

**APPLICATION AWARE
ROUTING IN SDN**

*Dissertation/ Project report submitted in partial fulfilment of the
requirement for the degree of*

**BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION
ENGINEERING**

By

RISHABH NAINWAL (131054)

ADITYA MAHAJAN (131087)

UTKARSH KHURANA (131092)

UNDER THE GUIDANCE OF

Dr. RAJIV KUMAR



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

May 2017

TABLE OF CONTENTS

	Page Number
DECLARATION BY THE SCHOLAR	3
SUPERVISOR'S CERTIFICATE	4
ACKNOWLEDGEMENT	5
LIST OF ABBREVIATIONS AND ACRONYMS	6
LIST OF FIGURES	7
LIST OF TABLES	8
ABSTRACT	9
CHAPTER – 1	
INTRODUCTION	10
1.1 TRADITIONAL NETWORK	
1.1.1MULTI=MEDIA	10
1.1.2 MULTI-POINT	11
1.1.3 MULTI-RATE	11
CHAPTER – 2	
LITERATURE SURVEY	14
CHAPTER – 3	
SOFTWARE DEFINED NETWORKING	18
3.1 INTRODUCTION TO SOFTWARE DEFINED NETWORKING	18
3.2 WORKING OF SOFTWARE DEFINED NETWORKING	20

3.3 HOW SDN IS DIFFERENT?	22
3.3.1 TRADITIONAL NETWORK CONFIGURATION	22
3.3.2 SOFTWARE DEFINED NETWORKING	24
3.4 ARCHITECTURE OF SDN	26
3.5 ADVANTAGES OF SDN	28
CHAPTER – 4	
OUR IMPLEMENTATION	29
4.1 C++ IMPLEMENTATION	31
CHAPTER – 5	
RESULTS AND OUTPUT	33
CHAPTER – 6	
APPLICATIONS OF SDN	36
6.1 Software Defined Networking Controllers and Applications	36
6.1.1 SDMN	37
6.1.2 SD-WAN	38
6.1.3 SD-LAN	38
6.1.4 Security using the SDN paradigm	38
CHAPTER – 7	
FUTURE OF SDN	41
REFERENCES	44

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the B-Tech thesis entitled **“Application aware routing in Software Defined Routing”** submitted at **Jaypee University of Information Technology, Wagnaghat, India**, is an authentic record of my work carried out under the supervision of Dr. **Rajiv Kumar**.

I have not submitted this work elsewhere for any other degree or diploma.

Rishabh Nainwal (131054)

Aditya Mahajan (131087)

Utkarsh Khurana (131092)

Department of Electronics and Communication

Jaypee University of Information Technology, Wagnaghat, India

Dated

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the B-Tech. entitled “**Application aware routing in Software Defined Routing**”, submitted by **Rishabh Nainwal, Aditya Mahajan and Utkarsh Khurana** at **Jaypee University of Information Technology, Wagnaghat, India** is a bonafide record of his / her original work carried out under my supervision.

This work has not been submitted elsewhere for any other degree or diploma.

(Dr. Rajiv Kumar)

Associate Professor

Dated

ACKNOWLEDGEMENT

We would like to show our gratitude to everyone who helped us make this project successful, our teachers, support staff, fellow students. Also, special mention of the following must be made.

First of all, we would like to thank our project guide **Dr. Rajiv Kumar** who helped us throughout this project and supported us. We are very thankful to him for giving his valuable time and attention to our project and helping us complete our project in time.

We are also thankful to our H.O.D. Dr. Sunil Bhooshan for giving us an opportunity to work on this project. Finally, we would like to thank all the members of staff who helped us and the lab faculty for allowing us to make use of lab equipment.

LIST OF ABBREVIATIONS AND ACRONYMS

SDN	Software Defined Networking
CLI	Command line interference
TCP	Transmission control protocol
UDP	User datagram protocol
HTTP	Hyper Text Transfer Protocol
FTP	File transfer protocol
ICMP	The Internet Control Message Protocol
DPI	Deep Packet Inspection

LIST OF FIGURES

Figure Number	Caption	Page Number
1.1	A traditional network architecture	13
3.1	A Software defined network architecture	20
3.2	Structure of Software defined Networking	22
3.3	An example of traditional network with switches and access points	25
3.4	A high-level overview of the software-defined networking architecture	27
4.1	Topology used for our implementation	31
5.1	Output window of our implementation	34
6.1	Brocade SDN Controller	37
6.2	Brocade flow optimizer	37
6.3	Brocade flow manager	37

LIST OF TABLES

TABLE 1.1 Network traffic types and their requirement	14
---	----

ABSTRACT

When the client transmits an IP packet, the switch inspects the packet and depending on the policy/rule installed in it, forwards the packet on a particular route.

If the switch doesn't have any policy/rule installed, it sends the packet to the controller. The controller inspects the packet header and/or the payload, determines the type of packet (TCP, UDP, HTTP, etc.), and installs a policy/rule on the switch instructing it to forward packets along a particular route.

CHAPTER– 1

INTRODUCTION

1.1 Traditional Networks

An ideal telecommunication network has following characteristics: broadband, multi-media, multi-point, multi-rate and economical implementation for a diversity of services (multi-services).

Conventional telephony communication used:

- the voice medium only,
- connected only two telephones per telephone call, and
- used circuits of fixed bit-rates.

Modern services can be:

- Multimedia,
- multi-point, and
- multi-rate.

These aspects are examined individually in the following three sub-sections

1.1.1 Multi-media

A multi-media call may communicate audio, data, still images, or full-motion video, or any combination of these media. Each medium has different demands for communication quality, such as:

- bandwidth requirement,
- Signal latency within the network, and

- signal fidelity upon delivery by the network.

The information content of each medium may affect the information generated by other media. For example, voice could be transcribed into data via voice recognition, and data commands may control the way voice and video are presented. These interactions most often occur at the communication terminals, but may also occur within the network.

1.1.2 Multi point

Traditional voice calls are predominantly two party calls, requiring a point-to-point connection using only the voice medium. To access pictorial information in a remote database would require a point-to-point connection that sends low bit-rate queries to the database and high bit-rate video from the database. Entertainment video applications are largely point-to-multi-point connections, requiring one-way communication of full motion video and audio from the program source to the viewers. Video teleconferencing involves connections among many parties, communicating voice, video, as well as data. Offering future services thus requires flexible management of the connection and media requests of a multi-point, multi-media communication call.

