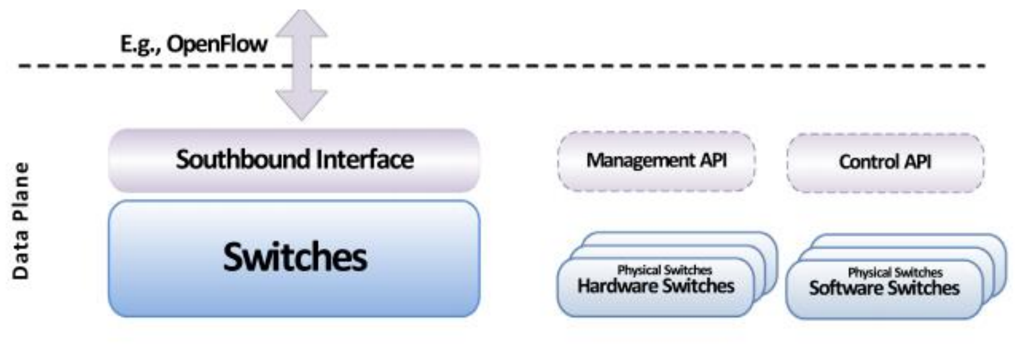


Fig 4.6: Data Plane

To allow a gradual and smooth transition from the traditional network architecture to that of the SDN, hybrid switches were needed to ensure interoperability. More on this in later sections.

4.3 Putting the Pieces together: SDN Mode of Operation

The best way to appreciate the working functionality is to consider the SDN architecture from the bottom up starting with the forwarding elements (switches). Consider figure 4.7 Below to appreciate the follow up explanation. The control plane and the application plane are housed in a dedicated high-performance machine.

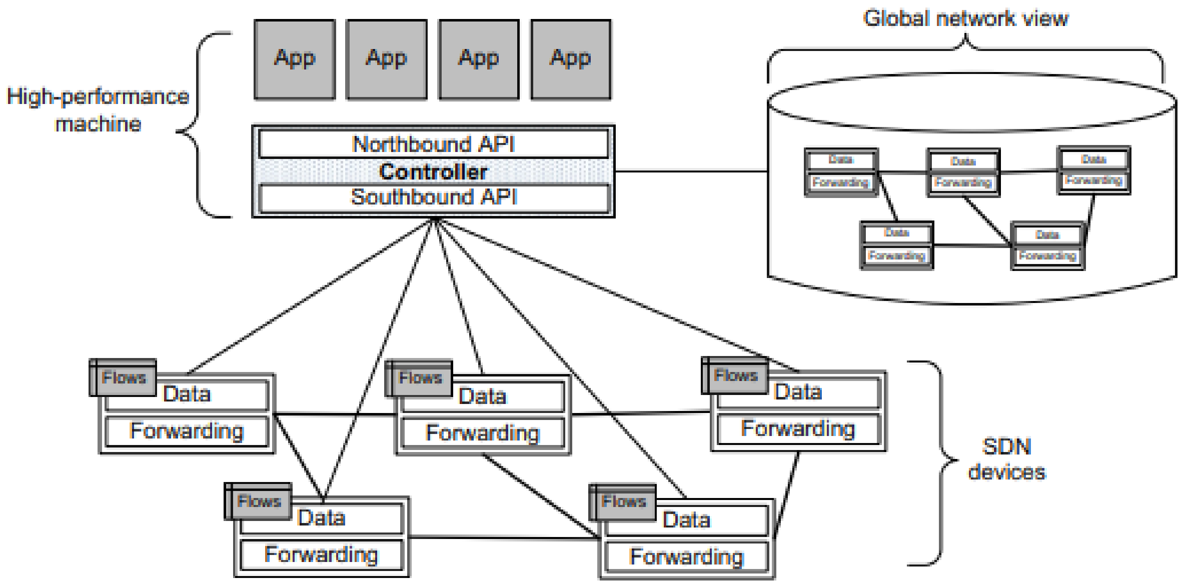


Fig 4.7: Overview of SDN Operation

The SDN devices refers to the flow switches which carry out the forwarding functionality and decides the necessary action to apply to each incoming packet. An action can be to *forward, drop, consume* or *replicate* the incoming packet. It may also transform the packet into some manner before taking further action. These actions are taken based on flow data which are defined by the controller as depicted in the upper left portion of each device.

A flow constitutes a set of packets transferred from one network endpoint to another endpoint (or set of endpoints). The endpoints may be defined as IP address-TCP/UDP port pairs, VLAN endpoints, layer three tunnel endpoints, and input port, among other things. One set of rules describes the actions that the device should take for all packets belonging to a particular flow. Flows are unidirectional in that packets flowing between the same two endpoints in the opposite direction could each constitute separate flows. Flows are represented on each SDN device as a flow entry which are stored in a flow table. The flow table contains the flows and the actions to be performed when a packet matching that flow is received by the forwarding element.

When a packet arrives at the SDN device, the forwarding element consults the flow table in search of a match. In the event that a flow is matched, it applied the necessary action (mostly forwarding) to the packet. If, however, it does not find a match, the switch can either drop the packet or pass it to the controller seeking further directives on how to handle the packer, depending on the version of OpenFlow and the configuration of the switch. Also, a packet may be dropped due to buffer overflow.

When the forwarding elements pass unmatched packets to the controller, the controller is tasked to generate a flow entry and download into the flow entry of the SDN devices. It does this by abstracting the network of the forwarding elements within its control domain and presenting these network resources as a vivid abstraction to the SDN applications configured to be running above. The SDN applications then define the flows based on the configured protocols and passes them back to the controller to be delivered to the flow switches.

As shown on the right-hand side of the controller that maintains a view of the entire network that it controls. This permits it to calculate optimal forwarding solutions for the network in a deterministic, predictable manner. Since one controller can control a large number of network devices, these calculations are normally performed on a high-performance machine, with an order-of-magnitude performance advantage over the CPU and memory capacity than is typically afforded to the network devices themselves. For example, a controller might be implemented on an eight-core two-GHz CPU versus the single-core one-GHz CPU more typical on a switch.

The SDN application interfaces with the controller, using it to set proactive flows on the devices and to receive packets which have been forwarded to the controller. Proactive flows are established by the application; typically, the application will set these flows when the application starts up, and the flows will persist until some configuration change is made. This kind of proactive flow is known as a static flow. Another kind of proactive flow is where the controller decides to modify a flow based on the traffic load currently being driven through a network device.

