CYB301 Software Defined Networking Lecture Note

Software Defined Networks (SDN)

Table of Contents

UNIT 1

1.0 Introduction

Computer networks are complex and difficult to manage. These networks have many kinds of equipment, from routers and switches to middleboxes such as firewalls, network address translators, server load balancers, and intrusion detection systems. Routers and switches run complex, distributed control software that is typically closed and proprietary. The software implements network protocols that undergo years of standardization and interoperability testing. Network administrators typically configure individual network devices using configuration interfaces that vary across vendors and even across different products from the same vendor. Although some network management tools offer a central vantage point for configuring the network, these systems still operate at the level of individual protocols, mechanisms, and configuration interfaces. This mode of operation has slowed innovation, increased complexity, and inflated both the capital and operational costs of running a network.

Software Defined Networking (SDN) is changing the way we design and manage networks. SDN has two defining characteristics. First, an SDN separates the control plane (which decides how to handle the traffic) from the data plane (which forwards traffic according to decisions that the control plane makes). Second, an SDN consolidates the control plane, so that a single software control program controls multiple data-plane elements. The SDN control plane exercises direct control over the state in the network's data-plane elements (i.e., routers, switches, and other middleboxes) via a well-defined Application Programming Interface (API). OpenFlow is a prominent example of such an API. An OpenFlow switch has one or more tables of packet-handling rules. Each rule matches a subset of traffic and performs certain actions on the traffic that matches a rule; actions include dropping, forwarding, or flooding. Depending on the rules installed by a controller application, an OpenFlow switch can behave like a router, switch, firewall, network address translator, or something in between. Over the past few years, SDN has gained significant traction in industry. Many commercial switches support the OpenFlow API. Initial vendors that supported Open-Flow included HP, NEC, and Pronto; this list has since expanded dramatically. Many different controller platforms have emerged. Programmers have used these platforms to create many applications, such as dynamic access control, server load balancing, network virtualization, energy efficient networking, and seamless virtual-machine migration and user mobility. Early commercial successes, such as Google's wide-area traffic-management system and Nicira's Network Virtualization Platform, have garnered significant industry attention. Many of the world's largest information-technology companies (e.g., cloud providers, carriers, equipment vendors, and financial-services firms) have joined SDN industry consortia like the Open Networking Foundation and the Open Daylight initiative.

Although the excitement about SDN has become more palpable during the past few years, many of the ideas underlying SDN have evolved over the past twenty years (or more!). In some ways, SDN revisits ideas from early telephony networks, which used a clear separation of control and data planes to simplify network management and the deployment of new services. Yet, open interfaces like OpenFlow enable more innovation in controller platforms and applications than was possible on closed networks designed for a narrow range of telephony services. In other ways, SDN resembles past research on active networking, which articulated a vision for programmable networks, albeit with an emphasis on programmable data planes. SDN also relates to separating the control and data planes in computer networks.

1.1 The history of Software Defined Networking (SDN)

Software Defined Network's origins can be traced to a research collaboration between Stanford University and the University of California at Berkeley that ultimately yielded the OpenFlow protocol in the 2008. The researchers were credited the idea of bringing the tenets of virtualization Networking and thus creating the software defined network market. Traditional networking use integrated hardware and software to direct traffic across a series of routers and switches. The original use case for SDN was to virtualize the network by separating the control plane that manages the network from the data plane where traffic flows. There is a smart control running specialized software that manages all network traffic in the datacenter and a series of routers and switches that forwards packets of traffic.

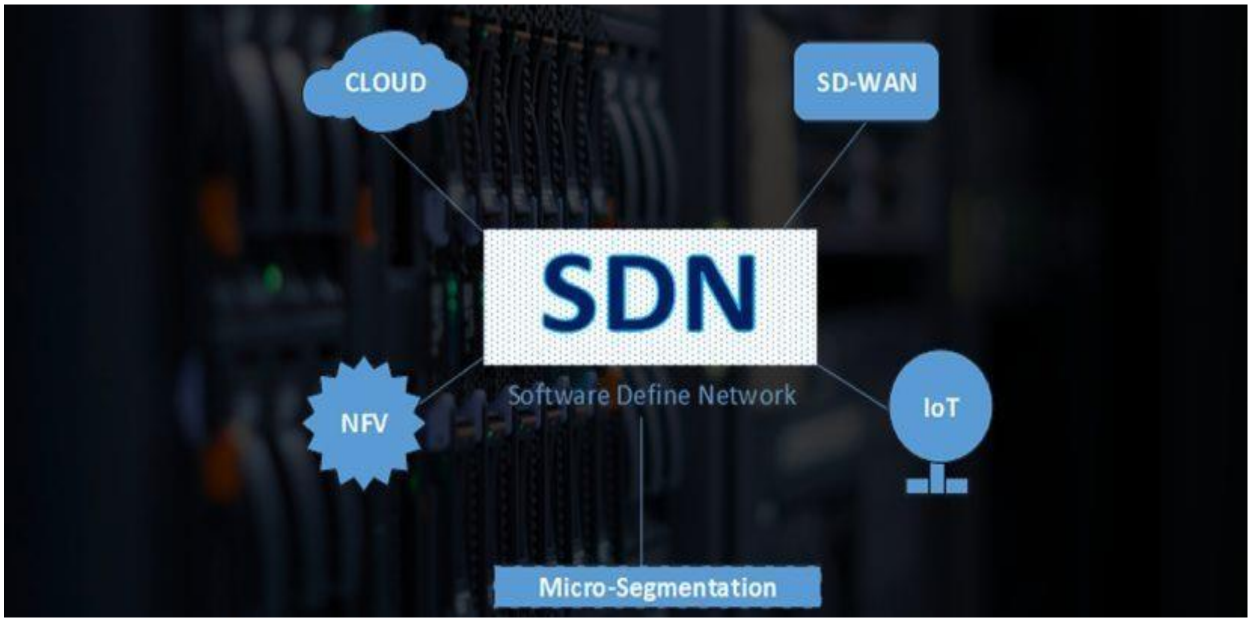
Virtualizing a network comes with advantages some of which are; networks can be spanned up and down dynamically, can be fine-tuned for specific application use cases and security policies for the installed on each individual network.

Today, the SDN market has evolved and it is breaking out of the datacenter. SDN is being used in the wide area network to control how enterprise connect their branch offices. This use-case called SDWAN (Software Defined Wide Area Networking) uses software to aggregate multiple types of network connections such as broadband, MPLS (Multiprotocol Label Switching) or wireless to create strong and cost effective connections.

Micro-segmentation is the idea of using SDN for security. Certain networks can be ultra-secure and carry sensitive data while other networks can be public facing. If a hacker gets into a public facing web server, they are restricted to the server segment of the network, limiting the hacker's ability to access any other segment such as secured datacenter networks.

SDN is also used in Network Function Virtualization (NFV), replacing specialized hardware like firewalls and load balancers with software running on off the shelf server hardware. Some vendors now use SDN to connect data centers to public cloud providers creating a hybrid cloud network that includes micro-segmentation or dynamic scaling abilities. Other SDNs could be used to help manage delude of traffic from the Internet of Thing (IoT) segmenting network traffic and helping to organize the data.

SDN has evolved from a specific use case to being applied to many different areas of networking, both within the data center unto the cloud and in the new world of IoT. As software are used to control the network, it becomes more agile, easy to manage and ready to adapt to whatever use case that emerge in the future.



1.2 Definition of SDN

Software-Defined Networking is a network architecture approach that enables the network to be intelligently and centrally controlled, or programmed, using software applications. This helps operators manage the entire network consistently and holistically, regardless of the underlying network technology.

SDN is the most frequently used means for application deployment, by enterprises to deploy their applications faster while also cutting the overall deployment and operating costs. IT administrators using SDN can manage and provision their network services from a centralized point.

1.2.1 The basic network functions are:

a. Allowing nodes in the network to communicate with one another.

b. Network elements connected by links form a topology

c. Each node runs some kind of distributed algorithm, e.g. Open Shortest Path First (OSPF), to figure out the path from A to B, in Figure 1.

d. Network administrator can change network parameters to achieve certain objective: e.g. changing routes

e. Limited programmability

f. Equipment vendors provide a set of routing (network control) choices: OSPF, Intermediate System- Intermediate System (ISIS), Boarder Gateway Protocol (BGP), etc

g. Making the network control like an APP that user can develop by themselves.

h. Making network control programmable.

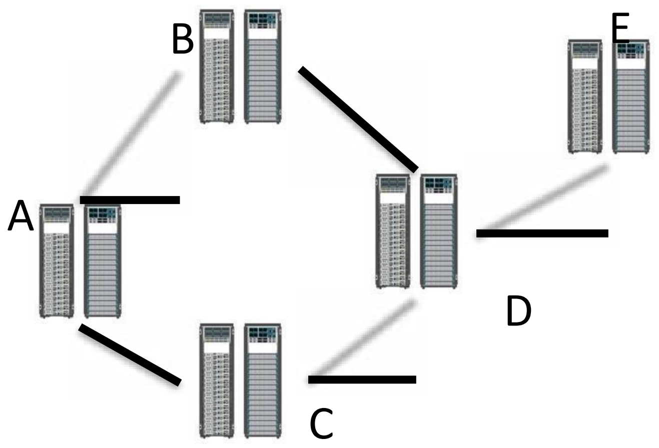


Figure 1.1. Route Paths

1.3 Motivation and Concept of SDN

The motivation and concepts of SDN are divided into short term and long term, and there are also reasons why we want to make network control programmable.

1.3.1 Short term:

a. Existing network control is no longer sufficient in several important areas, it needs innovation. examples; Data centers, Wireless, network security

b. Existing network control is getting too complicated.