1.1.3 Multi-rate

A multi-rate service network is one which flexibly allocates transmission capacity to connections. A multi-media network has to support a broad range of bit-rates demanded by connections, not only because there are many communication media, but also because a communication medium may be encoded by algorithms with different bit-rates. For example, audio signals can be encoded with bit-rates ranging from less than 1 kbit/s to hundreds of kbit/s, using different encoding algorithms with a wide range of complexity and quality of audio reproduction. Similarly, full motion video signals may be encoded with bit-rates ranging from less than 1 Mbit/s to hundreds of Mbit/s. Thus a network

transporting both video and audio signals may have to integrate traffic with a very broad range of bit-rates.

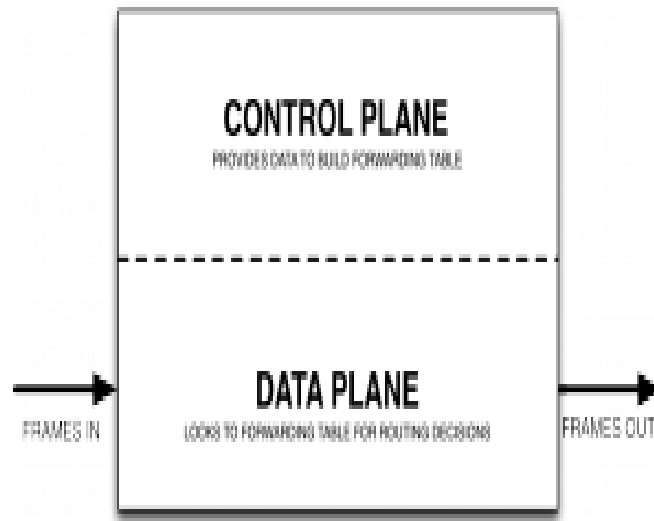


Figure 1.1: A traditional network architecture

Traditionally, the various services were carried via separate networks: voice on the telephone network, data on computer networks such as local area networks, video teleconferencing on private corporate networks, and television on broadcast radio or cable networks.

These networks were largely engineered for a specific application and are not suited to other applications. For example, the traditional telephone network is too noisy and inefficient for bursty data communication. On the other hand, data networks which store and forward messages using computers had limited connectivity, usually did not have sufficient bandwidth for digitised voice and video signals, and suffer from unacceptable delays for the real-time signals. Television networks using radio or cables were largely broadcast networks with minimum switching facilities.

The types of traffic found in a broadband network (with examples) and their respective requirements are summarised in Table 1.

Table 1: Network traffic types and their requirements

Traffic type	Example	Required bandwidth	Cell-delay	Latency
Constant	Voice, guaranteed circuit emulation	Minimal	Low	
Variable	Compressed video	Guaranteed	Variable	Low
Available	Data	Not guaranteed	Widely variable	Variable

CHAPTER– 2

LITERATURE SURVEY

OBJECTIVES

The objectives of APPLICATION AWARE ROUTING are as follows:

- 1) Single routing software for a cluster of switches as opposed to spawning a separate stack for each switch.
- 2) Routing based on type of application served by the network as opposed to considering only destination network.
- 3) Performing a better differential routing which increases the utilization of links in the network.

The controller installs a rule on the switch, instructing it to forward the packets along the different paths depending on the type of the packet. The controller inspects the Ethernet packet for IP content and further for TCP, UDP and ICMP content based on the protocol header. TCP packets are further inspected for HTTP and FTP traffic by performing a string search in the payload for keywords (HTTP 1.1 for HTTP traffic and FTP/SFTP for FTP traffic).

Traditional networks were designed to forward packets from source to destination using the shortest route possible. Routers and switches were mostly agnostic to applications being served by the network. In this kind of network, a service provider may not be able to distinguish between a user watching video and another browsing the web. In order to provide service differentiation and increase user experience, today's networks are forced to be application-aware. Software-Defined Networking (SDN) which is an emerging concept in networking forefront presents an opportunity to service providers to control their networks by means of a programmable interface. This dissertation aims at developing an application-aware routing system for SDN which does differential routing based on type of application to which the traffic belongs to. Deploying such SDN application will help service providers to

dynamically provision the network switches based on application characteristics and requirements, leading to a better overall user experience and reduction in bandwidth wastage.

Software defined networking is seen as an evolutionary paradigm shift, but still faces several challenges. The covered areas of challenges and effects provide a general view of what may slow down further development and what is the possibility when the technology is integrated successfully. A certain lack of know-how, combined with high complexity when it comes to integration into traditional networks, are main reasons for a delayed diffusion of the technology. Furthermore, the analyzed papers mostly describe software defined networking on a very detailed mathematical and technological basis, making it very hard for enterprises and organizations to assess if the technology can have a specific business impact (e.g. on increasing efficiency or reducing costs). Nevertheless, the steady increase of users and requirements leave providers faced with the need to rethink the usage of current network technologies in order to stay competitive and profitable. As the literature review has revealed, the separation of the control and data plane offers great benefits, such as easier management, enhanced features, like dynamic deployment of virtual networks as well as economic factors. Besides outlining the technical aspects these benefits should play an important role in further research, especially in the IS domain.

A key benefit of Software Defined Networks is fine-grained management of network flows made possible by the execution of flow-specific actions based upon inspection and matching of various packet fields. However, current switches and protocols limit the inspected fields to layer 2-4 headers and hence any customized flow-handling that uses higher-layer information necessitates sending the packets to the controller. This is inefficient and slow, adding several switch-to-controller round-trip delays. This report proposes an extended SDN architecture that enables fast customized packet-handling even when the information used is not restricted to L2-L4. We describe an implementation of this architecture that keeps most of the processing in the data plane and limits the need to send packets to the controller even when higher-layer information is used in packet-

handling. We show how some popular applications can be implemented using this extended architecture and evaluate the performance of one such application using a prototype implementation on Open vSwitch. The results show that the proposed architecture has low overhead, good performance and can take advantage of a flexible scale-out design for application deployment.

Moreover, existing implementations seek to achieve Interior Routing using existing other protocols (like OSPF/ISIS) from stacks like Quagga/XORP. The problem with that approach is resource (CPU/Memory) exhaustion since an entire stack (Quagga/XORP) or a Virtual Machine (VM) is spawned for each switch in the network. This may not be a concern for privately managed small-sized LAN/WAN. But this poses serious concern when scaling the network with hundreds and thousands of switches (like AS-wide deployment for a service provider). APPLICATION AWARE ROUTING aims to address this issue by being a light-weight plugin to controller and still be able to handle a cluster of switches. In addition, these approaches cannot perform Traffic Engineered (TE) routing as they depend only upon source IP and destination IP of the packet being routed. Google's B4 does a TE routing in order to maximize link utilization. But B4 also depend upon source IP and destination IP and does not look into transport layer ports. Hence packets having same SIP and DIP will be routed via the same path irrespective of the application that generated those packets. Though this approach provides a way to increase link utilization, it does not provide any mechanism for service differentiation.

