SDN as a SAAS in Cloud Computing

Deepak Surana
12BCE018

DEPARTMENT OF COMPUTER ENGINEERING
AHMEDABAD -382424

November 2014

SDN as a SAAS in Cloud Computing

Seminar

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Science and Engineering

Deepak Surana
12BCE018

DEPARTMENT OF COMPUTER ENGINEERING
AHMEDABAD -382424

November 2014

CERTIFICATE

This is to certify that the seminar entitled SDN as a SAAS in Cloud Computing submitted by Deepak Surana (12BCE018), towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in B.Tech of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this Seminar, to the best of my knowledge, havent been submitted to any other university or institution for award of any degree or diploma.

Prof. Jitendra Bhatia
Seminar Guide

Prof. Anitha Ashishdeep
Seminar Coordinator

Acknowledgements

I am profoundly grateful to **Asst. Prof. Jitendra Bhatia** for his expert guidance and continuous encouragement throughout to see that this seminar work achieves its target since its commencement to its completion.

I would like to express deepest appreciation towards **Prof. Anitha Ashishdeep**, whose invaluable guidance supported us in completing this seminar work.

DEEPAK SURANA
12BCE018

ABSTRACT

*E*merging mega-trends (e.g., mobile, social, cloud, and big data) in Information and Communication Technologies (ICT) are commanding new challenges to future Internet, for which ubiquitous accessibility, high bandwidth, and dynamic management are crucial. However, traditional approaches based on manual configuration of proprietary devices are cumbersome, error-prone, and cannot fully utilize the capability of physical network infrastructure. Recently, Software-Defined Networking (SDN) has been touted as one of the most promising solutions for future Internet. SDN is characterized by its two distinguished features, including decoupling the control plane from the data plane and providing the network operators and data centres to flexibly manage their networking equipment using software running on external servers. As a result, SDN is positioned to provide more efficient configuration, better performance, and higher flexibility to accommodate innovative network designs.On the other hand cloud computing materializes the vision of utility computing. Tenants can benefit from on-demand provisioning of networking, storage and compute resources according to a pay-per-use business model. Network virtualization is the key to the current and future success of cloud computing.

Contents

Chapter 1

Introduction

Networks of the twenty first century offer immense flexibility to the business and individual users, but at the cost of higher complexity. Controlling and managing such networks have become highly complex and specialized activities. In this context Software defined networking (SDN) is being looked upon as a promising paradigm that has the power of changing the networking world. Through SDN, network administrators would be able to get a better control over the traffic flows. They would also be able to easily program and modify network policies to manage these flows according to the user requirements. This becomes possible because SDN transfers control and policy functions from a large number of distributed devices to one or more general-purpose servers. There has been heightened interest in recent years in the academia, industry and the network operators in research and implementation of SDN. It now appears that the time for SDN has finally arrived.

The key idea of SDN is to decouple the control plane from the data plane and allow flexible and efficient management and operation of the network via software programs. Specifically, devices (e.g., switches and routers) in the data plane perform packet forwarding, based on rules installed by controllers. Controllers in the control plane oversee the underlying network and provide a flexible and efficient platform to implement various network applications and services. Under this new paradigm, innovative solutions for specific purposes (e.g., network security, network virtualization and green networking) can be rapidly implemented in form of software and deployed in networks with real traffic. Moreover, SDN allows logical centralization of feedback control with better decisions based on a global network view and cross-layer information.

Cloud computing has emerged as a widely accepted computing paradigm built around core concepts such as elimination of up-front investment, reduction of operational expenses, on-demand computing resources, elastic scaling, and establishing a pay-per-usage business model for information technology and computing ser-

vices.There are different models of cloud computing that are offered today as services like Software as a Service (SaaS), Platform as a Service (PaaS), Network as a Service (NaaS) and Infrastructure as a Service (IaaS) [1]. In spite of all recent research and developments, cloud-computing technology is still evolving. Several remaining gaps and concerns are being addressed by alliances, industry, and standards bodies.

SDN is an appealing platform for network virtualization since each tenants control logic can run on a controller rather than on physical switches. In particular OpenFlow offers a standard interface for caching packet forwarding rules in the flow table of switches, querying traffic statistics, and notifications for topology changes. Moreover, the emergence of the software-defined networking (SDN) paradigm provides a new opportunity to closely integrate application provisioning in the cloud with the network through programmable interfaces and automation.