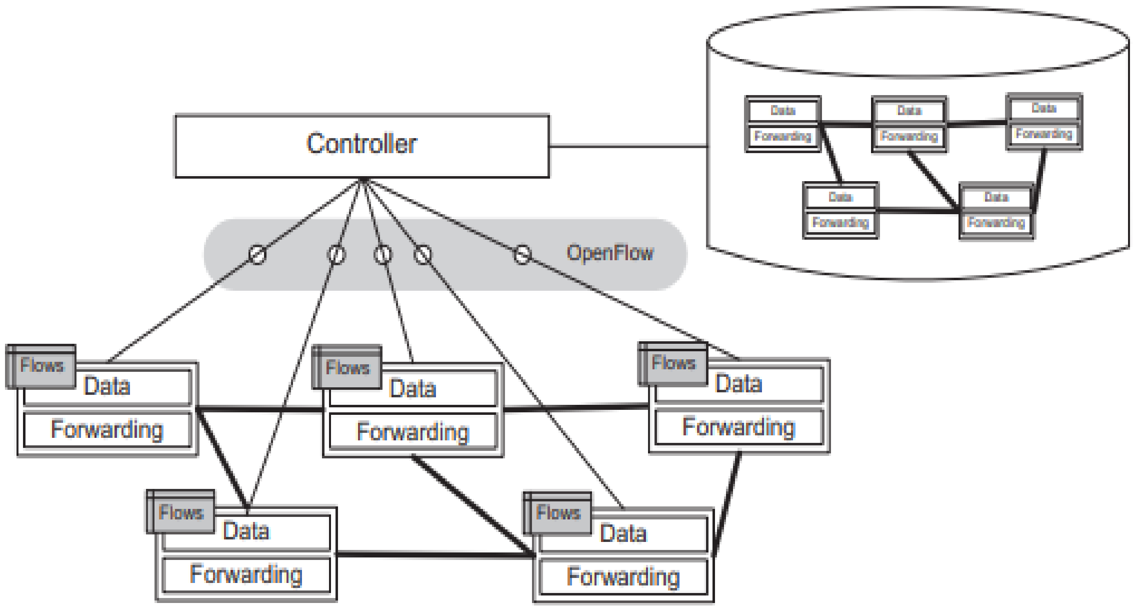


Fig. 4.8: Controller to device communication

In addition to flows defined proactively by the application, some flows are defined in response to a packet forwarded to the controller. Upon receipt of incoming packets that have been forwarded to the controller, the SDN application will instruct the controller as to how to respond to the packet, and, if appropriate, will establish new flows on the device in order to allow that device to respond locally the next time it sees a packet belonging to that flow. Such flows are called reactive flows. In this way, it is now possible to write software applications which implement forwarding, routing, overlay, multipath, and access control functions, among others.

There are also reactive flows that are defined or modified as a result of stimuli from sources other than packets from the controller. For example, the controller can insert flows reactively in response to other data sources such as Intrusion Detection Systems (IDS) or the NetFlow traffic analyzer. Figure 4.8 depicts the OpenFlow protocol as the means of communication between the controller and the device. OpenFlow is the defined standard for such communication in Open SDN and would be discussed in subsequent chapters.

CHAPTER FIVE: INTRODUCTION TO OPENFLOW

5.0 Overview

SDN and OpenFlow are mostly asserted by casual networking enthusiasts to be the same thing. Well, they are not. OpenFlow is a distinct subset of the technologies included under the big tent of the SDN technology. It is the only nonproprietary, general-purpose technology that provides SDN solutions to the network industry. Its specifications are managed by the Open Network Foundation (ONF) which was formed in 2011 by giant companies like Google, Facebook, Microsoft, etc. for the express purpose of accelerating the delivery and commercialization of SDN.

Over the years, OpenFlow has been established as a very important tool and catalyst for innovation. It defines the communication protocol between the SDN data plane and the SDN Control Plane. It further defines a part of the behaviour of the data plane. The control plane behaviour is, however, independent. It relies on OpenFlow merely for communication to the data plane.

A fully implemented OpenFlow system consists of an *OpenFlow controller* that communicates with one or more *OpenFlow Switches*. *The OpenFlow protocol* defines the specific messages and message formats exchanged between controller (control plane) and device (data plane). The OpenFlow behavior specifies how the device should react in various situations, and how it should respond to commands from the controller.

In summary, OpenFlow is an open-source programmable network protocol designed to manage and direct the flow of traffic among switches in an SDN environment. It allows an OpenFlow controller to instruct OpenFlow switches on how to handle incoming packets.

5.1 The OpenFlow Switch

There are two types of OpenFlow switches in the industry. The OpenFlow Only switch and the Hybrid OpenFlow Switch. The OpenFlow only switch is the real implementation of the ideal SDN forwarding element. It has no local intelligence and relies solely on an external controller to fully populate it flow tables for its operations. The Hybrid OpenFlow switch is a traditional switch that is capable of performing SDN operations. It has a local intelligence (a control plane) and an OpenFlow agent that links the switch to a controller. As such, the hybrid switch contains both forwarding tables and flow tables. In an SDN enable environment, the switch integrates well and operates as a forwarding element and in a traditional setting, it relies on its local intelligence to route and forward traffic. Hybrid switches require a preprocessing classification mechanism that directs packets to either OpenFlow processing or the traditional packet processing. This is a good design as interoperability is necessary for the gradual migration from the traditional networking model to the software-defined networking model.