   Example a lot of different middleboxes, each speaks its own language, and interferences with one another. Network Address Translators (NAT), firewall, IDS, WAN optimizer, load balancer, traffic shapers, transparent web proxy, application accelerators.

c. It would be nice to provide a unified mechanism to deploy and manage these middleboxes

1.3.2 Long term:

a. Innovation is good for the networking industry.

1.4 The Basic Model of SDN Key Features

SDN is modelled as a set of client-server relationships with the SDN controller at its core.

a. Service Customer•

Manage-control network services via SDN Controller•

Send/receive data using network resources Network/Service Provider•

Map customer service intention to resources•

Resources include forwarding, processing, and storage domains•

Recursively map services and resources to scale or to span multiple administrative domains•

Virtualize resources to views for customers•

Orchestrate resources needed for service fulfillment Spans all forms of•

b. Environments — Carrier, Enterprise Campus, Cloud•

c. Services — residential, business, intent-based, non-intent-based•

d. Resources — physical, virtual, compute, storage, forwarding Enables•

e. Security• - Policy enforcement•

Information hiding Scales•

From completely local to global service span•

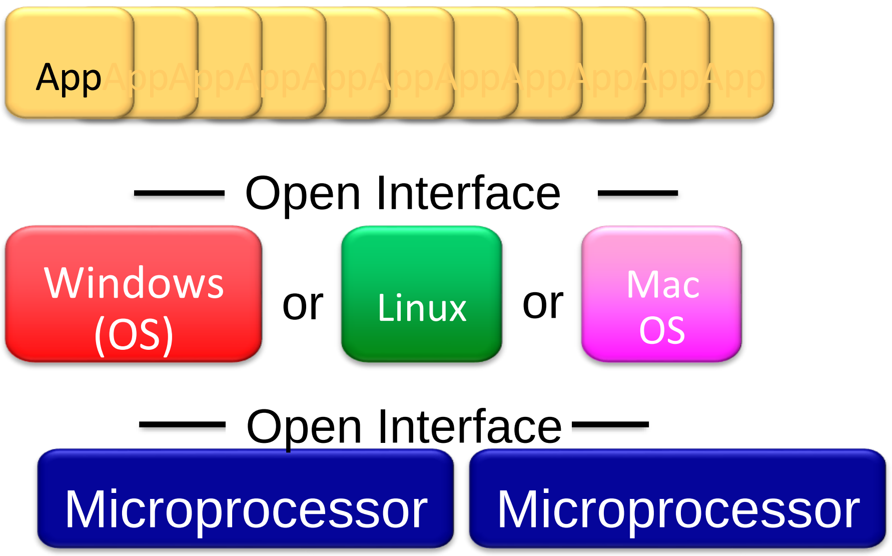
Hierarchical or peer relationship•

Combinations of the above

1.4.1 Computing systems once upon a time

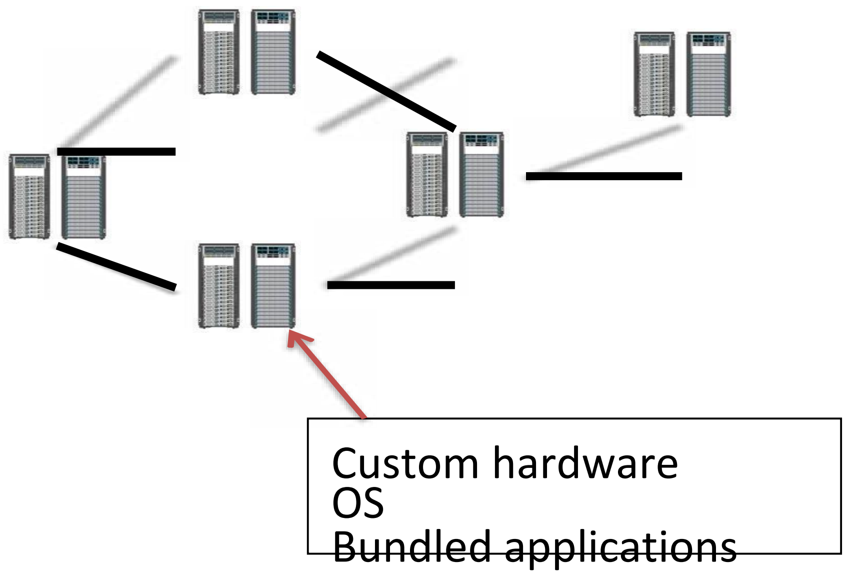
i. Vertically integrated systems
ii. Proprietary hardware
iii. Proprietary Operating System (OS)
iv. Proprietary applications – Highly reliable
v. Can you picture google, yahoo, Facebook on such a platform?
vi. Slow software innovation – Proprietary development
vii. Small industry

1.4.2 Computing Systems now



i. Open interfaces
ii. Fast innovation so Everyone can participate
iii. Hugh industry
iv. Software is now part of everything.

1.4.3 Conventional networking system today (before SDN)



Mainframe mindset:

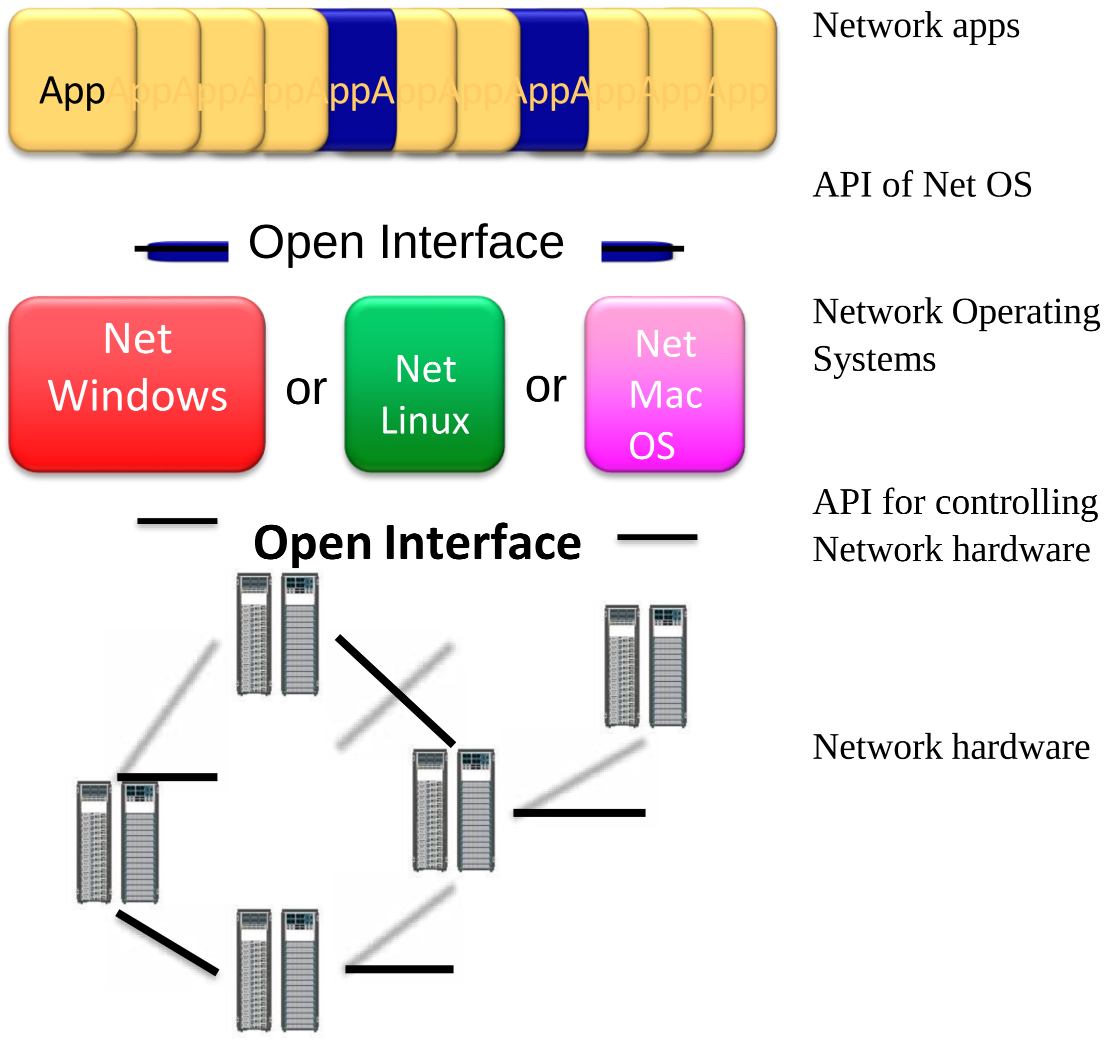
i. Software for the control plane cannot be separated from the forwarding hardware in the data plane.

ii. Vertically integrated, complex, closed, proprietary

iii. Innovation is only possible if one has access to the router box.

iv. No significant innovation in the past 40 years.

1.4.4 Ideal Networking System for Innovation



1.4.5 Control plane and data plane

a. Control plane of a network

– The functions of a network that control the behavior of the network

E.g: Which path to take for a packet? Which port to forward a packet? Should the packet be dropped?

– Control plane functions are typically realized by software such as routing protocols, firewall code, etc.