Chapter 2

SDN

The term software-defined networking (SDN) has been coined in recent years. However, the concept behind SDN has been evolving since 1996, driven by the desire to provide user-controlled management of forwarding in network nodes.

In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications.

2.1 Definition and description of SDN

Software-Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable.

SDN focuses on four key features:

1) Separation of the control plane from the data plane.

2) A centralized controller and view of the network.

3) Open interfaces between the devices in the control plane (controllers) and those in the data plane.

4) Programmability of the network by external applications.

In the context of SDN, its uniqueness resides on the fact that it provides programmability through decoupling of control and data planes. Specifically, SDN offers simple programmable network devices rather than making networking devices more complex as in the case of active networking. Moreover, SDN proposes separation of control and data planes in the network architectural design. With this design, network control can be done separately on the control plane without affecting data flows. As such, network intelligence can be taken out of switching devices and placed on controllers. At the same time, switching devices can now be externally controlled by software without onboard intelligence. The decoupling of control plane from data plane offers not only a simpler programmable environment but also

a greater freedom for external software to define the behavior of a network.

2.2 Why SDN ?

In traditional networks control and data planes are tightly integrated in the network devices like switches and routers. Once the forwarding policy has been defined, the only way to change it is by changing the configuration of all the affected devices. This is time consuming and puts a limit on scalability and also meeting challenges of mobility and big data. The actual networks have by now become complicated and expensive to maintain. At the same time, revenues are globally declining. Operators are, therefore, looking for solutions that can unify network management and provisioning across multiple domains. SDN has been developed to take care of what is missing in the traditional networks. This has been done by moving control out of the network nodes and keeping it centralized in a server called controller, which has the complete view of the network.This controller can then use the complete knowledge of the network to optimize flow management and support service-user requirements of bandwidth, scalability and flexibility. The separation of the forwarding hardware from the control logic allows easier deployment of new protocols and applications, simplifies network visualization and management. Making applications aware of the network enables greatly improved use of resources and opens up the potential for new applications with the associated potential for revenue generation.

1.NETWORKING THE OLD WAY :-

In traditional networks, the control and data planes are combined in a network node. The control plane is responsible for configuration of the node and programming the paths to be used for data flows. Once these paths have been determined, they are pushed down to the data plane. Data forwarding at the hardware level is based on this control information.

In this traditional approach, once the flow management (forwarding policy) has been defined, the only way to make an adjustment to the policy is via changes to the configuration of the devices. This has proven restrictive for network operators who are keen to scale their networks in response to changing traffic demands.

2.NETWORKING THE SDN WAY :-

From the service-focused requirements, SDN has emerged. Control is moved out of the individual network nodes and into the separate, centralized controller. SDN switches are controlled by a network operating system (NOS) that collects information using the API and manipulates their forwarding plane, providing an abstract

model of the network topology to the SDN controller hosting the applications.

The controller can therefore exploit complete knowledge of the network to optimize flow management and support service-user requirements of scalability and flexibility. For example, bandwidth can be dynamically allocated into the data plane from the application.

2.3 SDN Benefits

SDN, with its inherent decoupling of control plane from data plane, offers a greater control of a network through programming. This combined feature would bring potential benefits of enhanced configuration, improved performance, and encouraged innovation in network architecture and operations.

1) Enhancing Configuration: In network management, configuration is one of the most important functions. Specifically, when new equipment is added into an existing network, proper configurations are required to achieve coherent network operation as a whole. It is generally accepted that, with the current network design, automatic and dynamic reconfiguration of a network remains a big challenge.

SDN will help to remedy such a situation in network management. In SDN, unification of the control plane over all kinds of network devices, including switches, routers, Network Address Translators (NATs), firewalls, and load balancers, renders it possible to configure network devices from a single point, automatically via software controlling.

2) Improving Performance: In network operations, one of the key objectives is to maximize utilization of the invested network infrastructure. However, owing to coexistence of various technologies and stakeholders in a single network, optimizing performance of the network as a whole has been considered difficult.

Current approaches often focus on optimizing performance of a subset of networks or the quality of user experience for some network services. Obviously, these approaches, based on local information without cross-layer consideration, could lead to suboptimal performance, if not conflicting network operations. SDN allows for a centralized control with a global network view and a feedback control with information exchanged between different layers in the network architecture. As such, many challenging performance optimization problems would become manageable with properly designed centralized algorithms.