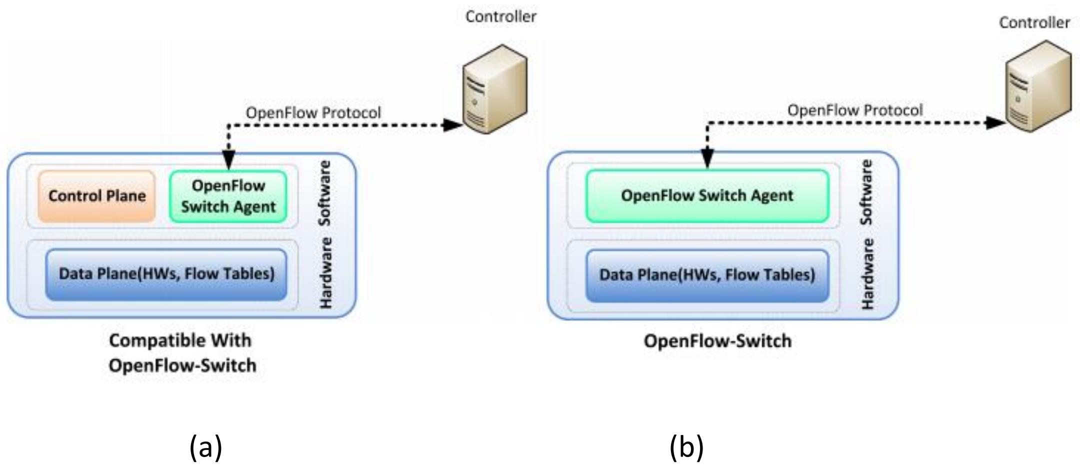


Fig. 5.1: (a) Hybrid OpenFlow Switch. (b) OpenFlow Only Switch

5.1.1 Operation of the OpenFlow Switch

For the purpose of this explanation, the OpenFlow only switch would be considered. Fig. 5.2 Below illustrates the basic functions of the OpenFlow switch and its relationship to an OpenFlow controller. The core function of the OpenFlow switch is to determine which egress port it should forward a received packet. In the figure, the OpenFlow switch has N ports. It receives a packet from port 2. Using the packet-matching function, the switch performs an exhaustive search of its flow table (*the table adjacent the packet matching function)* for a flow match. The wide, grey, double arrow initiates a decision logic, shows a match if any and directs the matched packet to the *action box* on the right where three fundamental options for the disposition of the packet can be performed:

Path A implies that a flow was found for the packet and hence the packet is to be forwarded out via a corresponding local port. Header modifications are performed, if necessary, before the packet exits the egress port.

Path B implies that the packet be dropped: Reasons could be that, that is the action dictated by the matching flow entry for the packet. Or rarely, but possibly, the buffer is full and hence, there is no temporary storage place to keep the packet before it gets processed.

Path C implies that no matching was found and hence, the packet needs to be passed to the controller to generate the required flow entry before further action can be performed.

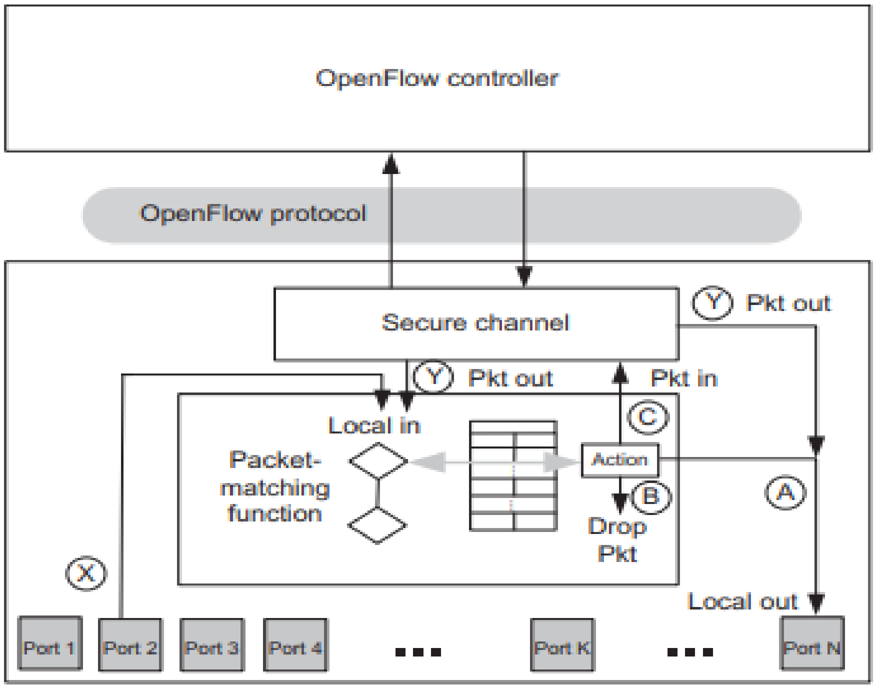


Fig. 5.2: OpenFlow Switch operation

When the Controller completes the generation of a flow for the received packet, it pushes it out via two channels both labelled as ***Y*** in the diagram. In the rightmost case, the controller directly specifies the output port and the packet is passed to that port N as shown. In the leftmost path Y, the controller returns the packet, together with the flow entry, to the OpenFlow switch. The switch updates its flow table and subjects the packet to the packet-matching function again to determine the necessary action to carry out on the packet.

5.2 The OpenFlow Controller

As discussed already, the controller is the power hub or the processing center for the entire SDN network. The OpenFlow Controller serves the exact same purpose. It programs data plane elements with the standard OpenFlow language. It computes the packet matching and forwarding rules that the OpenFlow switches use. Any changes that result in recomputing routes will be programmed onto the switch by the controller. It also interfaces with the application layer to add more programmability features to the network.

5.3 The OpenFlow Protocol

The OpenFlow protocol primarily resides on the Southbound interface. It defines the communication between an OpenFlow controller and an OpenFlow switch. The protocol includes a list of messages that can be sent between the controller and the switch. These messages allow the controller to program the switch in order to ensure maximum control of the switching of user traffic. The most basic programming *defines, modifies* and *deletes* flows. When the controller defines a flow, it is providing the switch with the information it needs to know how to treat incoming packets that match that flow. The possibilities for treatment have grown more complex as the OpenFlow protocol has evolved, but the most basic prescriptions for treatment of an incoming packet are denoted by paths A, B, and C that were described under the operations of the OpenFlow switch. Table 5.1 outlines the message types used by openflow for communication along the southbound interface.