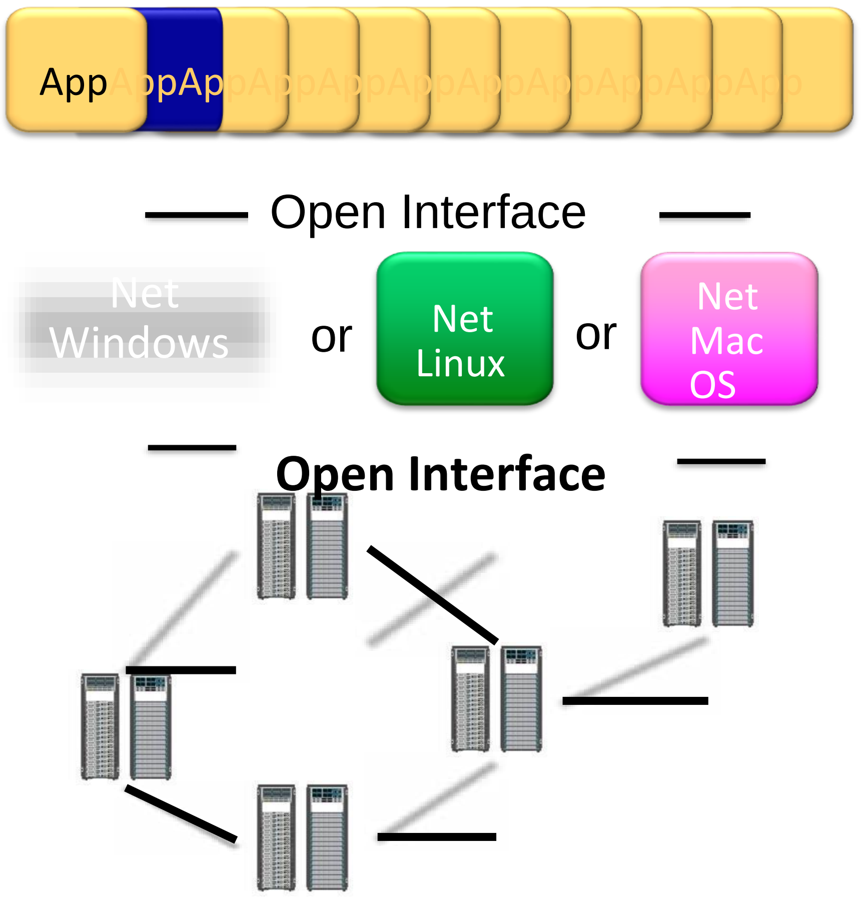
b. Data plane of a network

- The functions of a network that actually forward or drop packets.
- Data plane functions are typically realized by hardware

c. Control plane and data plane are vertically integrated in traditional networking equipment

- Separating software from hardware -separating control plane from data plane.

1.4.6 Ideal networking system for innovation



i. Separate hardware from software

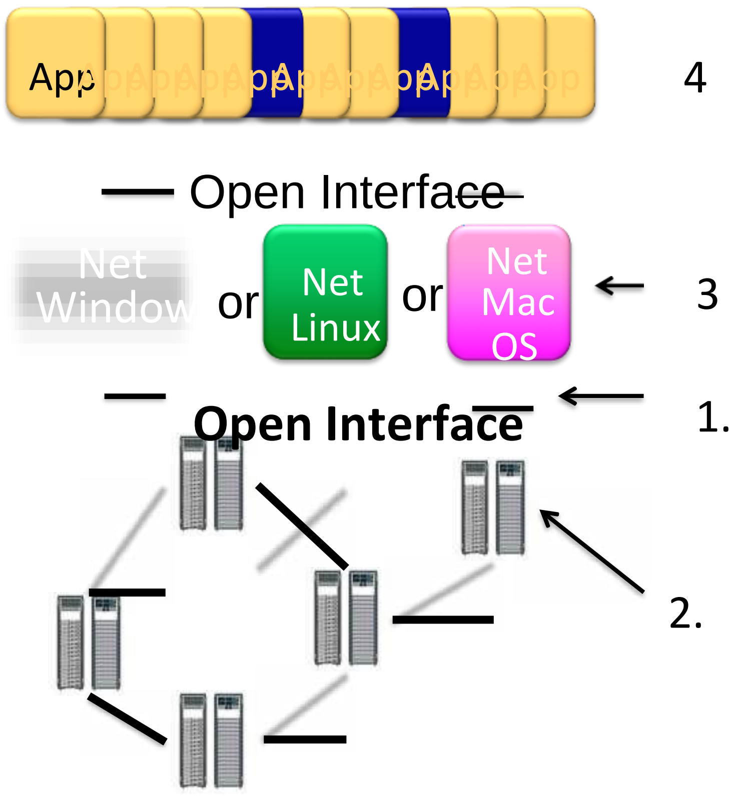
ii. Standardize the interface

   – Each layer provides an abstraction

iii. Innovation is possible for anyone just like software development for a computing system.

iv. This is the vision of SDN/OpenFlow.

1.4.7. SDN now: separate forwarding hardware from controlling software



i. OpenFlow: standardized for Ethernet/IP/TCP

ii. OpenFlow enabled switches/routers simple hardware doing forwarding only forwarding can be set by other entity through OpenFlow

iii. SDN Controllers; Floodlight, nox, etc.

iv. Firewall, Virtual network, TE, etc.

Northbound API, not Standardized yet

1.4.8 Major paradigm shift with SDN

i. No longer use distributed control protocols

ii. Design one distributed system (NOS) with the global view of the network – Use for all control functions

iii. Now just defining a centralized control function

Configuration = Function (global view)

iv. This may look easier, but this is not how it used to work, everything is new – innovation at all levels for this to happen.

High level programming languages to describe network configuration

v. Compiling and runtime system to realize the program efficiently, correctly, and safely.

Abstraction design

vi. Debugging infrastructure – network OS design – etc.

1.4.9 How an SDN operates

a. Network applications specification for the network functions (not the detailed implementation on the physical devices):

i. Access control: who can talk to who.

ii. Isolation: who can hear my broadcasts.

iii. Routing: only specify routing to the degree you care • Some flows over satellite, others over landline.

iv. Traffic Engineering (TE): specify in terms of quality of service, not routes.

b. Network OS (or something like a compiler) compiles the network application and computes the configurations on physical devices based on the global view.

c. Network OS distributes the configuration to physical devices through OpenFlow.

1.4.10 SDN promises

a. A lower-entry point for innovation in the network control.

b. Solve the issues in the current network configuration challenges.

c. Data plane interacts with many control entities

d. Configure locally to achieve a global network function.

1.4.11 Some SDN issues

a. Abstraction

- A new programming system to specify network functions (programming SDN)

- An API that provides network abstraction to network application (SDN controller design)
- Performance (scalability), Controller
- Communication between controller and devices

Forwarding

- Correctness and debugging – A SDN program has a higher bar than a typical program, multiple levels • Security

UNIT 2

2.0 The software defined Network architecture

A software-defined network (SDN) architecture (or SDN architecture) defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware. SDN architecture separates the SDN control plane and the SDN data plane of the networking stack.

SDN architectures can take a variety of forms. Following is an example of an architecture based on SDN controllers. The first tier in the SDN architecture is the physical infrastructure, which includes all the hardware devices and cabling required to support the network. Network control is decoupled from hardware and given to a software application, in this case an SDN controller. Controllers, which initiate and terminate traffic, make up the second tier of the architecture. The third tier is the SDN applications, which direct specific functions through the controller. Types of SDN apps include programs for network virtualization, network monitoring, intrusion detection (IDS) and flow balancing (the SDN equivalent of load balancing), among a great number of other possibilities.

Software defined architecture (SDA) provides a layer of virtualization between the software and its users, which connects users to a simple dashboard that masks the complex systems operating in the background.

2.1 The key features of SDN Architecture are:

a. **DIRECTLY PROGRAMMABLE**. Network control is directly programmable because it is decoupled from forwarding functions.

b. **AGILE**. Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.

c. **CENTRALLY MANAGED**.

2.2 Two main key components of software defined network infrastructure:

a. The SDN controller (only one, could be deployed in a highly available cluster)

b. The SDN-enabled switches (multiple switches, mostly in a Clos topology in a data center)

2.3 A typical representation of SDN architecture includes three layers:

a. The application layer

b. The control layer

c. The infrastructure layer

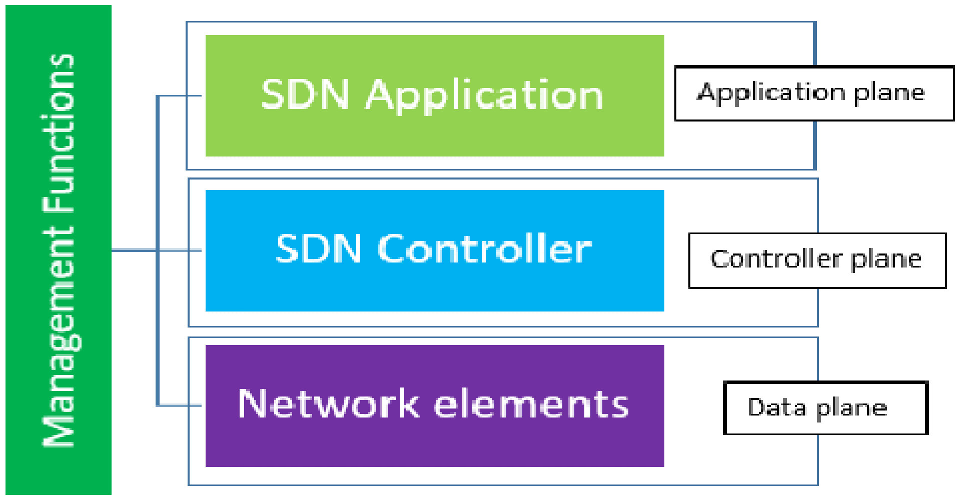


Figure 1: Basic SDN architectural model.

2.4 Models of SDN

Several decades ago, computer scientist and networking author Andrew S. Tanenbaum famously quipped that the nice thing about standards was that there were so many to choose from. The same could be said today, perhaps, about the number of software-defined networking models that lay before cloud providers.

But before providers can even consider a software-defined networking (SDN) deployment or even a pilot test, they must first choose which model or models of SDN to support. However, the wrong choice could waste time, strand assets and even put the provider's service offerings at competitive disadvantage.

The three primary models of SDN, explaining the basic goals, mechanisms, benefits and drawbacks of each.