3) Encouraging Innovation: In the presence of continuing evolution of network applications, future network should encourage innovation rather than attempt

to precisely predict and perfectly meet requirements of future applications. Unfortunately, any new idea or design immediately faces challenges in implementation, experimentation, and deployment into existing networks. The main hurdle arises from widely used proprietary hardware in conventional network components, preventing modification for experimentation. Besides, even when experimentations are possible, they are often conducted in a separate simplified testbed. These experimentations do not give sufficient confidence for industrial adaptation of these new ideas or network designs.

In comparison, SDN encourages innovation by providing a programmable network platform to implement, experiment, and deploy new ideas, new applications, and new revenue earning services conveniently and flexibly. High configurability of SDN offers clear separation among virtual networks permitting experimentation on a real environment. Progressive deployment of new ideas can be performed through a seamless transition from an experimental phase to an operational phase.

Chapter 3

Challenges to SDN

SDN holds great promise in terms of simplifying network deployment and operation along with lowering the total cost of managing enterprise and carrier networks by providing programmable network services. However, a number of challenges remain to be addressed. This section focuses on four specific questions arising from the challenges of SDN.

3.1 PERFORMANCE VS. FLEXIBILITY

One fundamental challenge of SDN is how to handle high-touch high-security high-performance packet processing flows in an efficient manner. There are two elements to consider: **performance and programmability/flexibility**.

Performance refers specifically to the processing speed of the network node considering both throughput and latency. Programmability means the capability to change and/or accept a new set of instructions in order to alter functional behavior. Flexibility is the ability to adapt systems to support new unforeseen features (e.g., applications, protocols, security measures).

It has to be ensured that the performance of SDN in terms of throughput and latency is commensurate with the traditional networks if not better. These networks should be flexible enough so that new features and capabilities can be programmed. Introduction of new protocols, applications and security features are examples of the changes that might be called for. Use of general-purpose processors, in the controllers, provides high flexibility. High-level programminglanguages and design tools enable the highest design abstraction and rapid development of complex packet processing functions. The limitation of centralized general-purpose processor implementation, however, is its performance and power dissipation. Looking at the tradeoff it is clear that only a hybrid approach will provide an effective technology solution for SDN. Use of technologies aimed at improving power dissipation, cost

and scalability would give the best combination of performance and programmability.With a programmable interface built on standard hardware, a multivendor equipped network becomes a possibility.

3.2 SCALABILITY

Assuming that the performance requirements can be achieved within the hybrid programmable architecture, a further issue that has seen some discussion but limited solution is scalability in SDN. The focus here is on controller scalability in which three specific challenges are identified. The first is the latency introduced by exchanging network information between multiple nodes and a single controller. The second is how SDN controllers communicate with other controllers using the east and westbound APIs. The third challenge is the size and operation of the controller back-end database.
Considering the first issue, a distributed or peer-to-peer controller infrastructure would share the communication burden of the controller. However, this approach does not eliminate the second challenge of controller-to-controller interactions, for which an overall network view is required.

Traditional packet networks lend themselves to scalable solutions because they do not require extensive state to be held between system units. Each network node is autonomous, requiring only limited knowledge of its neighbors. Routing protocols have been designed to control traffic with this in mind. In order to create resilient networks, alternative paths and secondary equipment are required. It may then be necessary to hold some state between systems to ensure that should a failure occur, there is little or no interruption in service.

Within a pure SDN environment, a single controller or group of controllers would provide control plane services for a wider number of data forwarding nodes, thus allowing a systemwide view of network resources. A specific solution to controller scalability is HyperFlow,that sits on NOX and allows the network operators to have as many controllers as required. All the controllers share the same consistent network-wide view. To achieve full scalability in SDN, it is being studied whether it would help if the CPU within the network node handles some work locally. A hybrid architecture can reduce the communication between the nodes and the controller and consequently the load on the controller. The backend database size of the controller and its storage requirements are simultaneously reduced.