In spite of recent and interesting attempts to survey this new chapter in the history of networks, the literature was still lacking, to the best of our knowledge, a single extensive and comprehensive overview of the building blocks, concepts, and challenges of SDNs. Trying to address this gap, the present study used a layered approach to methodically dissect the state of the art in terms of concepts, ideas and components of software-defined networking, covering a broad range of existing solutions, as well as future directions. We started by comparing this new paradigm with traditional networks and discussing how academy and industry helped shape software-defined networking. SDN has successfully managed to pave the way towards a next generation networking, spawning an innovative research and development environment, promoting advances in several areas: switch and controller

platform design, evolution of scalability and performance of devices and architectures, promotion of security and dependability. We will continue to witness extensive activity around SDN in the near future. Emerging topics requiring further research are, for example: the migration path to SDN, extending SDN towards carrier transport networks, realization of the network as-a-service cloud computing paradigm, or software-defined environments (SDE).

CHAPTER – 3

SOFTWARE DEFINED NETWORKING

3.1 Introduction to Software Defined Networking

Software-defined networking (SDN) is an architecture purporting to be dynamic, manageable, cost-effective, and adaptable, seeking to be suitable for the high-bandwidth, dynamic nature of today's applications. SDN architectures decouple network control and forwarding functions, enabling all the network controls to become directly programmable and the underlying infrastructure to be abstracted from applications and network services.

Software-defined networking (SDN) is an umbrella term encompassing several kinds of network technology aimed at making the network as agile and flexible as the virtualized server and storage infrastructure of the modern data centre. The goal of SDN is to allow network engineers and administrators to respond quickly to changing business requirements. In a software-defined network, a network administrator can shape traffic from a centralized control console without having to touch individual switches, and can deliver services to wherever they are needed in the network, without regard to what specific devices a server or other device is connected to. The key technologies are functional separation, network virtualization and automation through programmability.

Software-defined networking (SDN) is an approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behaviour dynamically via open interfaces and abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional networks doesn't support the dynamic, scalable computing and storage needs of more modern computing environments such as data centres. This is done by decoupling or disassociating the system that makes decisions about

where traffic is sent (the SDN controller, or control plane) from the underlying systems that forward traffic to the selected destination (the data plane).

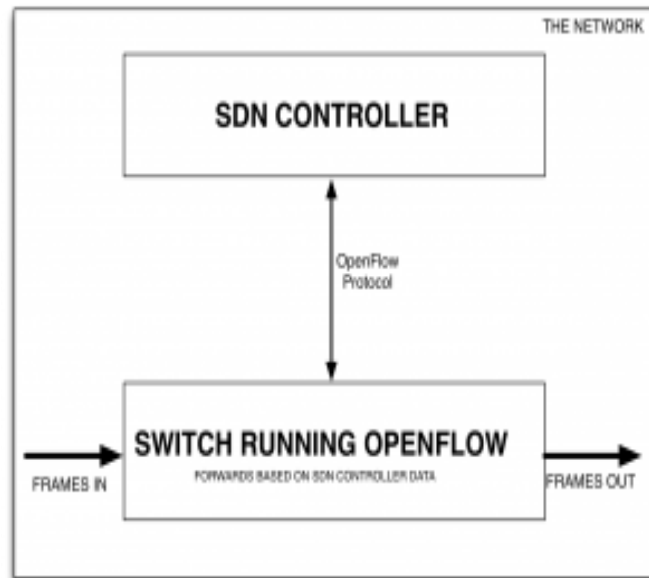


Figure 3.1: A Software defined network architecture

SDN focus solely on separation of the control plane of the network, which makes decisions about how packets should flow through the network from the dataplane of the network, which actually moves packets from place to place. When a packet arrives at a switch in the network, rules built into the switch's proprietary firmware tell the switch where to forward the packet. The switch sends every packet going to the same destination along the same path, and treats all the packets the exact same way. In a classic SDN scenario, rules for packet handling are sent to the switch from a controller, an application running on a server somewhere, and switches (also known as data plane devices) query the controller for guidance as needed, and provide it with information about traffic they are handling. Controllers and switches communicate through a controller's south bound interface, usually OpenFlow, although other protocols exist.

Where a traditional network would use a specialized appliance such as a fire wall or link-load balancer ,an SDN deploys an application that uses the controller to manage

dataplane behavior. Applications talk to the controller through its north-bound interface. As of the end of 2014, there is no formal standard for the application interface of the controller to match OpenFlow as a general south-bound interface. It is likely that the OpenDaylight controller's northbound application program interface (API) may emerge as a defacto standard over time, given its broad vendor support.

Software defined networking uses an operation mode that is sometimes called adaptive or dynamic, in which a switch issues a route request to a controller for a packet that does not have a specific route. This process is separate from adaptive routing, which issues route requests through routers and algorithms based on the network topology, not through a controller.

With SDN, the administrator can change any network switch's rules when necessary -- prioritizing, deprioritizing or even blocking specific types of packets with a very granular level of control. This is especially helpful in a cloud computing multi-tenant architecture, because it allows the administrator to manage traffic loads in a flexible and more efficient manner. Essentially, this allows the administrator to use less expensive commodity switches and have more control over network traffic flow than ever before.

3.2 Working of Software defined networking

Software-defined networking providers offer a wide election of competing architectures, but at its most simple, the Software Defined Networking method centralizes control of the network by separating the control logic to off-device computer resources. All SDN models have some version of an SDN Controller, as well as southbound APIs and northbound APIs:

- **Controllers:** The brains of the network, SDN Controllers offer a centralized view of the overall network, and enable network administrators to dictate to the underlying systems (like switches and routers) how the forwarding plane should handle network traffic.

- **Southbound APIs:** Software-defined networking uses southbound APIs to relay information to the switches and routers “below.” OpenFlow, considered the first standard in SDN, was the original southbound API and remains as one of the most common protocols. Despite some considering OpenFlow and SDN to be one in the same, OpenFlow is merely one piece of the bigger SDN landscape.
- **Northbound APIs:** Software Defined Networking uses northbound APIs to communicate with the applications and business logic “above.” These help network administrators to programmatically shape traffic and deploy services.