Three models of SDN:

a. The Network virtualization model

b. The Evolutionary approach model

c. The OpenFlow model

a. The network virtualization model

The simplest model the market considers SDN is the network virtualization model popularized by Nicira, a startup acquired by VMware in 2012. The primary goals of network virtualization are to eliminate the restrictions on local area network (LAN) partitioning that exist in the Ethernet virtual LAN (VLAN) standards and to address the scalability issues with multicasting (Multicast is communication between a single sender and multiple receivers on a network) in some Ethernet-based virtual network architectures. To accomplish this, network virtualization platforms augment a software element, usually the hypervisor, but cloud-building software like OpenStack could also be modified, with an interface that creates VLANs that are based on tunnels running on top of traditional Ethernet. Network equipment and operations are not affected, and tens of thousands of virtual networks (or more) could be created this way, in theory.

The greatest benefit of network virtualization is it supports multi-tenant clouds without requiring changes to the network itself. This SDN model maps easily to popular virtualization interfaces in cloud networking mechanisms, such as OpenStack's Quantum or most cloud DevOps tools *(DevOps Tool is an application that helps automate the software development process)* that support network provisioning. As a result, it becomes easy to integrate network provisioning with cloud service provisioning.

The greatest disadvantage is that the virtual networks, being "above" the network layer, simply appear as traffic to the network devices. Those devices can't prioritize individual virtual networks or report on their status unless deep packet inspection is used to identify the virtual network header. Finally, since software that's part of the cloud server stack creates the virtual networks, the virtual networks can only link virtual machines -- not users and devices.

b. The Evolutionary approach

The second model of SDN can be called the evolutionary model. With this model, the goal is to enhance software control of the network and its operations but within the boundaries of current networking technology. To achieve this, networking vendors are likely to align themselves with specific standards such as ***(Virtual eXtensible Local Area Network (VXLAN) is one of the Network Virtualization over Layer 3 (VXLAN),*** Generic Routing

Encapsulation (GRE), BGP and MPLS, and use them to partition the network into virtual communities and to manage traffic and Quality of Service. The vendors may then combine their solutions into a set of management interfaces that can be exercised from the cloud again, through DevOps tools or a cloud virtual interface like OpenStack Quantum.

Network devices implement this SDN model, making it fully integrated with network operations, FCAPS management and network monitoring. Conventional traffic engineering principles can be applied, and the virtual networks can theoretically extend from server to user, as long as the devices support the selected standards.

Most SDN vendors currently implement all of the network standards above, but some may not be available on all devices. This latter point is the first of several disadvantages with these evolutionary models, as providers will need to validate the standards existing equipment supports. A greater problem is that so far specific vendors offer these evolutionary SDN models, which may not fully interoperate with equipment from other vendors. This approach will also likely require special integration between the management systems and the cloud virtual networking or DevOps interfaces, and if the vendor doesn't provide this, then the operator will have to undertake the task.

c. The OpenFlow model

The final model is the OpenFlow model of SDN, the one most people associate with the term SDN. OpenFlow replaces the traditional, discovery-based creation of forwarding tables in switches and routers with centrally controlled forwarding, meaning that a central controller programs each device's forwarding table. This gives that central control point complete rule over how the network is segmented or virtualized, how traffic is managed and so forth. Any combination of controllers and switches that support compatible versions of OpenFlow (versions that support the network features needed) can be used in this model of SDN.

The greatest benefit of this final SDN model is that it's the model on which the concept of SDN was originally built. Early pilot tests and deployments suggest that OpenFlow can improve network availability and reliability while increasing network utilizations, thus reducing both capital infrastructure costs and operational expenses. If OpenFlow switches become pervasive over time, then it's possible that future networks could be built with open hardware at much lower cost.

The disadvantage of this model is the current lack of functional detail for all of the necessary components. OpenFlow is supported on most mainstream switches and routers, but not always with the same throughput traditional protocols could achieve. And, of course, this mechanism of OpenFlow support doesn't do much to drive down the cost of switching.

There are both open-source and commercial OpenFlow controllers available, but these do little more than send commands to switches to create paths, manage capacity and so forth. A set of higher-layer management applications are needed. These must connect through northbound APIs to the OpenFlow controllers, and these APIs are not standardized. The early implementations of OpenFlow demanded operators integrate multiple components to create a functional software-defined network; there was no complete commercial package available.

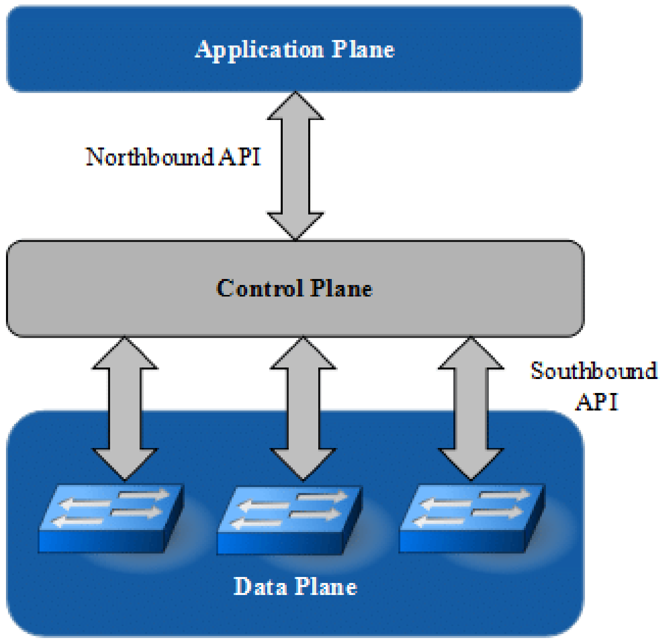


Figure 2: Basic SDN architectural model with API

Three models, but which is best?

Cloud providers struggling with VLAN limits on segmentation -- 4,095 VLANs per network -- or facing multicast issues in VLAN routing may want to look first at the virtual network model of

SDN. This model can be overlaid on the evolutionary SDN model too, though there may be some issues of harmonizing the management interfaces. Providers that have a large investment in data center networking equipment may want to look at this approach to avoid redundant costs.

The future is likely to mean at least some accommodation to OpenFlow, and so providers should look for OpenFlow support from their network equipment vendors, particularly when deploying new equipment. Fortunately, nearly all networking vendors have explicitly committed to supporting OpenFlow as part of their evolutionary SDN approach, which means that cloud providers can actually hope to use two or even three of the SDN models at once. As always, careful trial and test procedures are essential to ensure stable and profitable operation.

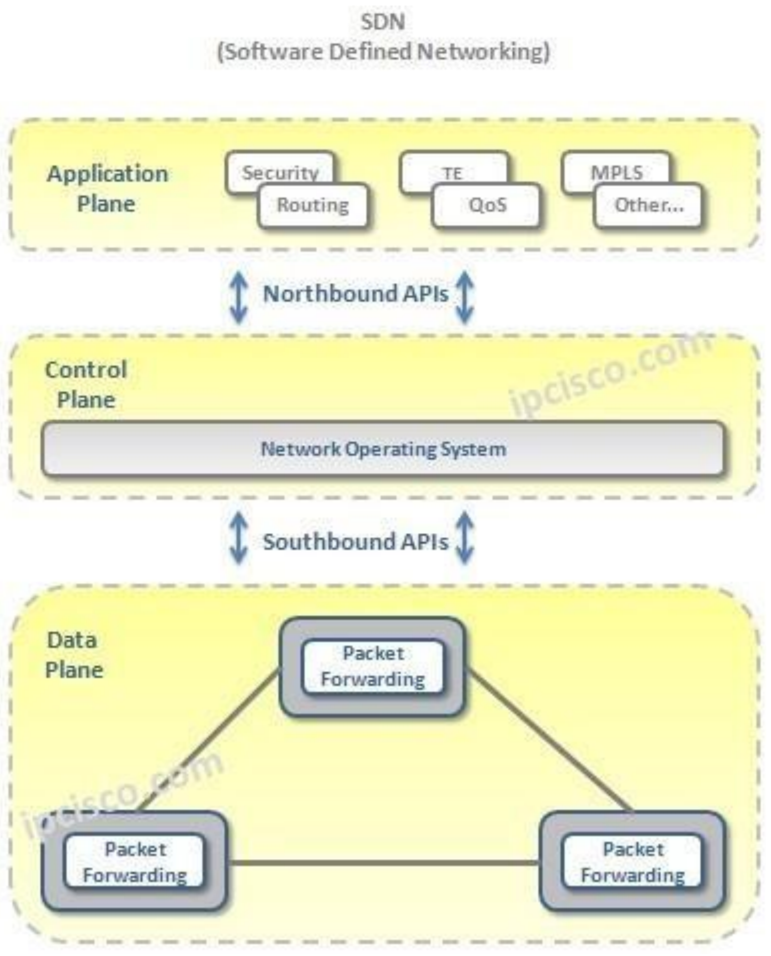


Figure 3: Basic SDN Architectural components

UNIT 3

3.0 Software Defined Networking Applications

SDN application is a software program which is designed to perform a task in a software-defined networking environment. This computer networking approach both helps network administrators to make systematic changes through open interfaces and potential lower-level functionality. It boosts functions that are accomplished in the hardware devices of a regular network through firmware by enlarging and subsisting upon them. SDN helps administrators manage the entire network using high-level functionality abstraction. Despite management abstraction, there must be some communication between the control plane and the data plane. The control plane manages traffic distribution and directs where traffic is sent, and the data plane is the underlying system that helps control the destination of the traffic.

a. Six Applications and Services SDN Supports and Optimizes include:

i. Security Services.
ii. Network Intelligence and Monitoring.
iii. Compliance and Regulation-Bound Applications.
iv. High-Performance Applications.
v. Distributed Application Control and Cloud Integration.