Table 5.1 OFPT Message Types in OpenFlow 1.0

Message Type	Category	Subcategory
HELLO	Symmetric	Immutable
ECHO_REQUEST	Symmetric	Immutable
ECHO_REPLY	Symmetric	Immutable
VENDOR	Symmetric	Immutable
FEATURES_REQUEST	Controller-Switch	Switch Configuration
FEATURES_REPLY	Controller-Switch	Switch Configuration
GET_CONFIG_REQUEST	Controller-Switch	Switch Configuration
GET_CONFIG_REPLY	Controller-Switch	Switch Configuration
SET_CONFIG	Controller-Switch	Switch Configuration
PACKET_IN	Async	NA
FLOW_REMOVED	Async	NA
PORT_STATUS	Async	NA
ERROR	Async	NA
PACKET_OUT	Controller-Switch	Cmd from controller
FLOW_MOD	Controller-Switch	Cmd from controller
PORT_MOD	Controller-Switch	Cmd from controller
STATS_REQUEST	Controller-Switch	Statistics
STATS_REPLY	Controller-Switch	Statistics
BARRIER_REQUEST	Controller-Switch	Barrier
BARRIER_REPLY	Controller-Switch	Barrier
QUEUE_GET_CONFIG_REQUEST	Controller-Switch	Queue configuration
QUEUE_GET_CONFIG_REPLY	Controller-Switch	Queue configuration

CHAPTER 6: PERFORMANCE OF THE SDN

6.0 Overview

To evaluate the performance of SDN, first, I access how SDN addresses the very key issues in the traditional networking architecture that prompted the need for a revolution in our way of implementing networks. These key issues from chapter 2 can be summarized as:

- ➢ Network Management is Tedious
- ➢ Networks are Rigid
- ➢ Limited Network view
- ➢ Cost Intensive Implementation

Secondly, I look at a couple of added benefits SDN introduces and the finally I evaluate the weakness that SDN introduces.

6.1 Does SDN Address the issues in the Traditional Network

SDN and its related research works were all as a result of our desire to simplify our networks and address the challenges that the traditional network setting posed to both our networks and to our network administrators.

6.1.1 Network Management is Tedious

With the traditional network approach, we saw that network admins were burdened with the task of manually configuring each individual core network node due to the distributed intelligence system. In SDN, with the decoupling of the control plane from the forwarding plane and the introduction of the application plane, network admins are now relieved of much stress. With access to and control over the centralized control plane and the rich applications available, network admins can now configure hundreds of switches from a single point of access on the network. Hence, the issue of network management being tedious has been ***resolved*** by SDN. Networks are much simplified in terms of management and configurations.

6.1.2 Networks are rigid

The rigidity of the traditional networks had to do with the lack of programmability. With ASIC and vendor specific software, it was nearly impossible to add new functionalities to a core node aside that which had been shipped by the vendor. With the introduction of the application plane and the corresponding APIs, SDN networks are very flexible. New functionalities can be deployed by writing the scripts and mini-programs that would be run on the controller which is a high-performance device as compared to the traditional networking nodes. Hence, SDN has brought about much flexibility.

6.1.3 Limited Network View

In the traditional network architecture, each device relied on data presented by their peering nodes in order to have a fair idea of the entire network. This affected the routing process. However, with SDN, the centralized controller has a global view of the entire network. This makes routing better optimized as flows are generated with a global knowledge of the network.

6.1.4 Cost Intensive Implementation

To achieve an efficient and secure network, several network nodes needed to be procured. These include Routers, Switches, Load Balancers, Firewalls, Intrusion Detection Systems, etc. All these come a high cost. With SDN, using the application plane and virtualization technology, one powerful high-performance computing device could be used to realize all these device roles thereby cutting down cost. The key technology that realizes this is the Network Function Virtualization which will be discussed briefly shortly.

In the light of these, all 4 critical issues that were raised concerns in the traditional networking architecture have been resolved. Now I discuss other added benefits the SDN brings on board.

6.3 Other Added Benefits of SDN

Improved Security

Security is well managed in the SDN network. With a centralized intelligence system, admins are able to maintain focus on the controller(s) and implement strict security policies to fend off all possible security threats. In the traditional network, since each node has its own intelligence, admins must maintain a broad level of security awareness as a single compromised node in the network domain could be used to create a lot of harm. For instance, with regards to the spanning tree protocol (STP), by altering the switch priority of a switch, an attacker can alter completely the paths taken by packets to favour a path that they have maximum control over. In SDN, all such controls are left to the controller, and hence no injected switching node can be used to implement such an attack.

Improved speeds

By making the switches mere forwarding elements, the speed of the network is improved. The use of network bandwidth and resources for exchanging bulk protocol information as done in the traditional networks are all massively reduced to simple switch-controller communications. With the right flow entries, switches are able to maximize their system resources for efficient forwarding at the line rates.

6.3 Challenges of the SDN

Security

Since the controller plays such a central role in the SDN architecture, security strategies must focus on protecting the controller and authenticating an application's access to the control plane. While this is great as security can be contained and implemented mainly from a limited scope of the controller, the controller presents a single point of failure in the network. Should the controller fail, the entire network operations are greatly affected. This is one key problem in the SDN networks. It can be addressed by using backup controllers and by the use of other proposed SDN architectures which will be discussed in later sections.

Also, new services can introduce security threats as programmers and network administrators may unwittingly introduce buggy scripts or codes that could make the controller or network vulnerable.

Scalability

Since the SDN architecture includes centralized interfacing with data planes on multiple devices, the possibility exists for the controllers to become a network bottleneck. In particular, large networks with volumes of networking requests can overwhelm controllers. As networks grow, the bottleneck tightens and network performance degrades.

Interoperability

For new networks, implementing SDN is fairly straightforward -- all network devices are SDN-ready. Transitioning a legacy network to SDN is another story as the legacy network is likely supporting active business and networking systems. Enterprises and most networking environments have to transition to SDN, requiring a period of interoperability with a hybrid legacy-SDN infrastructure.

SDN and legacy network nodes can operate together, with help from an appropriate protocol that supports SDN communications while providing backward compatibility with existing IP and

MPLS control plane technologies - reducing the cost, risk, and disruption of services while transitioning to SDN.

Performance

Performance is the greatest issue for all networks. Regardless of how robust, secure, scalable, or interoperable a network is, it is unusable if it lacks performance. The separate control and data plane architecture can introduce latency into SDN. In large networks this can build to an unacceptable level of delay, degrading network performance. Respective controller response time and throughput can contribute to poor performance, with the combined effect causing scalability issues.

