Scalable Network Design and Management with Decentralized Software-defined Networking

2019

Kuldip Singh Atwal

University of Central Florida

Find similar works at: https://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

Part of the Computer Sciences Commons

SCALABLE NETWORK DESIGN AND MANAGEMENT WITH DECENTRALIZED
SOFTWARE-DEFINED NETWORKING

by

KULDIP SINGH ATWAL
Masters in Computer Applications, Panjab University Chandigarh, India 2011

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2019

Major Professor: Mostafa Bassiouni

© 2019 Kuldip Singh Atwal

ABSTRACT

Network softwarization is among the most significant innovations of computer networks in the last few decades. The lack of uniform and programmable interfaces for network management led to the design of OpenFlow protocol for the university campuses and enterprise networks. This breakthrough coupled with other similar efforts led to an emergence of two complementary but independent paradigms called software-defined networking (SDN) and network function virtualization (NFV). As of this writing, these paradigms are becoming the de-facto norms of wired and wireless networks alike.

This dissertation mainly addresses the scalability aspect of SDN for multiple network types. Although centralized control and separation of control and data planes play a pivotal role for ease of network management, these concepts bring in many challenges as well. Scalability is among the most crucial challenges due to the unprecedented growth of computer networks in the past few years. Therefore, we strive to grapple with this problem in diverse networking scenarios and propose novel solutions by harnessing capabilities provided by SDN and other related technologies. Specifically, we present the techniques to deploy SDN at the Internet scale and to extend the concepts of softwarization for mobile access networks and vehicular networks. Multiple optimizations are employed to mitigate latency and other overheads that contribute to achieve performance gains. Additionally, by taking care of sparse connectivity and high mobility, the intrinsic constraints of centralization for wireless ad-hoc networks are addressed in a systematic manner. The state-of-the-art virtualization techniques are coupled with cloud computing methods to exploit the potential of softwarization in general and SDN in particular. Finally, by tapping into the capabilities of machine learning techniques, an SDN-based solution is proposed that inches closer towards the longstanding goal of self-driving networks. Extensive experiments performed on a large-scale testbed corroborates effectiveness of our approaches.

iii

To

My mother, Veejaan (वीजां, वीजां)

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF FIGURES

xii

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The astounding success of the Internet is one of the most remarkable breakthroughs that have a profound impact on humankind. Due to this stellar growth, computer networks get increasingly complex to deploy and operate, despite many ingenious developments along the way. Many types of equipment, proprietary software, and prolonged standardization process slowed network innovation and made it hard to manage. Part of the problem is to focus on individual protocols, models, and devices for solving every single problem in a unique way. As a result, network operators develop their own ad-hoc tools and scripts to design and maintain the networks. The lack of a holistic approach for deployment and troubleshooting has made it difficult to manage the networks in an efficient way. Furthermore, the static configuration of traditional networks does not allow dynamic reconfigurations or fault-tolerance mechanisms for the adaptive load changes or the fluctuating environment networks operate in. The tightly coupled control and data planes also hinder flexibility and agility to update the protocol stack. All these factors led for rethinking the way we design and interact with our networks and paved the way for a new paradigm called "network softwarization", which is the main theme of the following sections and subsequent chapters of this dissertation.

1.1 Software-defined Networking

Software-defined networking (SDN) is a recently emerging technology that strives to change the way we design and manage networks. The most striking features of SDN include separation of the control and data planes, programmable network elements, and the centralized control with a global network view. By splitting from the control logic, the forwarding fabric is greatly simplified in order to provide opportunities for the decision making rules to be pushed dynamically without disrupting the whole network. The centralized approach for the control plane allows net-

1

work management from a vantage point and enforces differential policies by considering broader goals of the overall system. The centralized control plane is realized by a controller node that is responsible for all decision making tasks which handle the forwarding devices for smooth network operations. However, it is important to emphasize that a single controller node may not be adequate for performance, reliability, and scalability of a production network. Therefore, the preferred approach is to design a logically centralized but physically distributed control plane with multiple controller nodes. The programmability aspect is attained by two different means. First, the controller instructs the forwarding devices with an open and well-defined programming interface that all elements of an SDN-based network conform to. The OpenFlow is a notable example of such an interface. Second, the network operators and programmers develop high-level network functions in form of applications and submit these applications to the controller that is responsible to deploy low-level policies at the infrastructure-level to meet requirements of the programmers.