3.1. Open Daylight

Open Daylight is a modular open platform for automating and customizing networks regardless of scale or size, Open Daylight (ODL) came from the SDN movement and the drive for network programmability. Created as a foundation for commercial solutions to address different use cases in today's network environments, ODL functions as an SDN controller platform for both internal and external SDN applications.

i. Internal SDN applications

Applications that host the ODL controller software and get deployed internally are run from inside the container, and must be scripted in Java, the native language of ODL. Internal SDN applications must follow the controller's established design and execution constraints and implement them in the controller's Java Virtual Machine (JVM). These internal SDN apps can access the model-driven service abstraction layer (MD-SAL) applications and Java application programming interfaces (API) of the controller that's running inside the Open Services Gateway Initiative (OSGi) Alliance container of the controller.

ii. External SDN applications

External SDN applications are applications that host the rest of ODL controller software but get deployed externally and are run remotely from outside the container, sometimes using a different host than their controller. Unlike internal SDN applications that require ODL's native language, external SDN applications allow the use of any scripting language when writing them. External SDN applications use the app that provides them with RESTful access for their services and the Representational State Transfer (REST) API that their controller provides for them.

3.2 The advantages of Software Defined Networking

SDN has the advantage of generating a framework that bolsters data-intensive applications, such as big data and virtualization. Big data and virtual machines are somewhat intertwined. SDN centralize and simplify the control of enterprise network management. It offers the following succinct advantages:

a. **Centralized network provisioning**. SDN helps centralize enterprise management and provisioning by offering a unified perspective on the whole network. SDN can also speed up service delivery and boost agility in provisioning virtual and physical network devices in a central location.
b. **Holistic enterprise management**. Networks must meet the rising demand for processing requests. SDN helps your IT department adjust your network configuration with no impact on your network. Also, unlike Simple Network Management Protocol (SNMP), SND bolsters the management of physical and virtual switches and network devices that are from a central controller.
c. **More granular security**. Virtual machines pose a challenge for firewalls and content filtering, a challenge that's further compounded by personal devices. By establishing a

central control point for regulating security and policy information for your enterprise, the SDN controller quickly becomes a boon for your IT department.

d. **Lower operating costs**. Several benefits to SDN, such as having an efficient administration, server utilization improvements, and improved virtualization control, can dually help cut operating costs. Because many regular network administration issues can be automated and centralized, SDN can also help reduce operating costs and grow administrative savings.
e. **Hardware savings and reduced capital expenditures**. SDN adoption helps revive older network devices and simplifies the process of optimizing commoditized hardware. By following the instructions from the SDN controller, older hardware can be repurposed while less costly hardware can be deployed to optimal effect. This process allows new devices to become veritable "white box" switches that have intelligence focused at the SDN controller.
f. **Cloud abstraction**. Using SDN to abstract cloud resources helps simplify the process of unifying cloud resources. SDN controllers can manage all the networking components that comprise the massive data center platforms.
g. **Consistent and timely content delivery**. One big benefit of SDN is the ability to manipulate data traffic. It's easier to have quality of service for Voice over Internet Protocol (VoIP) and multimedia transmissions if you can direct and automate data traffic. SDN also helps with steaming higher-quality videos since SDN bolsters network responsiveness and, therefore, creates an improved user experience (UX).

3.3 Traditional network

Traditional networking is rooted in fixed-function network devices, such as a switch or router. These devices each have certain functions that operate well together and support the network. If the network's functions are implemented as hardware constructs, then its speed is usually bolstered.

Flexibility is a recurring hurdle for traditional networks. Few APIs are exposed for provisioning and most switching hardware and software is proprietary. Traditional networks often work well with proprietary provisioning software, but this software can't be quickly modified as needed.

Traditional networking consists of the following traits:

a. The functions of traditional networking are primarily implemented from dedicated devices using one or more switches, as well as routers and application delivery controllers.
b. The functionality of traditional networking is largely implemented in dedicated hardware, such as application-specific integrated circuits (ASIC). One of the negative aspects of this traditional hardware-centric networking is its limitations.

3.4. The differences between Software Defined Network and traditional networking

S/N	**Software Define Network**	**Traditional Network**
(a)	**(b)**	**(c)**
1.	SDN is software-based	Traditional networking is usually hardware-based
2.	SDN is more flexible, allowing users greater control and ease for managing resources virtually throughout the control plane	Traditional networks use switches, routers and other physical infrastructure to create connections and run the network
3.	SDN controllers feature a northbound interface that communicates with APIs, therefore, application developers can directly program the network	Traditional networking uses protocols and cannot program network
4.	SDN lets users use software to provision new devices	Traditional Network uses physical infrastructure
5.	IT administrators can direct network paths and proactively arrange network services	Traditional Network cannot.
6.	SDN also has the ability to better communicate with devices using the network.	Traditional Network does not possess that great ability.

7.	SDN virtualizes your entire network, it generates an abstract copy of your physical network	Traditional Network cannot be virtualized
8.	SDN provisions resources from a centralized location	Traditional Network resources are distributed
9.	SDN became a popular alternative to traditional networking because it lets IT administrators provision resources and bandwidths as needed without requiring an investment of additional physical infrastructure	Traditional networking requires new hardware to increase its network capacity.
10.	SDN requires only keystrokes	Traditional networking requires more equipment for expansion
11.	Controller may not be in the same box as the forwarding hardware	Forwarding hardware and its control are in the same box
12.	Centralized routing algorithm with logically global view	Distributed routing algorithm
13.	Network functions are realized with a global view	Network functions must be realized in a distributed manner, error-prone
15.	New abstraction must be developed for the centralized view	Network abstraction is embedded in the distributed algorithms

UNIT 4

4.0 Software Defined Networking Controller

An SDN controller is an application in software-defined networking (SDN) that manages flow control to enable intelligent networking. SDN controllers are based on protocols, such as OpenFlow, that allow servers to tell switches where to send packets. The controller is the core of an SDN network. Southbound and Northbound are a part of the SDN with distinct responsibilities. The differences between the two terminologies include:

4.1. Southbound SDN

In SDN, southbound interfaces are the OpenFlow protocol specification that enables communication between controllers and switches and other network nodes, which is with the lower-level components. This further lets the router to identify network topology, determine network flows and implement request sent to it via northbound interfaces.

Southbound APIs allows the end-user to gain better control over the network and promotes the efficiency level of the SDN controller to evolve based on real-time demands and needs. In addition, the interface is an industry standard that justifies the ideal approach the SDN controller should communicate with the forwarding plane to modify the networks that would let it progressively move along with the advancing enterprise needs. To compose a more responsive network layer to real-time traffic demands, the administrators can add or remove entries to the internal flow-table of network switches and routers.

Some of the popular southbound APIs are OpenFlow, Cisco, and OpFlex and other switch and router vendors that support OpenFlow include IBM, Dell, Juniper, Arista and more.

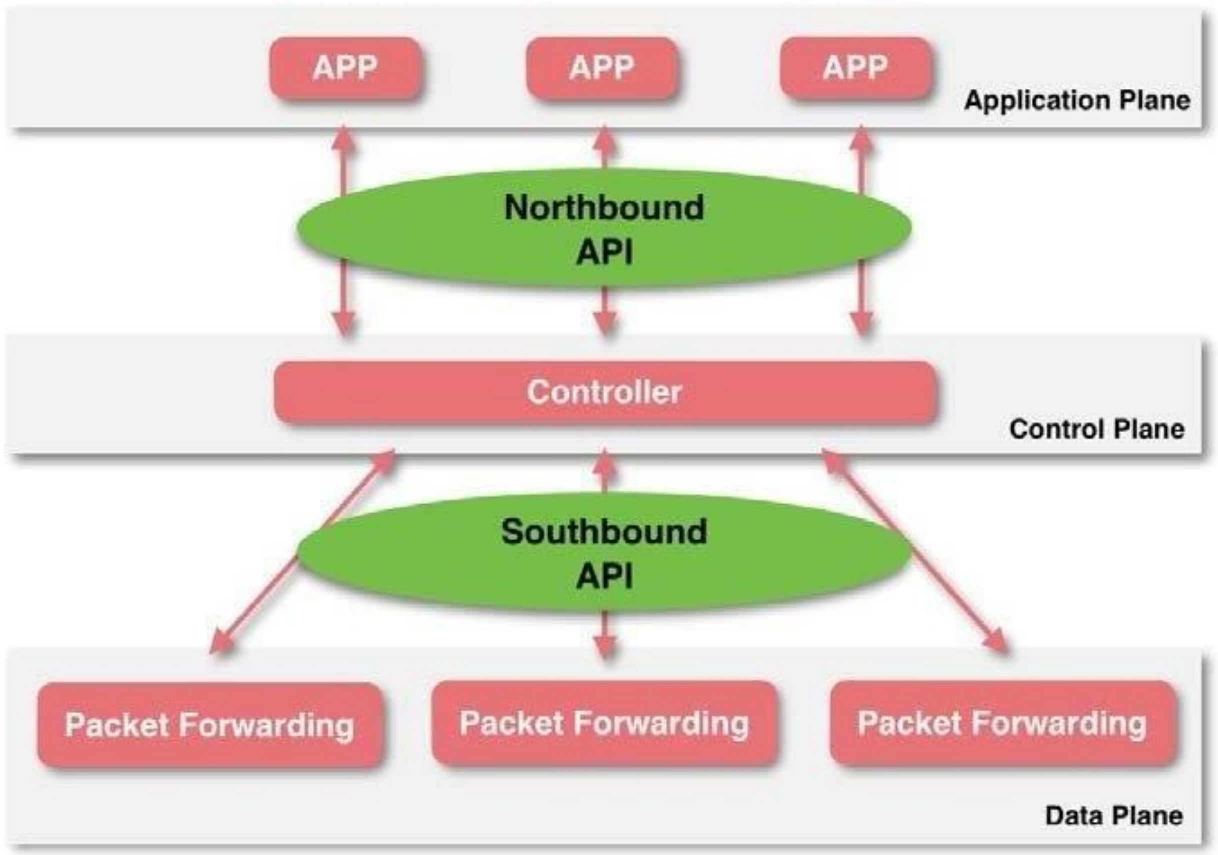
4.2. Northbound SDN

Contradictory to southbound API, northbound interfaces allows communication among the higher-level components. While the traditional networks use firewall or load balancer to control data plane behavior, SDN installs applications that uses the controller and these applications communicate with the controller through its northbound interface.