The solution for many performance issues in large and growing networks is to push more intelligence to the data plane or move to a distributed control plane architecture of some type. While this can improve SDN performance, it's moving somewhat away from the intent of SDN and replicating traditional networks built on fully distributed intelligent devices. A balance has to be sought where virtualization is maintained without degrading network performance or introducing potential single points of failure.

6.4 Remedying the SDN Challenges

To alleviate the problems listed for SDN, especially the issues of single point of failure, scalability and performance, three sub architectures of SDN have been proposed: *Centralized Controller, the Decentralized Controller* and *the Logically Centralized, Physically Distributed (LCPD) architectures.*

6.4.1 Centralized Controller Architecture

This is the original proposed SDN architecture that has been discussed so far. The only slight difference is the introduction of a standby or backup controller that takes control in the event that the active controller is unavailable. The standby controller syncs with the active controller at all times and uses the principle of *heartbeat* to know the status of the active controller. Also, all flow switches are connected to both the active controller and the standby controller. However, only one set of links are active at any point in time.

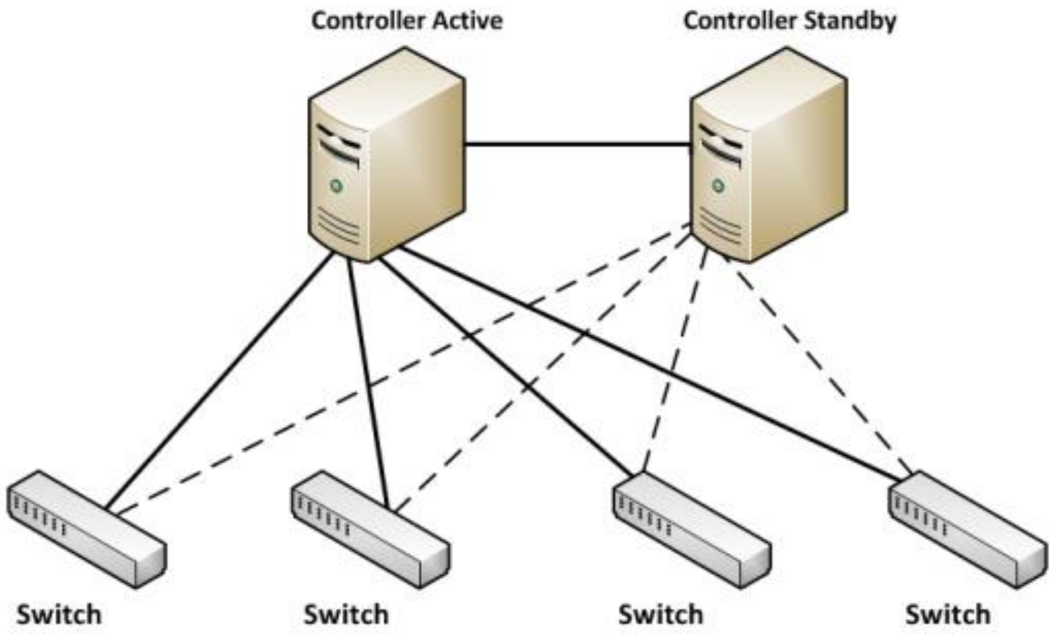


Fig. 6.1: Centralized Controller with Standby Architecture

Advantages of the Centralized Architecture

- ✓ *Security issue is improved.*

  System Availability is improved. With a standby controller, the risk of unavailable associated with the Single point of failure is reduced.

Disadvantages of the Centralized Architecture

- ✓ *Scalability issues not resolved*

  The standby controller does not function when the active controller is available and working. Hence, the possibility of bottleneck with the controller with regards to increasing traffic still persists thereby degrading the quality of service (QoS).

- ✓ *Added cost for standby controller*

  The additional controller comes at a cost. This cost spans beyond financial expenses on the purchase of the standby controller. Viable ports which could be used to ease traffic congestion and also serve as point of network extensions are used to connect to the standby controller.

6.4.2 Distributed Controller Architecture

In this architecture, the network is segmented into a number of domains. Each domain is controlled by a dedicated controller. Though this limits the global view of the network as defined in the centralized controller architecture, the controllers in this distributed architecture are connected together to exchange data about their respective subnetwork. Thereby, providing a certain level of network wide view to aid in the efficient operation of the SDN network.

Advantages of the Distributed Controller Architecture

- ✓ *Security: Single point of Failure Resolved*

  Due to the segmentation of the network and the distribution of the controllers across the various domains, failures are isolated to only a section of the network. If a controller in domain A should fail, only switches and traffic handled by that domain will be affected. Other network segments work.

- ✓ *Scalability Improved*

  Due to the distribution of switches over several network controllers, the traffic loads of the network are also distributed. The problems of bottleneck with increasing traffic are catered for since traffic is now handled by a number of controllers that have adequate computing resources at their disposal. This improves the overall quality of service of the SDN network.

Disadvantages of the Distributed Controller Architecture

- ✓ *Security: Availability*

  Failure of a controller isolates a section of the network. User traffic that needs to be routed through the domain or the paths controlled by the unavailable controller are affected thereby degrading the quality of service of the system.

- ✓ *Scalability not entirely resolved*

  There remains the possibility of an uneven distribution of loads across the controllers. One controller within a certain domain could be burdened with much traffic leading to bottleneck related issues whereas the other controllers could be lying idle. This leads to inefficient resource management and a degradation in the quality of service as well.

- ✓ *Implementation cost high*

  This architecture requires several separate controllers which operate independently as though they are different networks of their own. This incurs extra cost in the procurement and implementation of the architecture.

- ✓ *Constant Sync of Large data*

  There is a constant synchronization of data between the controllers. This consumes system's computing resources.