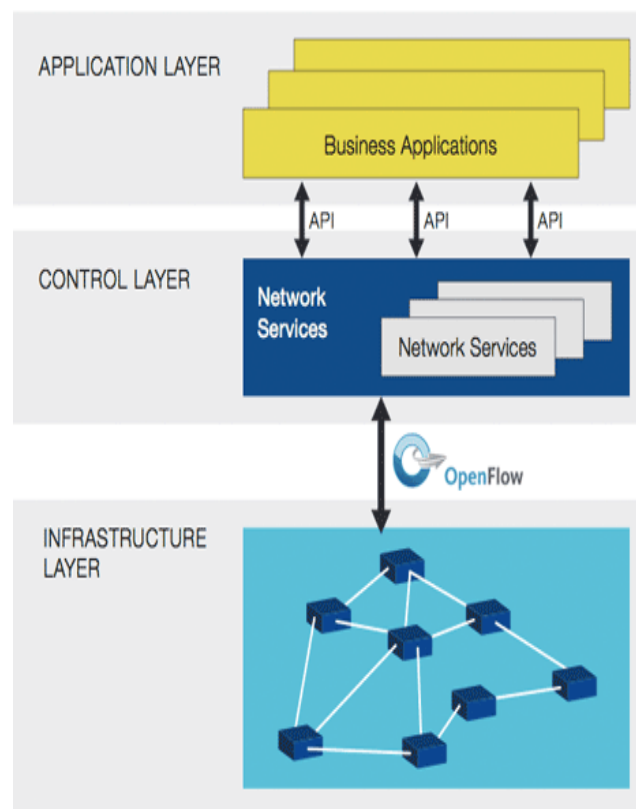


Figure 3.2: Structure of Software defined Networking

3.3 How SDN is different?

3.3.1 Traditional network configuration

The traditional approach to networking is characterized by two main factors:

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

2) Most functionality within this appliance is implemented in dedicated hardware. An Application Specific Integrated Circuit (or: ASIC) is often used for this purpose.

Organizations are increasingly confronted with the limitations that accompany this hardware-centric approach, such as:

- Traditional configuration is time-consuming and error-prone

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

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

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

- Multi-vendor environments require a high level of expertise

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

- Traditional architectures complicate network segmentation

A development further complicating networking matters, is the connectivity evolution that is currently taking place. In addition to tablets, PCs and smartphones, other devices such as alarm systems and security cameras will soon be linked to the internet. The predicted explosion of smart devices is accompanied by a new challenge for organizations: how to incorporate all these devices of different vendors within their network in a safe and structured manner. Many traditional networks place all types of devices in the same 'zone'. In case of a compromised device, this design risks giving external parties access to the entire network. This can be hackers exploit the internet connection of smart devices or vendors who can remotely log onto 'their' devices. In both cases, there is no apparent reason for giving them access to all network components. However, the administrative hassle described earlier makes network segmentation a complex process and quickly leads to network clutter.

In conclusion, to overcome these and other traditional networking limitations, the time has come to introduce a new perspective on network management.

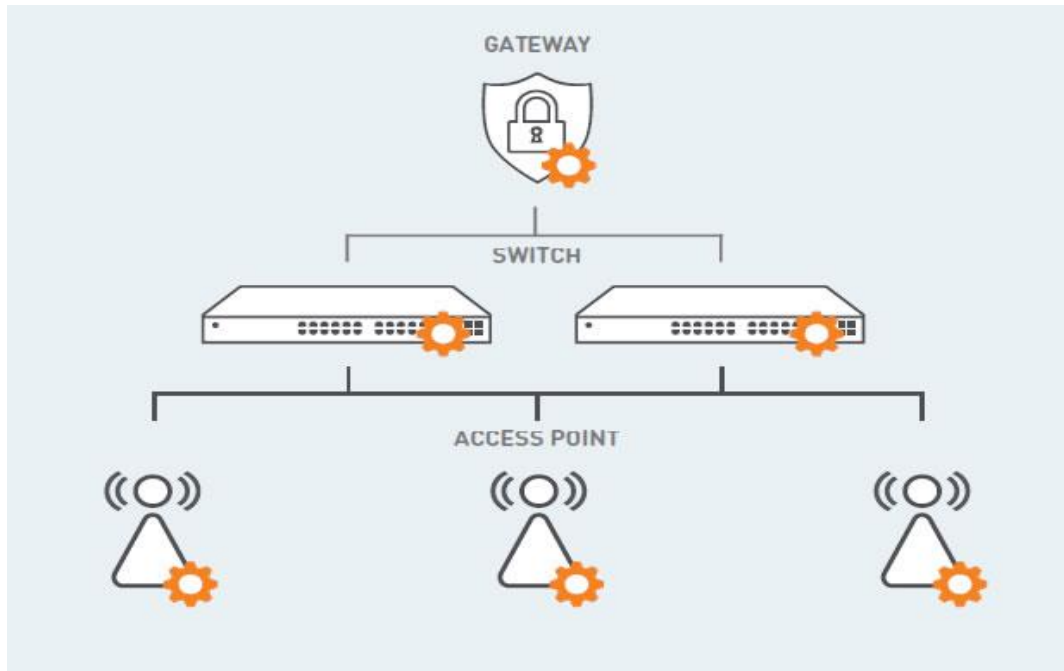


Figure 3.3: An example of traditional network with switches and access points

3.3.2 Software Defined Networking

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

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

- 1) The plane that determines where to send traffic (control plane)
- 2) The plane that executes these decisions and forwards traffic (data plane)

Decoupling these two planes involves leaving the data plane with network hardware and moving the control plane into a software layer. By abstracting the network from the hardware, policies no longer have to be executed on the hardware itself. Instead, the use of a centralized software application functioning as the control plane makes network virtualization possible. This process is similar to server virtualization:

- **Server virtualization**

Server virtualization is the masking of server resources, including the number and identity of individual physical servers, processors, and operating systems, from server users. The server administrator uses a software application to divide one physical server into multiple isolated virtual environments. The virtual environments are sometimes called virtual private servers, but they are also known as guests, instances, containers or emulations.