3.3 SECURITY

Security is one area on which there has been limited industry and research community discussion. A greater focus on security is, therefore, required if SDN is going to be accepted in broader deployment. At the controller-application level, questions have been raised around authentication and authorization mechanisms, in a multi-tenant setting, that would allow protection of resources of multiple organizations accessing the network. A security model must be evolved to take care of varying privilege requirements of applications. The controllers are a particularly attractive target for attack in the SDN architecture, open to unauthorized access and exploitation. If the controller is not secured then an attacker may steal its identity, masquerade as one and carry out malicious activities. A security technology such as transport layer security (TLS) with mutual authentication between the controllers and their switches can mitigate these threats. This security feature is optional in Openflow and the standard of TLS is not specified. A full security specification for the controller-switch interface must be defined to secure the connection and protect data transmitted across it. When there are multiple controllers in the network, potential for unauthorized access to nodes and alteration of its configuration and traffic rerouting may take place. It could lead to Denial of Service (DoS), which could have crippling effect on the network. SDN's strength in open interfaces and known protocols become a boon for attackers. However, the SDN architecture supports a highly reactive security monitoring, analysis, and response system.

3.4 INTEROPERABILITY

It would be straightforward to deploy a completely new infrastructure based on SDN technology. For this, all elements and devices in the network would be SDN-enabled. However, there is a vast installed base of networks supporting vital systems and businesses today. To simply swap out these networks for new infrastructure is not going to be possible, and is only well suited for closed environments such as data centers and campus networks.

The transition to SDN therefore requires coexistence of SDN and the legacy equipment. More developmental work is required to achieve a hybrid SDN infrastructure in which SDN enabled and traditional integrated nodes inter-operate. Such interoperability requires the support of an appropriate protocol that both introduces the requirements for SDN communication interfaces and provides backward compatibility with existing IP routing and multiprotocol label switching (MPLS) control

plane technologies. Such a solution would reduce the cost, risk, and disruption for enterprise and carrier networks transitioning to SDN. Different industry groups like IETF, ETSI and ONF are developing standards for SDN. Their work needs to be harmonized for developing the most effective standards to support migration from the traditional network model to SDN.

Chapter 4

Reference Model

The model consists of three layers, namely an infrastructure layer, a control layer, and an application layer, stacking over each other.

The infrastructure layer consists of switching devices (e.g., switches, routers, etc.) in the data plane. Functions of these switching devices are mostly two-fold. First, they are responsible for collecting network status, storing them temporally in local devices and sending them to controllers. The network status may include information such as network topology, traffic statistics, and network usages. Second, they are responsible for processing packets based on rules provided by a controller.

The control layer bridges the application layer and the infrastructure layer, via its two interfaces. For downward interacting with the infrastructure layer (i.e., the south-bound interface), it specifies functions for controllers to access functions provided by switching devices. The functions may include reporting network status and importing packet forwarding rules. For upward interacting with the application layer (i.e., the north-bound interface), it provides service access points in various forms, for example, an Application Programming Interface (API). SDN applications can access network status information reported from switching devices through this API, make system tuning decisions based on this information, and carry out these decisions by setting packet forwarding rules to switching devices using this API. Since multiple controllers will exist for a large administrative network domain, an eastwest communication interface among the controllers will also be needed for the controllers to share network information and coordinate their decision-making processes.

The application layer contains SDN applications designed to fulfill user requirements. Through the programmable platform provided by the control layer, SDN applications are able to access and control switching devices at the infrastructure layer. Example of SDN applications could include dynamic access control, seamless mobility and migration, server load balancing, and network virtualization.

4.1 INFRASTRUCTURE LAYER

At the lowest layer in the SDN reference model, the infrastructure layer consists of switching devices (e.g., switches, routers, etc.), which are interconnected to formulate a single network.The main research concerns associated with the infrastructure layer include both efficient operations of switching devices and utilization of transmission media.

A. Switching Devices: The architectural design of an SDN switching device, consisting of two logical components for the data plane and the control plane.

In the data plane, the switching device, in particular, through its processor, performs packet forwarding, based on the forwarding rules imposed by the control layer.

In the control plane, the switching device communicates with controllers at the control layer to receive rules, including packet forwarding rules at a switching level and link tuning rules at a data-link level, and stores the rules in its local memory.

The switching devices are simply responsible for gathering and reporting network status as well as processing packets based on imposed forwarding rules. It follows that the SDN switching devices are simpler and will be easier to manufacture.

1) Control Plane: In the control plane of SDN switching devices, one of the main design challenges resides on the efficient use of onboard memory.
In case of insufficient memory space, packets would be dropped or directed to controllers for further decisions on how to process them, resulting in a degraded network performance Another major principle in improving SDN switching device design is judicious combination of different storage technologies to achieve desired memory size, processing speed, and flexibility with reasonable price and complexity.