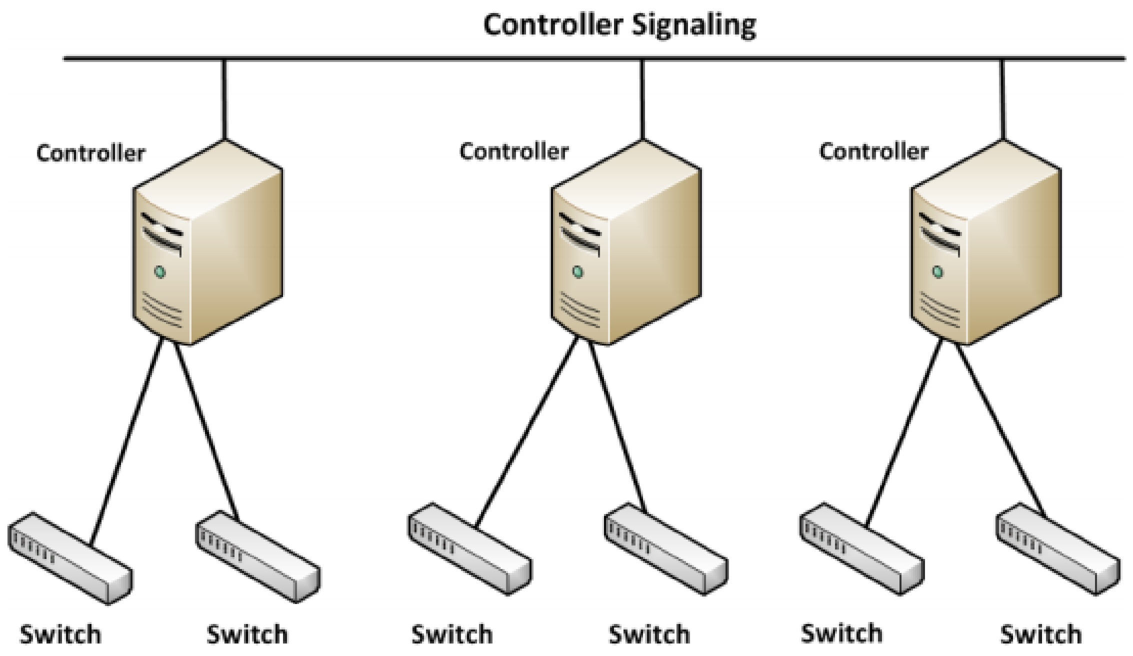


Fig 6.2: Decentralized Controller Architecture.

6.4.3 Logically Centralized Physically Distributed Architecture

This architecture stands out to be the best of the three SDN architectures. There are several physical controllers installed in the network and distributed over the switches thereby depicting the scenario of the distributed controller architecture. However, these controllers and the switches are engineered to work together as though they are all connected to a single controller. Hence, the term logically centralized. Switches in each domain or segment connect to an *Access Point* (a device that serves aggregation purpose). The access point of each domain connects to all the controllers within the network. However, each has a dedicated controller that serves as the active controller of the domain the access point connects to. The other controllers serve as backup controllers. With this setup, each switch has a global view of the entire network.

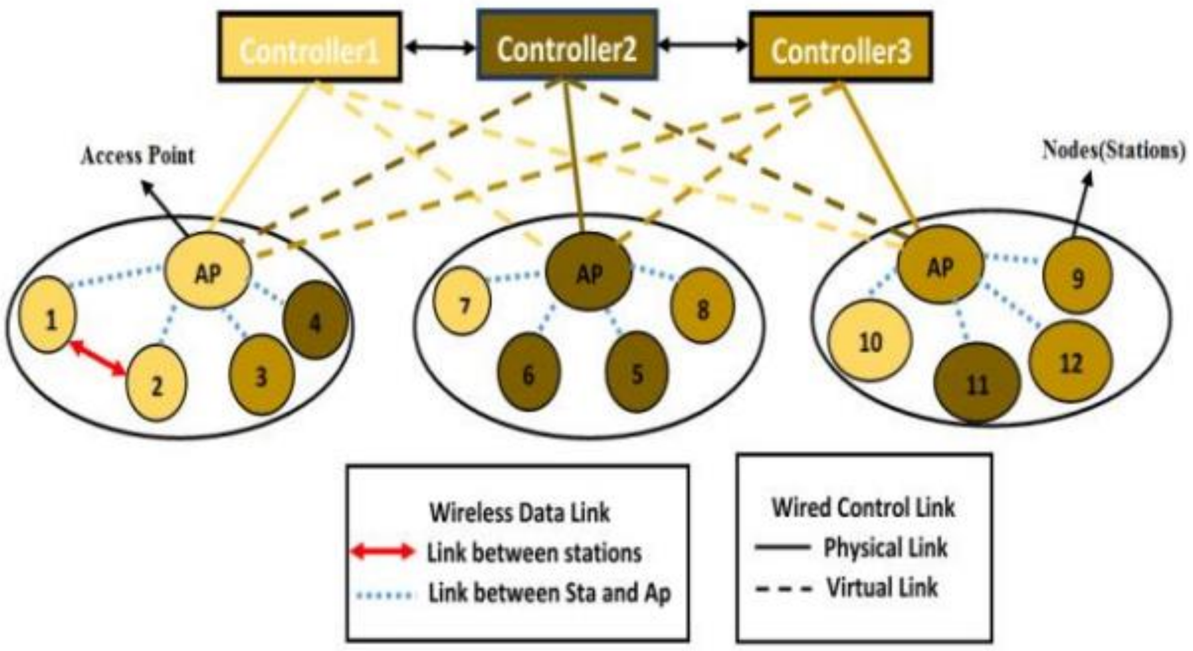


Fig. 6.3: Logically Centralized Physically Distributed Architecture

Advantages of the Logically Centralized Physically Distributed Architecture

- ✓ *Security: High Availability*

  The security, with respect to availability is highly improved. This architecture is very fault tolerant as all other controllers serve as backup for each other. A failure of one controller does not isolate that segment of the network. Instead, through the access point, traffic is redirected for flow processing to other controllers that are available.

- ✓ *Improved Scalability*

  This architecture supports load balancing. Hence, traffic can be redirected to other controllers for processing when the active controller for a domain is busy. Also, when traffic demands grow, all the controllers are able to work together to fairly distribute the load thereby avoiding bottleneck related problems. Quality of service is also improved greatly as a result.

Disadvantages of the Logically Centralized Physically Distributed Architecture.

- ✓ *Relatively Higher Cost*

  This architecture, like the distributed controller architecture, requires several separate controllers to be procured for the network design. Also, the need for an access point also adds up to the cost of implementing this system.

CHAPTER SEVEN: NETWORK FUNCTION VIRTUALIZATION

7.0 Overview

Network Functions Virtualization (NFV) is the decoupling of network functions from proprietary hardware appliances and running them as software in virtual machines (VMs). By so doing, NFV provides an efficient way to reduce cost and accelerate service deployment for network operators by decoupling functions like a firewall or encryption from dedicated hardware and moving them to virtual servers. Instead of installing expensive proprietary hardware, service providers can purchase commodity switches, storage and servers to run virtual machines that perform network functions. This collapses multiple functions into a single physical server, reducing costs and minimizing truck rolls.