The SDN-based data plane is formed by forwarding devices or switches that can be programmed with vendor-agnostic APIs. Such switches contain flow tables that implement the logic provided by the controller. Specifically, the flow table consists of multiple flow entries of match fields, priority, counters, instructions, and flags among other components. By extracting the packet header fields and the metadata, the match field is used to filter a set of packets that belong to a particular flow. Wildcards can also be applied to aggregate a set of flows. Then a set of actions specified by the controller are performed on the matched packets. The actions can range from simply forwarding a packet to a specific port, to rather sophisticated operations such as adding a packet to a specified queue for QoS support, directing a packet to a specified meter for rate control, or push-pop tags of VLAN or MPLS headers. In case a switch does not match an incoming packet with any of the match fields, the packet is either dropped or forwarded to the controller for further processing. To make room for new entries, the existing flow table entries are removed either after a hard time-out or due to being idle for long. Therefore, the controller has to setup flow table entries

2

from time-to-time so that switches know how to handle incoming traffic. Apart from the flow table setup, the controller also interact with switches for troubleshooting, network monitoring, and statistics collection that serve to maintain global network view. For all these operations, either controller or switch can initiate communication on a secure channel. The flow table maintenance and other management tasks can either be performed over a separately dedicated network (out-of-band connection), or the same channel can be utilized which is being used for data transmission (in-band connection).

1.2 Network Function Virualization and Cloud Computing

Complementary but independent to SDN, network function virtualization (NFV) is another significant technology that emerged around the same time. NFV was proposed to improve deployment of network services by leveraging capabilities of state-of-the-art virtualization technologies and commodity hardware that is cheaper to reprogram and maintain. Similar to SDN, NFV decouples the network services from the infrastructure layer by implementing network appliances in form of software modules which can be dynamically managed on the physical or the virtual abstraction layer. The capability to dynamically add, remove, migrate or upgrade the network functions depending on adaptive requirements provides immense opportunities to optimally provision and utilize physical resources without compromising performance or incurring a high cost. Like an SDN controller, an NFV orchestration engine is envisioned for the centralized control and management of network functions. Therefore, SDN and NFV are closely related in terms of certain features, but are completely different from the system architecture perspective. Particularly, NFV enables software implementation of network functions to reduce the cost of specialized hardware, while SDN simplifies network management with centralized control, programmable forwarding devices, and open interfaces. With this distinction in mind, SDN can serve NFV by providing cen-

tralized control for network functions, and similarly, NFV can be used to virtualize SDN controller or other components.

Considering the business model and design layout of cloud computing, SDN and NFV have the potential to play a significant role in this widely accepted computing paradigm. More precisely, SDN principles can be applied for data center network management. Additionally, SDN-based cloud federation can streamline multi-vendor service providers that give the end-users an opportunity to select the most suitable vendor among a wide range. NFV allows dynamic resources scaling, live migration, and cost-effective services availability, which makes it a favorable technology for on-demand and pay-per-usage service model of cloud computing.

1.3 Network Softwarization

The convergence of SDN, NFV, and cloud computing is ushering a new era of network softwarization. The futuristic network designs are predominately based on concepts such as programmability, virtualization, open and vendor-neutral interfaces, and centralized network management. This trend of network softwarization is expected to simplify network management, accelerate transformations, and reduce deployment and maintenance cost of the next generation networks. Reliance on software components by mitigating specialized hardware constraints is being considered as a major driving force towards this paradigm shift. However, this trend also introduces many technical challenges that were non-existent in traditional networks. We highlight some of these challenges throughout this dissertation. Another aspect worth attention is that network softwarization is not a solution to all networking problems. SDN is a tool for simplifying network management, that allows the development of new solutions to the longstanding problems. The centralized control still relies on the low-level rules and statistics collection mechanisms. Therefore, network softwarization and the enabling technologies are at their infancy stage with promising outcomes at the

outset, but require innovative vision and insightful thinking from a long-term perspective.

1.4 Key Contributions

The scalable network design with SDN principles is the prominent focus of this dissertation. We propose a peer-to-peer control plane architecture that addresses crucial challenges of initial SDN prototype, enabling large-scale deployment of SDN-based networks in practical scenarios. Apart from scalability, our proposal also deals with overheads, reliability, heterogeneity, and simplistic network management, which are prerequisites of any modern network design.

Secondly, we outline the challenges of considering SDN for wireless, mobile, and ad-hoc networks. The practically deployable architectures for wireless access networks and vehicular networks are proposed, that cope with limitations of SDN for such volatile scenarios. Without compromising the benefits of SDN, efficient mobility management and QoS techniques are presented to handle rapid mobility with handovers and provide end-user satisfaction. Additionally, some practical applications and benefits of our approaches are also discussed.

By effectively utilizing recent advances in machine learning, network softwarization, and large-scale data processing platforms, we propose a framework, called DeepSDN, which strives to connect the dots towards longstanding goal of self-driving networks. Specifically, we extend the earlier proposed parameter server architecture for distributed machine learning problems and presents a revised parameter server that uses centralized control, global view, and programmability features of SDN for implementing the learning techniques to achieve optimized control and management.

Finally, a reference framework for cloud-based SDN experiments is presented that itself is based on SDN principles. We designed a testbed platform of this framework and used it to perform all experiments for this dissertation. The comparative evaluation results validate the effectiveness of

5

our approaches.