2) Data Plane: The main function of an SDN switching devices data plane is packet forwarding. Specifically, upon receiving of a packet, the switching device first identifies the forwarding rule that matches with the packet and then forwards the packet to next hop accordingly.

Using a long vector for forwarding decision would undoubtedly increases processing complexity in computation, resulting a fundamental trade-off between cost and efficiency in SDN packet processing. First, in PC-based switching devices, using software for packet processing may result in inefficient performance, using hardware classification to increase processing throughput incoming packets are directed to an onboard Network Interface Controller (NIC) for hardware classification based on flow signatures. As a result, a CPU is exempted from the lookup process.

B. Transmission Media 1) Wireless Radio: Software-Defined Radio (SDR) permits control of wireless transmission strategy via software. Many computationally intensive processing blocks are common at the physical layer of all modern wireless systems, differing only in configurations. One instance is that almost all wireless systems use Fast Fourier Transform (FFT) with probably different FFT-lengths. This observation motivates them to propose OpenRadio to decouple wireless protocol definition from the hardware and use a declarative interface to program wireless protocols. In fact, SDR can benefit from the central control and global view of SDN, as used in Dyson. In return, SDN controllers can control SDR systems and have a widespread and precise control of all the network devices.

2) Optical Fibers: Optical fibers offer a high capacity with low power consumption. They are widely used in backbones for aggregated traffic. The idea of software reconfiguration used in wireless networks can also be adopted in optical networks by employing Reconfigurable Optical Add/Drop Multiplexers (ROADMs). Integrating these technologies into the SDN control plane helps achieve more precise and efficient control of the data plane suggest extending parameters used in forwarding rule matching from layer 2, 3, and 4 headers of a packet to include layer 1 switching technologies, such as time-slot, wavelength, and fiber switching. Thus it provides a single unified control plane over packet and optical networks. The proposed scheme provides a simplified control model, but needs to upgrade optical circuit switching devices to support this extension, propose to use a virtual switch on each optical switching node to obtain unified control plane. Each physical interface of an optical switching node is mapped to a virtual interface correspondingly. Messages between the controller and the virtual switch are converted to commands acceptable by the optical switching devices.

4.2 CONTROL LAYER

A logical design for SDN control layer, consists of four main components, namely a high level language, a rule update process, a network status collection process and a network status synchronization process.

A. Controller Design:

Objects: The SDN controller deals with two types of objects. One is used for network controlling, including policies imposed by the application layer and packet forwarding rules for the infrastructure. The other is related to network monitoring, in the format of local and global network status.

In the downward flow, the controller translates the application policy into packet forwarding rules, in respect to network status. The main concern of this process is to ensure validity and consistency of the forwarding rules.

In the upward flow, the controller synchronizes network status collected from the infrastructure for networking decision making.

Interfaces: The SDN controller has two interfaces. The south-bound interface, which is marked as the controller infrastructure interface deals with transactions with the infrastructure layer, i.e., collecting network status and updates packet forwarding rules to switching devices at the infrastructure layer accordingly. The north-bound interface, which is marked as the application-controller interface handles transactions with the application layer, i.e., receiving policies described in high level languages from SDN applications and providing a synchronized global view.

1) High Level Language - One of the key controller functions is to translate application requirements into packet forwarding rules. This function dictates a communication protocol (e.g., a programming language) between the application layer and the control layer. One straight forward approach is to adopt some common configuration languages. However, these common configuration languages only offer primitive abstractions derived from capabilities of underlying hardware. Given that they are designed for hardware configurations, they are typically inadequate to accommodate dynamic and stateful network status. Moreover, they are error-prone and demanding extra effort in the process of programming.

Flow-based Management Language (FML) , a well-developed programming language for SDN called Frenetic, and another high level language called Nettle.

2) Rules Update - An SDN controller is also responsible for generating packet forwarding rules describing the policies and installing them into appropriate switching devices for operation. At the same time, forwarding rules in switching devices need to be updated because of configuration changes and dynamic control, such as directing traffic from one replica to another for dynamical load balancing, Virtual Machine (VM) migration, and network recovery after unexpected failure. Strict Consistency: it ensures that either the original rule set or the updated rule set is used. Strict consistency could be enforced in a per-packet level, where each packet is processed, or in a per-flow level, where all packets of a flow are processed by either the original rule set or the updated rule set. Eventual Consistency: it ensures that the later packets use the updated rule set eventually after the update procedure finishes and allows the earlier packets of the same flow to use the original rule set before or during the update procedure.