This virtualization of network functions reduces dependency on dedicated hardware appliances for network operators, and allows for improved scalability and customization across the entire network. Different from a virtualized network, NFV seeks to offload network functions only, rather than the entire network.

7.1 NFV architecture

The NFV architecture is standardized and defined by the European Telecommunications Standards Institute (ETSI). It consists of the following elements:

- ✓ **Network functions virtualization infrastructure (NFVi)**

  The NFVi provides the necessary virtualization layer (either a hypervisor such as KVM, or a container management system such as Kubernetes) and the physical computing infrastructure (CPU, storage and networking components, etc) that would be required to host and run the network virtualized function applications. The NFVi is managed through the NFVi Infrastructure Manager (VIM), which is responsible for the allocation of resources for the Virtualized Network Functions (VNFs). A perfect example of the VIM is Red Hat's OpenStack Platform.

- ✓ **The Virtualized Network functions (VNFs)**

  These are the software applications that provide one or more network services as performed by the physical network devices like routers, firewalls, load balancers, etc. The VNFs use the virtualized Infrastructure provided by the NFVi to connect into the network and provide programmable and scalable network services needed. VNF Managers support the lifecycle of VFN instances and management of a VFN software.

- ✓ **Management, Automation and Network Orchestration (MANO)**

  The MANO provides the framework for managing NFV infrastructure and provisioning new VNFs. It instantiates the network services through the automation, provisioning and coordination of workflows to the VIM and VFN Managers that instantiate the VNFs and the overlay networking service chains. The MANA connects the NFV architecture with the existing OSS/BSS.

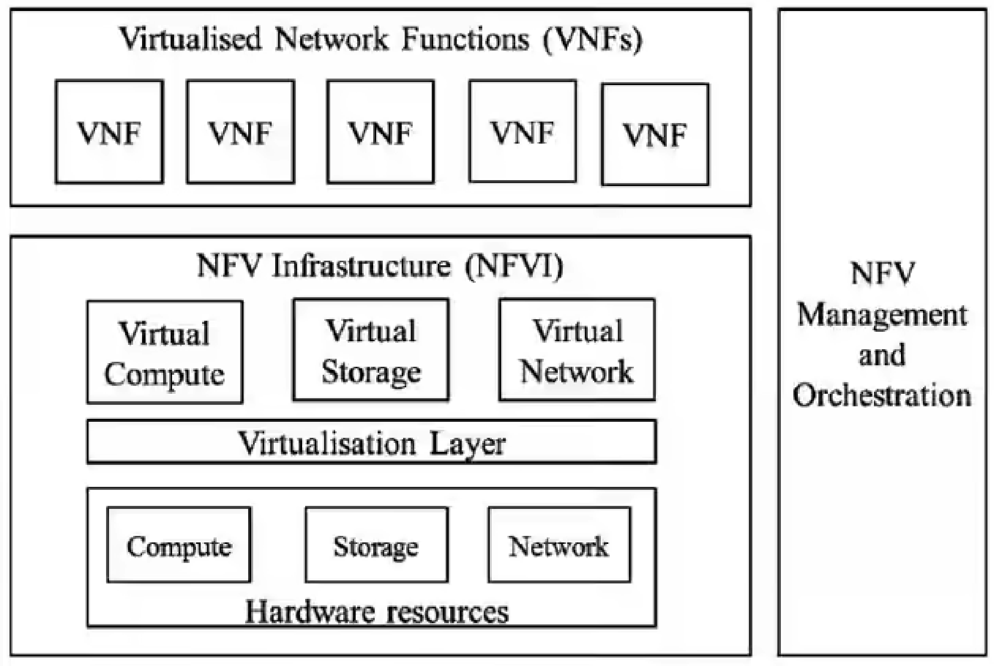


Fig. 7.1: NFV Architecture

7.2 Software-defined networking (SDN) and NFV Relationship

SDN and NFV mostly walk hand in hand yet they are not dependent on each other. They, however, have some similarities. They both rely on virtualization and use network abstraction just that they do so differently with regards to their functions. SDN separates network forwarding functions from network control functions with the goal of creating a network that is centrally manageable and programmable. NFV, on the other hand, abstracts network functions from hardware. NFV supports SDN by providing the infrastructure on which SDN software can run.

NFV and SDN can be used together, depending on what you want to accomplish, and both use commodity hardware. With NFV and SDN, you can create a network architecture that is more flexible, programmable, and uses resources efficiently.

7.3 The benefits of NFV

With NFV, service providers can run network functions on standard hardware instead of dedicated hardware. Also, because network functions are virtualized, multiple functions can be run on a single server. This means that less physical hardware is needed, which allows for resource consolidation that results in physical space, power, and overall cost reductions.

NFV gives providers the flexibility to run VNFs across different servers or move them around as needed when demand changes. This flexibility lets service providers deliver services and apps faster. For example, if a customer requests a new network function, they can spin up a new VM to handle that request. If the function is no longer needed, the VM can be decommissioned. This can also be a low-risk way to test the value of a potential new service.

In summary, the benefits are as follows:

Cost Effective: Network operators who virtualize their network can save money, shorten the time-to-market for new or updated products, and better scale and adjust resources available to applications and services.

Greater Resource Efficiency: A virtualized data center is more efficient to operate because more can be done with less. Data center footprint, power consumption, and cooling requirements can all be reduced or kept the same, but with increased workload capacity. This is possible because a single server can run multiple VNFs at once, so not as many servers are needed to do the same amount of work. When network demand changes, an organization can update its infrastructure through software instead of doing another truck roll. The instances where an organization needs to physically update its network and data centers are significantly reduced.

Flexibility: Organizations can use the agility of NFV to quickly adapt to changing business requirements and new market opportunities. In other words, the time-to-market period is shortened because the network infrastructure can be changed to adequately support the organization's new products. A network that has gone through NFV is also able to adjust quickly and easily to changes in resource demand as traffic coming to the data center increases or decreases. Scaling up and down in the number of VMs and the resources provided to them can be done automatically through SDN software.