Experts say that it would be rather difficult to enhance the network infrastructure, as without a northbound interface the network applications will have to come directly from equipment vendors, which can make it harder to evolve. In addition, the northbound API makes it easier for network operators to innovate or customize the network controls and processing this task doesn't require help from expertise, as the API can be cleaned by a programmer who excels in programming languages like Java, Python, or Ruby.

Northbound APIs are used in different verticals that include non-profit sectors, educational institutes, IT companies and many more.

The functions of northbound API within an enterprise data center is to develop management solutions for automation and orchestration and exchange of actionable data between systems, whereas southbound works to deliver network virtualization protocols, interact with the switch fabric or integrate distributed computing network.



4.3 Examples of some controllers

a. **Beacon**: Beacon is a fast, cross-platform, modular, Java-based Open Flow controller that supports both event-based and threaded operation.

b. **FAUCET** is an OpenFlow controller for multi table OpenFlow 1.3 switches, that implements layer 2 switching, VLANs, ACLs, and layer 3 IPv4 and IPv6 routing, static and via BGP. It supports:

i. OpenFlow v1.3 (multi table) switches (including optional table features), hardware and software
ii. Multiple data paths and distributed switching under a single controller
iii. VLANs, mixed tagged/untagged ports
iv. ACLs matching layer 2 and layer 3 fields
v. IPv4 and IPv6 routing, static and via BGP
vi. Controller health and statistics via Prometheus
vii. Unit and systems tests run under Travis based on mininet and OVS

c. **Cherry** is an OpenFlow controller written in Go that supports OpenFlow 1.0 and 1.3 protocols. This project is not designed for general purpose, and it instead focuses on SDN (Software-Defined Networking) for an IT service provider.

Features

i. Supports OpenFlow 1.0 and 1.3 protocols
ii. Focuses on compatibility with commercial OpenFlow-enabled switches
iii. Supports network topology that has loops in it
iv. Provides several northbound applications: ProxyARP, L2Switch, Floating-IP, etc.
v. Provides simple plugin system for northbound applications
vi. RESTful API to manage the controller itself

d. **Open Network Operating System (ONOS)** is the leading open source SDN controller for building next-generation SDN/NFV solutions. ONOS supports both configuration and real-time control of the network, eliminating the need to run routing and switching control protocols inside the network fabric.

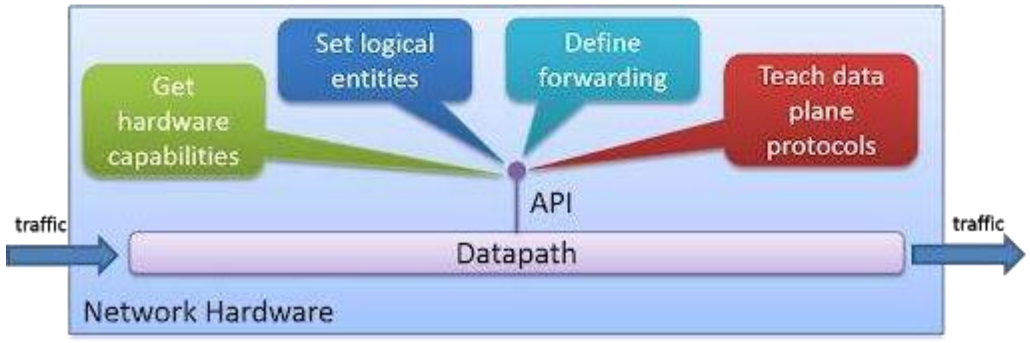
e. **Open vSwitch** is a production quality open source software switch designed to be used as a vswitch in virtualized server environments. A vswitch forwards traffic between different VMs on the same physical host and also forwards traffic between VMs and the physical network.

Open vSwitch is a multilayer software switch licensed under the open source Apache 2 license. The goal is to implement a production quality switch platform that supports standard management interfaces and opens the forwarding functions to programmatic extension and control.

UNIT 5

5.0 SDN Datapath

The SDN Datapath is a logical network device, which exposes visibility and uncontended control over its advertised forwarding and data processing capabilities. The logical representation may encompass all or a subset of the physical substrate resources. An SDN Datapath comprises a CDPI agent and a set of one or more traffic forwarding engines and zero or more traffic processing functions. These engines and functions may include simple forwarding between the datapath's external interfaces or internal traffic processing or termination functions. One or more SDN Datapaths may be contained in a single (physical) network element—an integrated physical combination of communications resources, managed as a unit. An SDN Datapath may also be defined across multiple physical network elements. This logical definition neither prescribes nor precludes implementation details such as the logical to physical mapping, management of shared physical resources, virtualization or slicing of the SDN Datapath, interoperability with non-SDN networking, nor the data processing functionality.



Control-Data-Plane Interface (CDPI) Agent. On top, SDN Applications exist in the application plane, and communicate their requirements via NorthBound Interface (NBI) Drivers. In the middle, the SDN Controller. translates these requirements and exerts low-level control over the SDN Datapaths, while providing relevant.

5.1 Open System Interconnection (OSI) MODEL

The Open Systems Interconnection model (OSI model) is a conceptual model that characterises and standardises the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard communication protocols.

a. **The physical layer** is responsible for the transmission and reception of unstructured raw data between a device and a physical transmission medium. It converts the digital bits into electrical, radio, or optical signals. Layer specifications define characteristics such as voltage levels, the timing of voltage changes, physical data rates, maximum transmission distances, modulation scheme, channel access method and physical connectors. This includes the layout of pins, voltages, line impedance, cable specifications, signal timing and frequency for wireless devices. Bit rate control is done at the physical layer and may define transmission mode as simplex, half duplex, and full duplex. The components of a physical layer can be described in terms of a network topology. Physical layer specifications are included in the specifications for the ubiquitous Bluetooth, Ethernet, and USB standards. An example of a less well-known physical layer specification would be for the CAN standard. Layer 2: Data Link Layer[edit]
b. **The data link layer** provides node-to-node data transfer—a link between two directly connected nodes. It detects and possibly corrects errors that may occur in the physical layer. It defines the protocol to establish and terminate a connection between two physically connected devices. It also defines the protocol for flow control between them.
c. **The transport layer** controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. Some protocols are state- and connection-oriented. This means that the transport layer can keep track of the segments and retransmit those that fail delivery. The transport layer also provides the acknowledgement of the successful data transmission and sends the next data if no errors occurred. The transport layer creates segments out of the message received from the application layer. Segmentation is the process of dividing a long message into smaller messages.
d. **The session layer** controls the dialogues (connections) between computers. It establishes, manages and terminates the connections between the local and remote application. It provides for full-duplex, half-duplex, or simplex operation, and establishes procedures for check pointing, suspending, restarting, and terminating a session. In the OSI model, this layer is responsible for gracefully closing a session. This layer is also responsible for

session check pointing and recovery, which is not usually used in the Internet Protocol Suite. The session layer is commonly implemented explicitly in application environments that use remote procedure calls. In the modern TCP/IP system, the session layer is non-existent and simply part of the TCP protocol.

e. **The presentation layer** establishes context between application-layer entities, in which the application-layer entities may use different syntax and semantics if the presentation service provides a mapping between them. If a mapping is available, presentation protocol data units are encapsulated into session protocol data units and passed down the protocol stack. This layer provides independence from data representation by translating between application and network formats. The presentation layer transforms data into the form that the application accepts. This layer formats data to be sent across a network. It is sometimes called the syntax layer.[26] The presentation layer can include compression functions.
f. **The application layer** is the OSI layer closest to the end user, which means both the OSI application layer and the user interact directly with the software application. This layer interacts with software applications that implement a communicating component. Such application programs fall outside the scope of the OSI model. Application-layer functions typically include identifying communication partners, determining resource availability, and synchronizing communication. When identifying communication partners, the application layer determines the identity and availability of communication partners for an application with data to transmit. The most important distinction in the application layer is the distinction between the application-entity and the application. For example, a reservation website might have two application-entities: one using HTTP to communicate with its users, and one for a remote database protocol to record reservations. Neither of these protocols have anything to do with reservations. That logic is in the application itself. The application layer has no means to determine the availability of resources in the network.

UNIT 6

6.0 Control Data Plane Interface

The control plane is the part of a network that carries signaling traffic and is responsible for routing. Control packets originate from or are destined for a router. Functions of the control plane include system configuration and management.

The control plane, the data plane and the management plane are the three basic components of a telecommunications architecture. The control plane and management plane serve the data plane, which bears the traffic that the network exists to carry. The management plane, which carries administrative traffic, is considered a subset of the control plane.

In conventional networking, all three planes are implemented in the firmware of routers and switches. Software-defined networking (SDN) decouples the data and control planes, removesthe control plane from network hardware and implements it in software instead, which enables programmatic access and, as a result, makes network administration much more flexible.

Moving the control plane to software allows dynamic access and administration. A network administrator can shape traffic from a centralized control console without having to touch individual switches. The administrator can change any network switch's rules when necessary, prioritizing, de-prioritizing or even blocking specific types of packets with a very granular level of control.

6.1 Data Plane Security

The data plane can be further divided into two categories, the wired and wireless data plane. Both data planes may share some security concerns but there are also security concerns that are specific to either one of the data planes. The detailed security concerns will be discussed as follows.

6.1.1. Wired Data Plane

As illustrated in Figure 6.1, the wired data plane is much simpler compared to the wireless data plane. It involves only the switches, hosts, or any other devices that are connected through the switch. Even if the east/west-bound and southbound communication channels are secure, it does not guarantee that the communication between the devices within the data plane is secure.