3) Network Status Collection - In the upward flow, controllers collect network status to build a global view of an entire network and provide the application layer with necessary information, for example, network topology graph, for network operation decisions. One main network status is traffic statistics, such as, duration time, packet number, data size, and bandwidth share of a flow.

Each switching device collects and stores local traffic statistics within its own storage. These local traffic statistics may be retrieved by controllers (i.e., a pull mode), or proactively reported to controllers (i.e., a push mode).

4) Network Status Synchronization - Delegating control to a centralized controller can cause performance bottleneck at the centralized controller. A common solution to overcome this bottleneck is deploying multiple controllers acting peer, backup, or replicate controllers. Maintaining a consistent global view among all controllers is essential to ensure proper network operations. Inconsistent or stale states may result in the application layer making incorrect decisions, which then leads to inappropriate or suboptimal operations of the network.

HyperFlow that allows sharing of a synchronized consistent network-wide view among multiple controllers.

B. Policy and Rule Validation :

Consistency in policies and rules stands out as an important design issue to stabilize the routing choice in SDN networks. This is due to the fact, in SDN networks, multiple applications could connect the same controller, and multiple controllers could be figured for performance improvement. As a result, conflicting configurations might surface, demanding an internal coordination among different participating units. Specifically, policies and rules should be validated to identify potential conflicts.

On one hand, the rules can be checked statically for certain network invariants, such as reachability, loop-free, and consistency, based on network topology. On the other hand, it is also useful to check rules in real-time, as network state evolves. However, achieving extremely low latency during these checks is ultimately important.

C. Control Layer Performance :

The performance of SDN networks highly depends on the control layer, which, in turn, is constrained by the scalability of centralized controllers. Indeed, all the transactions in the control plane are involved with controllers. Switching devices need to request controllers for packet forwarding rules reactively when the first packet of each flow arrives. Rule update and network status collection also involve

in frequent communication between controllers and switching devices. In this aspect, bandwidth consumption and latency of frequent communication affect control layer scalability significantly.

1) Increasing Processing Ability - It exploits parallelism together with additional throughput optimization techniques, such as, input and output batching, core and thread binding.

2) Reducing Request Frequency - One strategy is to modify switching devices so as to handle requests in the data plane or near the data plane.Following the strategy of handling requests in the data plane, distributing rules across authority switches. Packets are diverted through authority switches as needed to access appropriate rules, thus all packets can be handled in the data plane without requesting to controllers.

DevoFlow to handle most mice flows in switching Devices. Kandoo has a two-layer architecture to handle most of frequent events locally. The bottom layer is a group of controllers without a network-wide view that handle most of frequent events. Elephant flow detection, which needs to constantly query each switching device to see whether a flow has enough data to be an elephant flow, can be done at the bottom layer. At the same time, the top layer is a logically centralized controller that maintains a network wide view and handles rare events, for example, requesting for routing decisions.As a result of the two-layer architecture, heavy communication burden is offloaded to highly replicable local controllers at the bottom layer.

3) Performance Benchmarking - Controller performance benchmarking can be used to identify performance bottlenecks and is essential to increase processing ability of a controller. Cbench tests controller performance by generating requests for packet forwarding rules and watching for responses from the controller.OFCBenchmark provides statistics of response rate, response time, and number of unanswered packets for each switching device.

4.3 APPLICATION LAYER

SDN applications can programmatically implement strategies to manipulate the underlying physical networks using a high level language provided by the control layer. In this aspect, SDN offers Platform as a Service model for networking.

A. Adaptive Routing :

Packet switching and routing are the main functions of a network. Traditionally, switching and routing designs are based on distributed approaches for robust-

ness which have limited ability to achieve adaptive control [181]. As an alternative solution, SDN offers closed loop control, feeding applications with timely global network status information and permitting applications to adaptively control a network.

1) Load Balancing - A common practice of load balancing in data centers is deploying front-end load balancers to direct each clients request to a particular server replica to increase throughput, reduce response time, and avoid overloading of network.