CONCLUSION

Software-Defined Networking is quite an interesting concept in computer networking. It simplifies network control and management while maintaining a high-level quality of service within the SDN enabled networking environment. In this report, I have reviewed the basic concepts underlying the operation of SDN and the challenged SDN has addressed with regards to the traditional or conventional approach to networking. Also, Network Function Virtualization (NFV), which seeks to virtualize various network nodes as mere functions on a single server, was briefly introduced.

Research in SDN is very vibrant given that it is a relatively new concept and it is the future of Computer Networking. The following are a few research on the application areas and issues that might interest.

1. SDN for Internet of Things - [1], [2]
2. SDN for 5G Mobile Networks – [3]
3. Artificial Intelligence (AI) – Enabled SDN – [4][5]
4. SDN in Vehicular Ad Hoc Networks – [6]
5. Quality of Service in SDN – [7]
6. VNF Orchestration and Network Slicing – [8]
7. Load balancing in SDN – [9]
8. SDN in Wireless Networks – [10], [11]
9. Multi-domain SDN, Hierarchical SDN controllers [12]
10. Energy Efficiency in SDN – [13], [14], [15]
11. Security in SDN - [16]

REFERENCES

Report

1. Software-defined Networks: A comprehensive Approach, Second Edition. By Paul Goransson, Chuck Black & Timothy Culver.
2. Software Defined Networking Lecture Notes by Mohammad Alizadeh (MIT).
3. Software-Defined Networking (SDN) Lecture Notes, Scott Shenker (Berkeley).
4. Introduction to Software Defined Network (SDN), Hengky "Hank" Susanto, Sing Lab, HKUST.
5. Catherine Nayer Tadros et al., Logically Centralized-Physically Distributed Software Defined Network Controller Architecture, 2018 IEEE Global Conference on Internet of Things (GCIoT).
6. Advanced Computer Networks: Software-defined Networking.

Sample papers for the Research Areas

[1] Tryfon Theodorou, Lefteris Mamatas: Software defined topology control strategies for the Internet of Things; IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Nov. 2017; DOI: 10.1109/NFV-SDN.2017.8169884

[2] Samaresh Bera, Sudip Misra, Athanasios V. Vasilakos: "Software-Defined Networking for Internet of Things: A Survey"; IEEE Internet of Things Journal, Vol. 4(6), Dec. 2017; DOI: 10.1109/JIOT.2017.2746186

[3] Pedro Neves, Rui Calé, Mário Costa, Gonçalo Gaspar, Jose Alcaraz-Calero, Qi Wang, James Nightingale, Giacomo Bernini, Gino Carrozzo, Ángel Valdivieso, Luis Javier García Villalba, Maria Barros, Anastasius Gravas, José Santos, Ricardo Maia, Ricardo Preto: "Future mode of operations for 5G – The SELFNET approach enabled by SDN/NFV"; Computer Standards & Interfaces, Vol. 54(4), Nov. 2017; DOI: 10.1016/j.csi.2016.12.008

[4] Majd Latah, Levent Toker: “Application of Artificial Intelligence to Software Defined Networking: A Survey”; Indian Journal of Science and Technology, Vol. 9(44), Nov. 2016; DOI: 10.17485/ijst/2016/v9i44/89812

[5] Elisa Rojas: “From Software-Defined to Human-Defined Networking: Challenges and Opportunities”; IEEE Network, Vol. 32 (1), Jan.-Feb. 2018

[6] Manisha Chahal, Sandeep Harit, Krishn K. Mishra, Arun Kumar Sangaiah, Zhigao Zheng: “A Survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases”; Sustainable Cities and Society, Vol. 35, Nov. 2017

[7] Murat Karakus, Arjan Durresi: “Quality of Service (QoS) in Software Defined Networking (SDN): A survey”; Journal of Network and Computer Applications, Vol. 80, Febr. 2017; DOI: 10.1016/j.jnca.2016.12.019

[8] Daniel King, Arsham Farshad, Jamie Bird, Lyndon Fawcett, Nektarios Georgalas, Matthias Gunkel, Kohei Shiomoto, Aijun Wang, Andreas Mauthe, Nicholas Race, David Hutchison: “Network service orchestration standardization: A technology survey” ; Computer Standards & Interfaces, Vol. 54(4), Nov. 2017; DOI: 10.1016/j.csi.2016.12.006

[9] Hong Zhong, Yaming Fang, Jie Cui: “LBBSRT: An efficient SDN load balancing scheme based on server response time”; Future Generation Computer Systems, Vol. 68, March 2017; DOI: 10.1016/j.future.2016.10.001

[10] Nachikethas A. Jagadeesan, Bhaskar Krishnamachari: “Software-defined networking paradigms in wireless networks: A survey”; ACM Computing Surveys, Vol. 47(2), Jan. 2015

[11] Israat Tanzeena Haque, Nael Abu-Ghazaleh: “Wireless Software Defined Networking: A Survey and Taxonomy”; IEEE Communications Surveys & Tutorials, Vol. 18(4), May 2016; DOI: 10.1109/COMST.2016.2571118

[12] Franciscus X.A. Wibowo, Mark A. Gregory, Khandakar Ahmed, Karina M. Gomez: “Multidomain Software Defined Networking: Research status and challenges”; Journal of Network and Computer Applications, Vol. 87, June 2017; DOI: 10.1016/j.jnca.2017.03.004

[13] Yanwen Wang, Hainan Chen, Xiaoling Wu, Lei Shu: An energy-efficient SDN based sleep scheduling algorithm for WSNs; Journal of Network and Computer Applications, Vol. 59, Jan. 2016; DOI: 10.1016/j.jnca.2015.05.002

[14] Hao Zhu, Xiangke Liao, Cees de Laat, Paola Grosso: "Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform"; Journal of Network and Computer Applications, Vol. 63, March 2016

[15] Mehmet Fatih Tuysuz, Zekiye Kubra Ankarali, Didem Gözüpek: A survey on energy efficiency in software defined networks; Computer Networks, Vol. 113, Febr. 2017; DOI: 10.1016/j.comnet.2016.12.012

[16] Ivan Farris, Tarik Tableb, Yacine Khettab and Jaeseung Song: A survey on emerging SDN and NFV security mechanisms for IoT systems. (Accepted for publication IEEE)