Possible attacks that can be launched within the data plane are Denial-of-Service (DoS) attack and man-in-the-middle attacks that intercept messages from the insecure communication channel and so forth.

6.1.2. Wireless Data Plane

With the growing use of mobile and Internet of Things (IoT) devices, the wireless data plane becomes enormous in size and involves complicated topologies. The capability crisis arises when mobile devices grow at a pace that exceeds the wireless spectrum capability. Therefore, the wireless resource management becomes crucial in order to sustain the network performance for the vast wireless data plane.

This also drives telecommunication providers to look for alternatives to fully utilize their network resources especially for the wireless portion as the wireless bandwidth is limited and expensive. One such solution for them will be to manage their wireless data plane with SDN.

Besides the wireless resource management, certificate management will also be involved if Transport layer security (TLS) were to be used to secure the data plane. The certificate management can be very complicated due to the enormous amount of devices involved. Besides that, it is also bandwidth consuming to perform the TLS handshake that involves certificate exchanges for authentication.

Unlike the wired data plane, the malicious user does not need to have physical access to the switch to perform any malicious activity. As long as the malicious user is within the wireless coverage range, he/she can simply intercept the wireless communication or even modify the information if the wireless data plane is not protected. This compromise both the data integrity and confidentiality, and hence security is not a luxury feature but a necessity in the wireless communication especially for IoT devices that are deeply involved in personal privacy or even life threatening in the case of IoT devices used in healthcare.

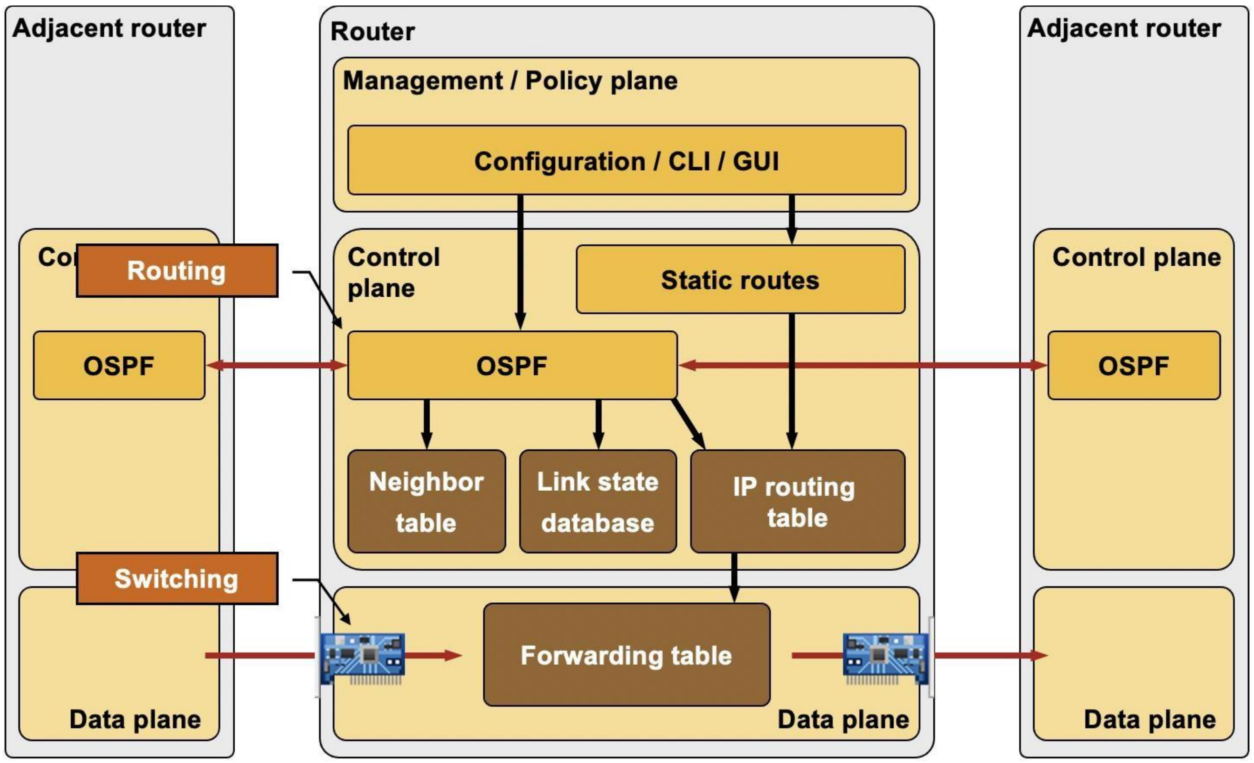


Figure 6.1. Wired data plane

UNIT 7

7.0 SDN Deployment

The SDN technology abstracts the lower level functions of a network onto a normalized control plane, allowing administrators to direct traffic flows from a centralized console. The concept promises to lower operating expenses (OPEX) and capital expenses (CAPEX) by making networks more interoperable and less reliant on expensive proprietary hardware. However, companies cannot reap the benefits of SDN without first choosing a model. Selecting the wrong model can waste company time and lead to unnecessary drawbacks.

Three different models for deploying SDN while highlighting their advantages and disadvantages.

7.1. Switch-based SDN (OpenFlow)

The idea of SDN was originally based on the switch-based model. The switches process packets with a system like OpenFlow, which dictates the behavior of the network switches. This provides a central control point that manages how the switches handle network traffic. The model can use a mix of controllers and switches that support compatible versions of OpenFlow. Businesses can choose between virtual or physical OpenFlow-enabled switches.

A disadvantage of the switch-based model is it requires purchasing switching fabric, meaning businesses may have to update their firmware or hardware. Furthermore, not all switches and routers support OpenFlow. Once the system is activated, other switches may no longer operate at a line rate, which doesn’t help reduce the price of switches.

7.2. Hybrid SDN (Evolutionary)

The hybrid SDN model combines two or more networking technologies in a single environment. A network manager, for instance, can configure the SDN control plane to manage specific traffic flows while traditional networking protocols manage the other traffic flows on the network. Hybrid SDN uses a variety of networking types, including VPN, Ethernet, MPLS, among others, to connect multiple locations and workers. A hybrid

SDN approach provides flexibility by using the newest SDN technologies to meet the demands of each location.

A challenge with a hybrid approach is it creates a complex network that can be difficult to troubleshoot. Both IT staff and network administrators must familiarize themselves with multiple systems, which could require investing in company training hours or additional staff. In addition, the hybrid SDN model can be difficult to deploy given the financial front needed to launch the network. Consequently, hybrid SDN tends to be better suited for mid-to-large sized companies.

7.3. SDN Overlay (Network Virtualisation)

The SDN overlay model tunnels through the physical network to run multiple virtual network (VN) topologies on top of an existing infrastructure. This allows a VN to be a layer 2 or layer 3 network, and a physical network to be a layer 2 or layer 3, based on the type of overlay. Under the overlay model, VN configurations determine higher level policy, and physical networks undergirding the architecture forward packets. Since virtualization takes place at the edge of the network, the layer 2 and layer 3 network do not change nor need configurations to maintain the virtualization of the network. Moreover, the overlay optimizes network resources by distributing bandwidth into channels and allocating each channel to a particular device or service. In addition, the overlay can boost network efficiency and productivity by performing various tasks typically reserved for network administrators automatically.

This model has vices running counter to its virtues. The physical network does not adjust to alterations automatically as the virtual networks mature. In addition, the processing power of virtual broadband network gateway applications are sometime overcome by high traffic volumes. As a result, gateway applications may have to rely on physical hardware switches. Furthermore, encapsulation can limit the ability to utilize existing networks.

UNIT 8

8.0 SDN APPLICATION ENVIRONMENT

8.1 Internal SDN Applications:

Applications that are hosting the rest of the OpenDaylight controller software and are deployed internally, run inside the container. These applications must be written in the native language which is Java for ODL. Internal SDN applications must also adhere to the execution and design constraints of the controller. It must also execute in the same Java Machine as the controller which means that these types of the application must run locally with the controller. It can also access the Model Driven – Service Adaptation Layer (MD-SAL) applications and Java APIs of the controller running inside the controller's OSGi container.

Applications that are hosting the rest of the Open Daylight controller software, and are deployed externally, run outside the container. Any language can be used for writing External SDN applications that are scripting languages such as Bash. These applications can be run remotely which means on a different host than the controller. These applications will also use the application providing Restful access to their services and REST API provided by the controller.

8.2 Top Application and Service that can benefit from SDN are:

a. Security Services

The modern virtualization ecosystem supports specific virtual service that is running within the network layer. It means an incorporating function like NFV into SDN platforms. This type of network security creates a genuinely proactive environment that is capable of risk reduction and responds to the incidents very quickly. Whenever a violation occurs, every second is quite critical to stop the attack. It is also essential to identify the attack and also to ensure that other network components are safe from the attack. As the organization in the modern era becomes even more digitized, and as the network layer becomes even more critical, we will see even more attacks and more advanced sophisticated advanced persistent threats. You will be able to create a more proactive environment that is capable of responding to the changes if you integrate potent services into the SDN layer.

b. Network Monitoring and Intelligence

Modern SDN technologies help in abstracting one of the most critical layers within the data centre that is the network. Network architectures are very much complicated and have to handle a lot more data than ever before. This means it's critical to know what is following through your environment. Do you have remission issues on a port? What will happen if you are running heterogeneous network architecture? Or, are you passing a lot of traffic and are heavily virtualized through the network architecture? All of these challenges or issues are diminished if you have a solid network monitoring and intelligence layer. However, you also gain benefit and true insight if you integrate these technologies into your SDN architecture. Even optimization, alerting, hypervisor integration, port configurations, and traffic flow can be incorporated into network monitoring and intelligence technologies. Also, these types of agile systems will also help you to monitor network traffic between your cloud ecosystem and your data centre.

c. Bandwidth Management

With the help of SDN applications, operators can use bandwidth management to ensure the end users to receive online video watching and optimal browsing experiences. This SDN application can also monitor the bandwidth requirements then provision user flows to match the latency and bandwidth requirements of the layer 7 application. This type of application-aware approach to bandwidth management will also ensure a better user experience with zero buffering through better video playback. At this stage in the game, there is little doubt that SDN is becoming a reality in operator networks.

d. Content Availability

There will be content servers used for media delivery or caching, on a service-provider edge network. These are installed by the content delivery network or operator service providers. Content that is to be served to the users is distributed and preoccupied across multiple content servers and also across various geographies in some cases.