2) Cross-Layer Design - A cross-layer approach is a highly touted technique to enhance integration of entities at different layers in a layered architecture, for example, the OSI reference model, by allowing entities at different layers to exchange information among each other. As SDN offers a platform for applications to easily access network status information, cross-layer approaches can be easily developed on this platform. Utilizing QoS information for appropriate network resource reservation represents one effective cross-layer approach to achieve guaranteed QoS.

B. Boundless Roaming : Smartphones and tablets are becoming dominating devicesin the Internet access. These mobile devices access the Internet wirelessly. To ensure continuous connectivity while these devices move from one location to another, connections may be handed over from one base station to another, or even from one wireless network to another. In SDN, networks of different carriers with different technologies could have a common unified control plane. This enables boundless mobility with seamless wireless connection handover between different technologies and carriers.

C. Network Maintenance : Centralized and automated managementand consistent policy enforcement, inherent in SDN networks,help reduce configuration errors.With a global view and central control of configuration, SDN offers opportunities to design comprehensive network diagnosis and prognosis mechanisms for automated network maintenance.

A key benefit of SDN-based prognosis mechanisms is that central control of an SDN implementation can directly resolve network failures with shorter routing convergence time

D. Network Security : Due to the heterogeneity in network applications, ensuring exclusive accesses by legitimate network applications involves implementation of a network-wide policy and tedious configuration of firewalls, proxy servers, and other devices. In this aspect, SDN offers a convenient platform to centralize, merge and check policies and configurations to make sure that the implementation

meets required protection thus preventing security breaches proactively. Ability to collect network status of SDN allows analysis of traffic patterns for potential security threats. Attacks, such as low-rate burst attacks and Distributed Denial-of- Service (DDoS) attacks, can be detected just by analyzing traffic pattern. At the same time, SDN provides programmatic control over traffic flows. Consequently, traffic of interest can be explicitly directed to Intrusion Prevention Systems (IPSs) for Deep Packet Inspection (DPI). If attacks are detected, SDN can install packet forwarding rules to switching devices to block the attack traffic from entering and propagating in a network.

Controllers will specify translation between the virtual IP and real IP while maintaining configuration integrity.

E. Network Virtualization : Network virtualization is a popular technique to allow multiple heterogeneous network architectures to cohabit on a shared infrastructure and plays a significant role in the IaaS model.

FlowVisor is located between guest controllers and switching devices acting as a transparent proxy to filter control messages such that a guest controller can only see and manipulate its own virtual network.

F. Green Networking : It turns out that SDN switching devices may not directly offer benefits in energy reduction in network operation. However, SDN could offer significant promises in supporting minimization of network-wide energy consumption.

Chapter 5

SDN for Cloud Computing

Cloud computing is changing the way people do computing and business. It provisions computing and storage resources on demand and charges on usage with server and network virtualization. SDN provides opportunities to extend the service provisioning model of IaaS beyond computing and storage resources to include a rich set of accompanying network services for more flexible and efficient cloud computing. Cloud computing has emerged as a widely accepted computing paradigm built around concepts such as elimination of up-front investment, reduction of operational expenses, on-demand computing resources, elastic scaling and establishing a pay-per-usage business model for information technology and computing services. Applications may need to be rewritten or reconfigured before deployment in the cloud to address several network related limitations. It is said that networks create cloud and therefore managing the interplay between networks and clouds is key to efficiency of clouds and in turn their success. SDN is increasingly accepted as the path to design of cloud inter-networks for use of cloud computing on a massive scale.

5.1 Cloud inter-network and SDN

Two models of SDN have emerged: the ”overlay model” and the ”network model.” In the overlay model, software creates a virtual network. In the network model, network devices create those virtual networks. Overlay SDNs, such as VMware’s recently acquired Nicira technology, use software to partition IP or Ethernet addresses into multiple virtual subnetworks. Network APIs allow applications to access these subnetworks as though they were IP or Ethernet networks. The software keeps the traffic of multiple subnetworks secure and isolated. Network-hosted SDNs are built from network devices; therefore, they manage SDN traffic directly. Both the overlay and network models and the SDN clash in the WAN. Overlay SDN depends on software to create virtual networks and getting all the users to have the required

software is difficult. The network devices have to be software based so that they can be easily updated and therefore cannot use overlay virtualization. Therefore, the network-hosted SDN becomes kind of unavoidable if the cloud virtual networks are to be extended to the user. Additionally, adequate QoS cannot be assured with an overlay SDN. A network-hosted SDN can manage traffic and ensure QoS.