- **Network virtualization**

Network virtualization is an SDN technology application that creates unmatched network agility and dramatically reduces costs of network operations by automating network provisioning for both increasingly dynamic virtual workloads as well bare metal workloads. A network virtualization program eliminates the conventional shortcomings and time consuming provisioning tasks related to legacy network segmentation technologies, like switched VLANs, routed subnets, and firewall ACLs. Alternatively, an SDN-based network virtualization application supports arbitrary assignment of IP/MAC addressing schemes, while at the same time automating network configuration tasks and enforcing expected network segmentation. Network virtualization leverages the OpenFlow protocol to dynamically and automatically provision virtual network segments and virtual routing services on both physical and virtual networking devices. You can automatically provision network security policies through a cloud orchestration platform, like OpenStack, or automatically assign workloads according to workload attributes like MAC, subnet, VLAN, IP protocol, or other attributes. Network virtualization applications eliminate the time consuming provisioning process required with legacy networking gear, enable unmatched network agility, and dramatically reduce the cost of network operations.

3.4 Architecture of SDN

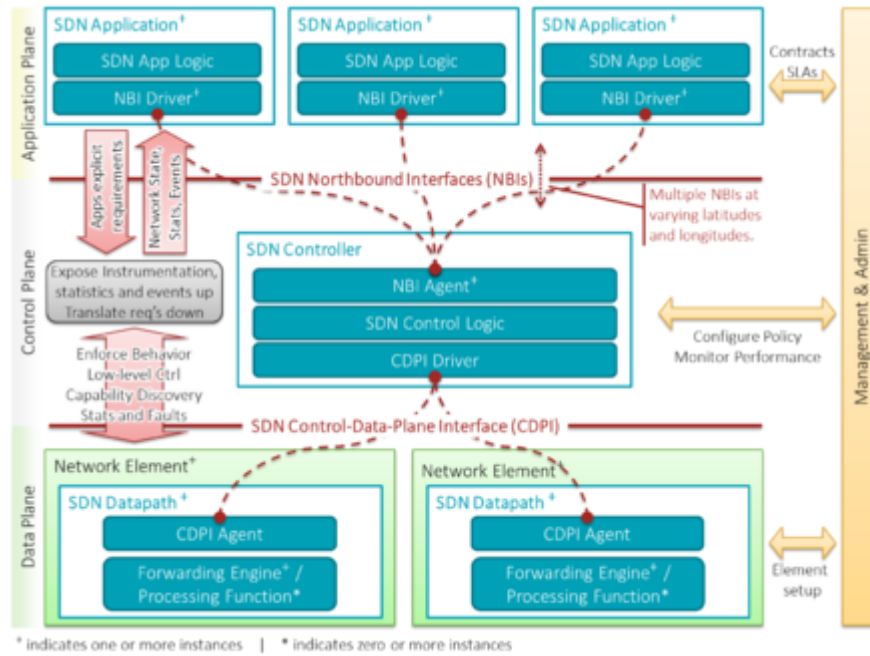


Figure 3.4: A high-level overview of the software-defined networking architecture

The following list defines and explains the architectural components:

- **SDN Application**

SDN Applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to the SDN Controller via a northbound interface (NBI). In addition they may consume an abstracted view of the network for their internal decision-making purposes. An SDN Application consists of one SDN Application Logic and one or more NBI Drivers. SDN Applications may themselves expose another layer of abstracted network control, thus offering one or more higher-level NBIs through respective NBI agents.

- **SDN Controller**

The SDN Controller is a logically centralized entity in charge of (i) translating the requirements from the SDN Application layer down to the

SDN Datapaths and (ii) providing the SDN Applications with an abstract view of the network (which may include statistics and events). An SDN Controller consists of one or more NBI Agents, the SDN Control Logic, and the Control to Data-Plane Interface (CDPI) driver. Definition as a logically centralized entity neither prescribes nor precludes implementation details such as the federation of multiple controllers, the hierarchical connection of controllers, communication interfaces between controllers, nor virtualisation or slicing of network resources.

- **SDN Datapath**

The SDN Datapath is a logical network device that exposes visibility and uncontested 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, which can include OSI layer 4-7 functions.

- **SDN Control to Data-Plane Interface (CDPI)**

The SDN CDPI is the interface defined between an SDN Controller and an SDN Datapath, which provides at least (i) programmatic control of all forwarding operations, (ii) capabilities advertisement, (iii) statistics

reporting, and (iv) event notification. One value of SDN lies in the expectation that the CDPI is implemented in an open, vendor-neutral and interoperable way.

- **SDN Northbound Interfaces (NBI)**

SDN NBIs are interfaces between SDN Applications and SDN Controllers and typically provide abstract network views and enable direct expression of network behavior and requirements. This may occur at any level of abstraction (latitude) and across different sets of functionality (longitude). One value of SDN lies in the expectation that these interfaces are implemented in an open, vendor-neutral and interoperable way.

3.5 Advantages of Software Defined Networking

- **Directly programmable:** Network control is directly programmable because it is decoupled from forwarding functions.
- **Agile:** Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- **Centrally managed:** Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- **Programmatically configured:** SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- **Vendor-neutral and Open standards-based:** When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

CHAPTER – 4

OUR IMPLEMENTATION

The goal of SDN is to allow network engineers and administrators to respond quickly to changing business requirements. In a software-defined network, a network administrator can shape traffic from a centralized control console without having to touch individual switches, and can deliver services to wherever they are needed in the network, without regard to what specific devices a server or other device is connected to. The key technologies are functional separation, network virtualization and automation through programmability.

The OER Application-Aware Routing: PBR feature introduces the capability to optimize traffic based on portions of an IP packet, other than the destination address. Independent Optimized Edge Routing (OER) policy configuration is applied to only a subset of the traffic carried by the monitored prefix. You can use this feature to apply very granular OER policy configuration based on the type of application or IP packet, without changing OER policy configuration for other traffic that is carried by the monitored prefix. The master controller uses policy-based routing (PBR) to send the subset of traffic to the external interface that conforms to the independent policy configuration.

We have created different paths for TCP, UDP and ICMP traffic. We have further classified TCP traffic into HTTP and FTP traffic. For each different type of traffic, we have assigned a dedicated path.

Our implementation could easily be extended to be adaptive, so that the features of an application-aware network mentioned above could be realized.