SDN apps will be able to provision flows in the network based on the availability and types of the content which is built to handle content availability. SDN applications can also check the availability of the content in the content servers before routing requests to servers. A content-routing application will provide intelligence on its availability along with enabling discovery of content in the content servers.

This intelligence can be further used to route requests to the correct servers wherever the content is residing. Therefore, SDN application will direct requests from those websites which are non-cache-able and that generate active content to a server that provides active content rather than a caching server which significantly reduces network discontinuation.

e. Regulation and Compliance-Bound Applications

Major cloud vendors are now providing the capability to work and store with compliance and regulation-bound workloads. Now organizations have the option to extend architectures which have initially been very limited because of regulation into the cloud and distributed environments. How can you segment the traffic? How can you ensure that regulation and compliance workloads are persistently monitored and secured? Here SDN can be a great help for you.

Network points, network traffic travelling between switches, and even hypervisors can be controlled in SDN architecture. You should also remember that this layer abstracts virtual hardware and functions controls. This powerful layer can then span various virtualization points, locations, and even cloud locations.

f. High –Performance Applications

We are all seeing a rise in new types of application technologies. The delivery of rich apps like graphics design software, engineering, CAD, and GIS is allowed by virtualization. Traditionally, these workloads are required bare-metal architectures with their own connections. However, with the help of virtualization, VDI can help in creating powerful desktop experiences and applications are streamed. We can also see the integration of SDN into application control at the network layer. All of these functions like segmenting heavy traffic, securing confidential data, creating powerful QoS policies, and even creating threshold alerts around bottlenecks within SDN will help to support rich and high-performance applications which are being delivered through virtualization.

g. Distributed Application Control and Cloud Integration

The capability to extend across the entire data centre is one of the most significant benefits of SDN. This type of agility integrates distributed cloud, locations and the organization as a whole. SDN also allows for critical network traffic to pass between various locations irrespective of the type of underlying network architecture. You also permit easier movement of data between cloud locations and data centre by abstracting critical network controls. You can utilise powerful APIs to not only integrate with a cloud provider, but you can also control specific network services as well because SDN is a form of network virtualization. While keeping your business agile, this allows you to manage your workloads granularly.

UNIT 9

9.0 SECURITY USING THE SDN PARADIGM

SDN architecture may enable, facilitate or enhance network-related security applications due to the controller's central view of the network, and its capacity to reprogram the data plane at any time. While security of SDN architecture itself remains an open question that has already been studied a couple of times in the research community.

Several researches work on SDN have already investigated security applications built upon the SDN controller, with different aims in mind. Distributed Denial of Service (DDoS) detection and mitigation, as well as botnet and worm propagation, are some concrete use-cases of such applications: basically, the idea consists in periodically collecting network statistics from the forwarding plane of the network in a standardized manner (e.g., using Openflow), and then apply classification algorithms on those statistics in order to detect any network anomalies. If an anomaly is detected, the application instructs the controller how to reprogram the data plane in order to mitigate it.

Another kind of security application leverages the SDN controller by implementing some moving target defence (MTD) algorithms. MTD algorithms are typically used to make any attack on a given system or network more difficult than usual by periodically hiding or changing key properties of that system or network. In traditional networks, implementing MTD algorithms is not a trivial task since it is difficult to build a central authority able of determining - for each part of the system to be protected - which key properties are hiding or changed. In an SDN network, such tasks become more straightforward thanks to the centrality of the controller. One application can for example periodically assign virtual IPs to hosts within the network, and the mapping virtual IP/real IP is then performed by the controller. Another application can simulate some fake opened/closed/filtered ports on random hosts in the network in order to add significant noise during reconnaissance phase (e.g., scanning) performed by an attacker.

Additional value regarding security in SDN enabled networks can also be gained using FlowVisor and FlowChecker respectively. The former tries to use a single hardware forwarding plane sharing multiple separated logical networks. Following this approach, the same hardware resources can be used for production and development purposes as well as separating monitoring, configuration and internet traffic, where each scenario can have its own logical topology which is called slice. In conjunction with this approach FlowChecker realizes the validation of new OpenFlow rules that are deployed by users using their own slice. SDN controller applications are mostly deployed in large-scale scenarios, which requires comprehensive checks of possible programming errors.

Many security issues related to the traditional network architecture also apply to the SDN architecture. Unfortunately, the new features that provide great flexibility, real-time programmability and simplified controls through the centralized SDN controller also introduce new security challenges. In fact, SDN is exposed to various sources of security risk from its network architecture design perspective, which includes the control plane, application plane and data plane layers.

One of the most significant security risk factors is the possibility of a compromised SDN controller attack at the control plane layer. Due to the centralization design of the SDN, the SDN controller becomes the brain of the SDN architecture. Attackers can focus on compromising the SDN controller in an attempt to manipulate the entire network. If the attacker successfully gains access, the compromised SDN controller can be used to direct the network devices it controls (e.g., switches) to drop all incoming traffic or launch serious attacks against other targets, such as sending useless traffic to a victim to deplete its resources. To mitigate this security risk, it is critical to harden the operating system that hosts the SDN controller and prevent unauthorized access to the SDN controller. Furthermore, the control plane layer is susceptible to a distributed denial-of-service (DDoS) attack. SDN switches may cause the SDN controller to be flooded with many queries that may potentially cause a delay or drop of queries. One possible defence against a DDoS attack is to implement multiple physical SDN controllers instead of just one. When switches are connected to multiple SDN controllers, one of these controllers can act as the master of the switches. When this master controller needs to process a high load of queries, it can direct the load to other lightly loaded controllers to be the master for some of its assigned switches. This keeps the load balanced among the SDN controllers, which mitigates DDoS attacks.

If attackers compromise the SDN controller, they can hack the SDN applications to manipulate security applications to reprogram the network traffic flow through the SDN controller.

At the data plane layer, switches are vulnerable to denial-of-service (DoS) attacks as well. A malicious user can flood the switches with large payloads, causing legitimate packets to be dropped when a switch’s buffering capability is exceeded. There are many ways to address this attack, including proactive rule caching, rule aggregation and decreasing the switch-to-SDN-controller communication delay. Also, increasing the switch's buffering capability can mitigate the risk of a DoS attack.

Communicating messages between the control plane layer and the data plane layer is subject to man-in-the-middle attacks. The attacker can potentially modify rules sent from the SDN controller to switches to take control of the switches. One of the most effective solutions to such attacks is to encrypt the messages with the use of digital signatures for securing and proofing the integrity and authenticity of the messages.

The real-time programmability is also open to serious vulnerability at the application plane layer. Specifically, if the attacker can hack the SDN security applications, it can manipulate the network traffic flow through the SDN controller. If the SDN applications are compromised, the whole network is, too. To effectively mitigate such security risk, it is critical that security coding practices be enforced with comprehensive change management and integrity check processes as part of the software development life cycle.

9.1 Security and the SDN Controller

To make the most of the controller's potential to positively impact network security and guard against the controller becoming the prime attack surface, requires Including security in your design plans right from the beginning to avoid problems down the road.

Just one of the enterprise network management and security possibilities with the SDN controller is its ability to push global security policy updates out centrally across the network. This makes network edge packet filtering possible via a virtualized switch for suspicious traffic redirection to other security devices for more analysis. SDN controller programmability gives engineers the power to install northbound interface security applications that create new ways for applying network security policies.

The major attack vulnerability in the controller requires that access be tightly controlled as a fundamental means of preventing unauthorized activity. To further secure the controller requires:

- Auditing, reviewing, monitoring, and updating role-based access policies
- A high-availability controller architecture for distributed denial-of-service (DDoS) attack prevention
- Encrypting northbound communication via Transport Layer Security (TLS) or SSH with secure coding of northbound applications

- Eliminating default application password use and implement application authentication for controller communication approval
- The use of TLS to authenticate endpoints for southbound communication
- Segregating control protocol traffic from the primary data flows via an out-of-band network.

SDN architecture is only as secure as the design, so IT groups should be prepared to use additional methods for securing the network, such as a holistic integration of third-party security solutions.

9.2. Strengthening SDN Security

SDN and network security require that network managers and admins learn how to appropriately set security levels to meet the dynamic needs of the environment. An example would be better leveraging of network telemetry data or flow-capture data that can be used to spotanomalies. This provides the ability to dynamically establish specific flow rules that divert flows to centralized or multiple enforcement points.

SDN can also be used for traffic engineering to direct network flows to specific security services or devices. These could include firewalls, intrusion detection systems/intrusion prevention systems (IDS/IPS), and web application firewalls (WAFs).

Another way that SDN can provide greater security is through micro-segmentation where policies are applied to individual workloads for greater attack resistance. By applying more granular segmentation to data centre workloads, organizations can decrease the network's attack surfaces. In complex networks, east-west, as well as north-south traffic, can pose a security risk.

Solutions like SD-Access further facilitate centralized policies for all network switches so that security policies follow users wherever they move about your environment.

This level of granular segmentation makes it possible to increase documentation, integrate applications with network monitoring and add security features quickly and accurately.

Although SDN is emerging as the most promising option for enterprise network management and protection, it should be seen by enterprises as the foundation on which to build rather than an end in itself. This is because as the network changes to meet evolving enterprise needs, it must also evolve to meet new threats and attack vectors.