5.2 Integration of SDN and cloud computing

The demand for virtualization and cloud services has been growing rapidly and attracting considerable interest from industry and academia. The challenges it presents include rapid provisioning, efficient resource management, and scalability that can be addressed using SDNs control model. A high level description of key building blocks for an SDN-based cloud federation includes: 1) an OpenFlow enabled cloud backbone edge nodes, which connect to the enterprise and cloud provider data center, 2) an OpenFlow enabled core nodes which efficiently switch traffic between these edge nodes, 3) an OpenFlow/SDN-based controller to configure the flow forwarding tables in the cloud backbone nodes and providing a WAN network virtualization application 4) a hybrid cloud operation and orchestration software to manage the enterprise and provider data canter federation, inter-cloud workflow, and resource management of compute/storage and inter-data center network management.

In clouds, offering Infrastructure as a Service (IaaS), users only get a logical view of the underlying network and have limited control. SDN technologies have the capabilities to facilitate delegation of network controls and provide some level of network abstractions to end-users to enable them to configure their slice of the network. Delegating more control to the end-users could raise security concerns for the providers. SDN-based clouds will allow enterprises to have multi-vendor networks to avoid vendor lock-in. They can access dynamic bandwidth for ad-hoc, timely inter-data center workload migration and processing; and eliminate the burden of underutilized, costly high-capacity fixed private leased lines. SDN-enabled bandwidth-on-demand services provide automated and intelligent service provisioning, driven by cloud service orchestration logic and customer requirements.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

SDN is a framework that increases the flexibility of the network through separation of control and data planes. This separation makes the network switching and routing devices simpler and less expensive. The control plane could be implemented in a general purpose commodity server. This server as a centralized controller takes care of routing and other policies according to which the network devices function.

Some of the challenges in the existing Cloud Networks are: guaranteed performance of applications when applications are moved from on-premises to the cloud facility, flexible deployment of appliances (e.g., intrusion detection systems, or firewalls), and associated complexities to the policy enforcement and topology dependence. SDN provides a new, dynamic network architecture that transforms traditional network backbones into rich service-delivery platforms. By decoupling the network control and data planes, SDN-based architecture abstracts the underlying infrastructure from the applications that utilize it. This makes the networking infrastructure programmable and manageable at scale. SDN adoption can improve network manageability, scalability and dynamism in enterprise data centre.

6.2 Future Scope

Many challenges in SDN still need further research attention and many organizations have started research projects in various aspects of SDN.The list of open issues covering the whole lifecycle of SDN from standardization, implementation, to deployment:

Standardization of SDN: Being the de facto implementation of the SDN concept, OpenFlow is by no means the only SDN implementation.Besides, OpenFlow specification evolves rapidly and allows different interpretations. Therefore, differ-

ent implementations may behave inconsistently and cause unpredictable disorders.

Implementation of SDN: The current SDN approach of decoupling control plane completely from data plane suggests a total removal of any onboard routing protocols from switching devices. This approach may be too idealistic, which may prevent SDN from widespread adaptation. An immediate and direct transform to idealistic SDN will need a lot risky investment to replace all the conventional network devices. In the transition phase from conventional switching devices to fully decoupled SDN switching devices, semi-decoupled switching devices, which run routing protocols and can also be remotely controlled are helpful to smooth the evolution to idealistic SDN.

Deployment of SDN: Further study of SDN in carrier networks, wireless mesh networks with fast client mobility, and wireless sensor networks which require high reliability and reachability is also needed for wide deployment of SDN.

References

[1] *A Survey on Software-Defined Networking* ; Wenfeng Xia, Yonggang Wen, Senior Member, IEEE, Chuan Heng Foh, Senior Member, IEEE, Dusit Niyato, Member, IEEE, and Haiyong Xie, Member, IEEE

[2] *SDN-Based Cloud Computing Networking* ;Siamak Azodolmolky, Philipp Wieder, Ramin Yahyapour

[3] *SDN: Development, Adoption and Research Trends* ;Lav Gupta

[4] *Are We Ready for SDN? Implementation Challenges for Software-Defined Networks* ;Sakir Sezer, Sandra Scott-Hayward, and Pushpinder Kaur Chouhan, CSIT, Queens University Belfast

[5] *A roadmap for traffic engineering in software defined networks* ;Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, Wu Chou