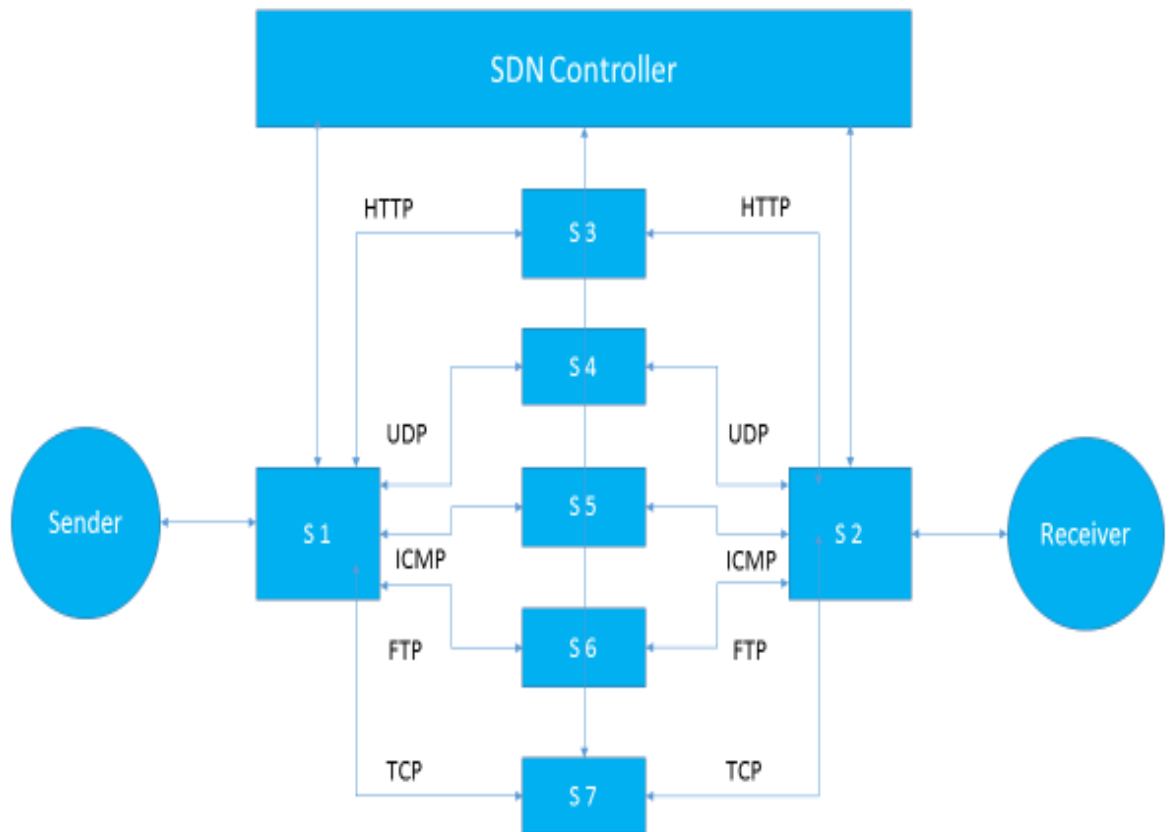


Figure 4.1: Topology used for our implementation

When the client transmits an IP packet, the switch inspects the packet and depending on the policy/rule installed in it, forwards the packet on a particular route.

If the switch doesn't have any policy/rule installed, it sends the packet to the controller. The controller inspects the packet header and/or the payload, determines the type of packet (TCP, UDP, HTTP, etc.), and installs a policy/rule on the switch instructing it to forward packets along a particular route.

4.1 C++ Implementation

When the switch forwards the packet it receives to the controller, the controller checks the Ethernet packet for IP content.

- If it's indeed an IP packet, the controller then checks the protocol header to determine if it's a TCP, UDP or an ICMP packet.
 - If the packet is a TCP packet, then it further determines if the packet is HTTP or FTP by inspecting the port number.
- After the controller determines the type of packet, it instructs the switch to send the packet along a specific path/route.
- According to our topology -
 - HTTP packets are routed along S3.
 - FTP packets are routed along S6.
 - Other TCP packets are routed along S7.
 - UDP packets are routed along S4.
 - ICMP packets are routed along S5.

We identify different packets as being TCP/UDP using the identifier in the IP header for the transport protocol, after IP packets themselves are identified using an identifier in the Ethernet header.

The applications are identified using the server port being contacted by the client. We class packets as belonging to an FTP connection if the server port number is 21, and as HTTP if the server port being contacted is 8000.

We have also performed Deep Packet Inspection (DPI) using C++, where a similar topology was used. The controller installs a rule on the switch, instructing it to forward the packets along the different paths depending on the type of the packet.

The controller inspects the Ethernet packet for IP content and further for TCP, UDP and ICMP content based on the protocol header.

TCP packets are further inspected for HTTP and FTP traffic by performing a string search in the payload for keywords (HTTP 1.1 for HTTP traffic and FTP/SFTP for FTP traffic).

Only the first packet from a flow is sent to the controller, as in Pyretic/POX. Thereafter, a rule is installed on the switch for the flow. This allows the flow to be dealt with by the switch itself, saving the controller from having to deal with too many packets.

CHAPTER – 5

RESULTS AND OUTPUT

```
CAUsers\Marcus\documents\visual studio 2013\Projects\CS6250nwk\Simulation\Debug\CS6250nw...
switch 1 sending data to receiver through switch 7
Switch 7: Data received, forwarding data to receiver

source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
Switch 1 sending data to receiver through switch 7
Switch 7: Data received, forwarding data to receiver

source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
Switch 1 sending data to receiver through switch 7
Switch 7: Data received, forwarding data to receiver

enter flow type
Flow_http_tcp

enter number of packets in flow
3
source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
-----
Controller called
setting rules
asking controller

setting new rules for http forwarding

switch 1 sending data to receiver through switch 3
Switch 3: Data received, forwarding data to receiver

source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
Switch 1 sending data to receiver through switch 3
Switch 3: Data received, forwarding data to receiver

source ip from packet is
192.168.2.101
destination ip from packet is
199.59.150.42
Switch 1 sending data to receiver through switch 3
```

Figure 5.1:Output window of our implementation

We concluded that the packets were received at the particular node from the particular source as defined in our topology

According to our topology the following nodes had particular receptions

- HTTP packets are routed along S3.
- FTP packets are routed along S6.
- Other TCP packets are routed along S7.
- UDP packets are routed along S4.
- ICMP packets are routed along S5.

SDN is an emerging technology that is likely to revolutionize traditional networking business. By embracing network automation, organizations will save a tremendous amount of time and significantly improve a network's flexibility. Now to prepare for this transition- extensively evaluating which solution meets your business needs best in terms of integration, visibility & reporting, security and reliability is an important first step. Taking enough time to do this thoroughly will be an investment worthwhile to prepare for this inevitable networking transition.

Recent developments in information and Communications Technology domain, for example, mobile, multimedia, cloud, and big data, are demanding for more convenient Internet access, more bandwidth from users, as well as more dynamic management from service providers. SDN is considered as a promising solution to meet these demands. We have presented the concept of SDN and highlighted benefits of SDN in offering enhanced configuration, improved performance, and encouraged innovation. Moreover, we have provided a literature survey of recent SDN researches in the infrastructure layer, the control layer, and the application layer.

Traditional networks are complex and hard to manage. One of the reasons is that the control and data planes are vertically integrated and vendor specific. Another, concurring reason, is that typical networking devices are also tightly tied to line products and versions. In other words, each line of product may have its own particular configuration and management interfaces, implying long cycles for producing product updates (e.g., new firmware) or upgrades (e.g., new versions of the devices). All this has given rise to vendor lock-in problems for network infrastructure owners, as well as posing severe restrictions to change and innovation. SDN created an opportunity for solving these long standing problems. Some of the key ideas of SDN are the introduction of dynamic programmability in forwarding devices through open southbound interfaces, the decoupling of the control and data plane, and the global view of the network by logical centralization of the network brain. While data plane elements became dumb, but highly efficient and programmable packet forwarding devices, the control plane elements are now represented by a single entity, the controller or NOS. Applications implementing the

network logic run on top of the controller and are much easier to develop and deploy when compared to traditional networks. Given the global view, consistency of policies is straight forward to enforce. SDN represents a major paradigm shift in the development and evolution of networks, introducing a new pace of innovation in networking infrastructure. In spite of recent and interesting attempts to survey this new chapter in the history of networks, the literature was still lacking, to the best of our knowledge, a single extensive and comprehensive overview of the building blocks, concepts, and challenges of SDNs. Trying to address this gap, we used a layered approach to methodically dissect the state of the art in terms of concepts, ideas, and components of SDN, covering a broad range of existing solutions, as well as future directions. We started by comparing this new paradigm with traditional networks and discussing how academy and industry helped shape SDN. Following a bottom-up approach, we provided an in-depth overview of what we consider the eight fundamental facets of the SDN problem: 1) hardware..infrastructure;2) southbound interfaces; 3) network virtualization (hypervisor layer between forwarding devices and the NOSs); 4) NOSs (SDN controllers and control platforms); 5) northbound interfaces; 6) virtualization using slicing techniques provided by special purpose libraries and programming languages and compilers; 7) network programming languages; and finally, 8) network applications.

SDN has successfully managed to pave the way toward a next-generation networking, spawning an innovative research and development environment, promoting advances in several areas: switch and controller platform design, evolution of scalability and performance of devices and architectures, promotion of security and dependability. We will continue to witness extensive activity around SDN in the near future. Emerging topics requiring further research are, for example: the migration path to SDN, extending SDN toward carrier transport networks, realization of the network-as-a-service cloud computing paradigm, or SDEs.

CHAPTER – 6

APPLICATIONS OF SDN

6.1 Software Defined Networking Controllers and Applications

A dynamic, open, and logically centralized architecture, Software-Defined Networking (SDN) is ideal for today's bandwidth-hungry applications. With Brocade SDN solutions, you can adjust network traffic flow on the fly to meet changing needs, put profitable services in place rapidly, and reduce operational risk.

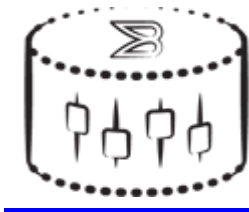


Figure 6.1: Brocade SDN Controller

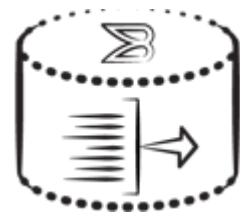


Figure 6.2: Brocade flow optimizer



Figure 6.3: Brocade flow manage

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.

Then I evaluate what the SDN controller can do for us. The SDN controller has end-to-end visibility of a network. With this information, SDN is uniquely positioned to have a positive impact on the network. It can also perform existing applications better. Let us look at some of these benefits:

1. Anywhere access to information via the cloud
2. Network management
3. Resource utilization
4. Security
5. Network Functions Virtualization (NFV)
6. Network service chaining
7. Faster innovation
8. Bandwidth calendaring
9. Network programmability
10. Energy management

- **6.1.1 SDMN**

Software-defined mobile networking (SDMN) is an approach to the design of mobile networks where all protocol-specific features are implemented in software, maximizing the use of generic and commodity hardware and software in both the core network and radio access network. It is proposed as an extension of SDN paradigm to incorporate mobile network specific functionalities.

- **6.1.2 SD-WAN**

An SD-WAN is Wide Area Network (WAN) managed using the principles of software-defined networking. The main driver of SD-WAN is to lower WAN costs using less expensive leased lines, as an alternative or partial replacement of more expensive MPLS lines. Control and management is separated from the hardware, with central controllers allowing easier configuration and administration.

- **6.1.3 SD-LAN**

A SD-LAN is a Local area network (LAN) built around the principles of software-defined networking, though there are key differences in topology, network security, application visibility and control, management and quality of service. SD-LAN decouples control management, and data planes to enable a policy driven architecture for wired and wireless LANs. SD-LANs are characterized by their use of a cloud management system and wireless connectivity without the presence of a physical controller.

- **6.1.4 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, the following paragraphs only focus on the security applications made possible or revisited using SDN.

Several research works 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 defense (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 hid 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. A system to do this called NICE was described in 2012. Introducing an overarching security architecture requires a comprehensive and protracted approach to SDN. Since it was introduced, designers are looking at possible

ways to secure SDN that do not compromise scalability.
One architecture called SN-SECA (SDN+NFV) Security Architecture.

CHAPTER – 7

FUTURE OF SDN

SDN is one of the latest technologies designed for Network Control. SDN can allow a Network Manager to change the way the network devices, i.e. Routers, Switches..etc,handle packets by allowing complete control of the rules set into network devices from a central console. SDN allows for the entire control of as network to allow quick response to changing network or business needs. SDN also has a lot of diagnostic capability.

SDN has a problematic future as it has issues to overcome. The first issue is the console/remote capability with today's security issues many will not want to expose their network to a potential hacker takeover. It is an Open Source Technology.

Another issue is the current state of Network Management policies and practices with single device or single path focus. When a Network Manager looks at SDN he only sees how it can help or hurt his network but SDN is much bigger and a lot of education still remains to be done to make SDN or similar technologies palatable.

SDN is a Human Centric Technology where today's technology is Device Centric which is and always will be a challenge to get managers to adopt especially when one person can completely change your network, storage, WAN..etc fabric.

Many talk about SDN's ability to help with the "Cloud" but before we get SDN involved we need to get the "Cloud" under control. There are also serious financial, training and primary deployment issues.

SDN opens lots of questions for the future of advance networking and computing technologies. We need more responsive technology but not at the cost of security and control. In the long run SDN may be deployed in provider networks but individual corporations may find it just too much to deploy. SDN is in need of a lot more development and proof of being a secure and deployable technology.

SDN is an innovation that will dramatically change this traditional thinking over the next few years, adding intelligence, adaptability, and much more. While not yet close to pervasive, vendors are introducing new SDN products at a rapid rate. The next refresh cycle is right around the corner - SDN will indeed be everywhere.

SDN in fact points to a future that is much more driven by availability and transparency than has historically been the case in networking. The interruptions to service (and, often, the associated extended periods of unreliability) that have traditionally accompanied network upgrades are no longer any more necessary than they are acceptable. And routine changes to policies – even, in some cases, automatically put in place as anticipated conditions arise – are now beginning to substitute for what was once viewed as unavoidable congestion, outages, and forced upgrades with numerous associated risks. So, then, the future of SDN is nothing short of..bright. In fact, we expect SDN to become broadly pervasive over the next five years, as standards progress, the benefits become quantified, and end-user organizations proceed down the experience curve.

One of the very interesting possibilities now appearing is the merging of SDN with that other leading-edge trend in networking, network function virtualization (NFV). NFV, like all virtualization technologies, substitutes software-based constructs for what would otherwise require dedicated hardware.

A virtual machine, for example, is really just a software implementation of a real processor, using specialized hardware to speed certain elements. Moving network functionality into virtual machines and even the Cloud, which may become the most popular form of NFV, is already at work in carrier networks, substituting code running on otherwise general-purpose processors for dedicated hardware implementations of functionality defined in standards.

Virtualization in this case, sometimes called networking as a service (NaaS), enables a greater degree of reliability and fault-tolerance, scalability - even on demand, as customer-driven traffic volumes grow - and the ability to grow solutions organically without disruption.

Open source will also play a role in the eventual dominance of SDN. The OpenStack infrastructure-as-a-service (IaaS) offering has already achieved a degree of popularity,

and includes a good degree of networking functionality. The OpenDaylight Project is specifically designed to implement an open-source platform for both SDN and NFV. Other open-source implementations are well underway, but we must admit that some proprietary solutions will also see success.

SDN may also help us address one of the truly insoluble problems in networking and IT overall – security. Because SDN implements policy, a high degree of traffic-flow monitoring and management is essential. SDN implementations can thus be programmed to look for anomalies that might represent a denial-of-service attack, an attempted break-in, or unusual activity that might indicate data being compromised.

Networking has evolved over the past three decades driven by extremely high (and increasingly mission-critical) demand, rapidly evolving hardware, software and protocol technologies, and a hyper-competitive marketplace.

The result, perhaps inevitably, has been complexity, albeit with a high degree of interoperability in basic services. SDN promises to re-shape networking into a more unified, “softer” facility that simplifies network services and operations, improves scalability, and simplifies operations (essential as demand in terms of number of users and devices as well as traffic volume and time-boundedness continues to grow), improves interoperability, and all within the framework of a contemporary, policy-driven strategy.

REFERENCES

1. Harishkumar Bhokare, Mansi Bhonsle
Application Aware Routing in SDN
International Journal of Science and Research (IJSR)
2. Prof. Chao-Li Tarng, Anudeep Reddy Ramashayam, Sneha Viswalingam
SDN Based Monitoring and Filtering Application
3. Hyunwoo Nam, Kyung-Hwa Kim and Henning Schulzrinne, The Internet Real-Time Lab., Columbia University
Application-aware Routing Platform using SDN
4. Application-aware Routing Platform using SDN
https://issuu.com/aricenttech/docs/aricent_whitepaper___application_aw
5. https://en.wikipedia.org/wiki/Software-defined_networking#Architectural_components
6. <http://globalconfig.net/software-defined-networking-vs-traditional/>
7. <https://www.sdxcentral.com/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/>
8. <https://www.sdxcentral.com/sdn/definitions/software-defined-sdn-wan/>

9
28/4/17

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
LEARNING RESOURCE CENTER

PLAGIARISM VERIFICATION REPORT

Date: 28/04/17

Type of Document (Tick): Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: UTKARSH, ADITYA, RISHABH Department: ECE

Enrolment No. 131092, 131087, 131054 Registration No. _____

Phone No. 9816927591 Email ID. utkarshkhurana@gmail.com

Name of the Supervisor: Dr. Rajiv Kumar

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

APPLICATION AWARE ROUTING IN SDN

Kindly allow me to avail Turnitin software report for the document mentioned above.

Aditya
Aditya
(Signature)

FOR ACCOUNTS DEPARTMENT:

Amount deposited: Rs. 900 Dated: 28/4/17 Receipt No. CRV1700/353
(Enclosed payment slip)

(Account Officer)

FOR LRC USE:

The above document was scanned for plagiarism check. The outcome of the same is reported below:

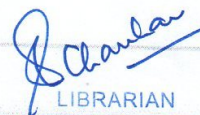
Copy Received on	Report delivered on	Similarity Index in %	Submission Details	
			Word Counts	Character Counts
28-04-17	01-05-17	27%	Page counts	47
			File Size	761.44K
			Word Counts	6074
			Character Counts	49789

Schankar
Checked by
Name & Signature

Schankar
LIBRARIAN
LEARNING RESOURCE CENTER
Jaypee University of Information Technology
Waknaghat, Dist, Solan (Himachal Pradesh)
Pin Code: 173234

Submission Info

SUBMISSION ID	807579326
SUBMISSION DATE	01-May-2017 10:11
SUBMISSION COUNT	1
FILE NAME	Project_Report_Utkarsh...
FILE SIZE	761.44K
CHARACTER COUNT	49789
WORD COUNT	6074
PAGE COUNT	47
ORIGINALITY	
OVERALL	27%
INTERNET	22%
PUBLICATIONS	15%
STUDENT PAPERS	19%
GRADEMARK	
LAST GRADED	N/A
COMMENTS	0
QUICKMARKS	



LIBRARIAN
LEARNING RESOURCE CENTER
Jaypee University of Information Technology
Waknaghat, Distt, Solan (Himachal Pradesh)
Pin Code: 173234