1.5 Dissertation Organization

The dissertation is organized as follows. Chapter 2 describes the existing research related to our work. Chapter 3 presents a reference framework for the evaluation testbed that we use for all experiments in the dissertation. A scalable control plane architecture for SDN is proposed in Chapter 4. Furthermore, Chapter 5 outlines an SDN-based architecture for mobile access networks. Subsequently, Chapter 6 demonstrates an SDN-based approach for mobility management and QoS support for vehicular ad-hoc networks (VANETs). Chapter 7 introduces a machine learning based architecture that aims to fulfill the goal towards self-driving networks. Finally, Chapter 8 discusses the conclusion of the dissertation.

CHAPTER 2: LITERATURE REVIEW

This chapter is dedicated to the related research studies that helped to formulate this dissertation. The first section presents the foundation and use-cases of SDN. Then, the scalability aspect of SDN is elaborated with respect to existing literature. Furthermore, the challenges pertaining to consideration of SDN for wireless and mobile networks are discussed along with various solutions proposed by the research community. And finally, consideration of SDN for self-driving networks is presented in last section.

2.1 Introduction

Although the quest for the programmable networks exists since the late 90s [1], [2], OpenFlow brings the major breakthrough when a programmable switch was introduced for Stanford University campus network [3]. Initially it focused on a short-term question that "As researchers, how can we run experiments in our campus networks?". And as expected, this idea rapidly expanded to other campuses and enterprise networks. The other similar concepts were emerging around the same time, ForCES among the most notable ones [4]. Apart from the programmable switch, Open-Flow also borrowed the concept of the control plane and the data plane separation from SCP and RCP [5], [6]. A standalone entity called "controller" was introduced to manage forwarding devices with an open interface. The controller enabled the centralized control of the network from a vantage point, that distinguished the OpenFlow-enabled networks from conventional networks which operates on the principles of distributed computing. All these developments led to a new networking paradigm called software-defined networking (SDN). The centralized control, programmability, global network view, open interfaces, and the dichotomy of data and control planes emerged as the pillars of SDN. The Open Networking Foundation (ONF) was formed as a collaborative commu-

7

nity to streamline SDN-related projects and to lay the groundwork for software-defined standards. The ONF provided OpenFlow-switch specification that became the de-facto norm for SDN-based networks. However, the intrinsic features of SDN introduced new challenges for the networking community which were not applicable in conventional networks. Some of these challenges are discussed in the following sections.

2.2 SDN Scalability

The concern of SDN scalability stems from the centralized control, which is one of its core propositions. A standalone controller introduces single-point-of-failure, capacity and performance bottleneck, and geographic constraints, among other challenges. A benchmark study reports that a single SDN controller can handle 1.6 million requests per second with an average response time of 2 ms [7]. However, it suggests that a single physical controller is not enough to manage a sizable network and multiple controllers are needed in order to maintain high availability and low response time. Therefore, the next reasonable step forward was to have a logically centralized but physically distributed control plane that retains the properties of SDN and achieves performance comparable to conventional networks. Subsequently, many control plane designs were proposed that were composed of either flat or hierarchical layer of multiple controllers which exploits centralized databases or file systems to maintain a consistent and logically centralized network view. Hyberflow [8], Kandoo [9], and Orion [10] are the notable works in this line of research. A recent work presents a dynamically scalable control plane that manages congested controllers and in-band control channels using control flow tables [11]. A comprehensive survey of SDN control plane scalability highlights the major challenges that must be addressed for SDN deployment at large-scale [12]. As discussed in Chapter 4, we propose a peer-to-peer control plane architecture that addresses the scalability aspect of SDN in a systematic manner.

8

2.3 Challenges and Considerations of SDN for Wireless Realm

Although the original OpenFlow switch was designed for wired networks, it didn't take a long time to be considered for the wireless networks as well [13]. The benefits of the key use-cases of SDN motivated the wireless community to apply softwarization principles for tackling longstanding issues of wireless domain. Therefore, the software-defined wireless networking (SWDN) paradigm emerged as a new building block, with the goals to achieve SDN-like success [14]. However, wireless and mobile networks possess distinct characteristics which do not exist in case of wired networks. Unpredictable mobility, unreliable channels characteristics, limited resources, and unstable topology are such properties of mobile wireless networks that need a judicious attention. Therefore, SDN principles cannot be applied for the wireless domain in a straightforward manner. Considering these aspects, Li et al. posit that SDN can simplify the design and implementation of cellular networks, and proposes an extended controller platform to address scalability and other challenges of SDN [15]. Similarly, Gudipati et al. argue that the existing control plane of LTE is suboptimal to efficiently utilize limited spectrum for connectivity purpose [16]. To address this problem, they propose SoftRAN, which is a centralized control plane architecture based on SDN principles. MobileFlow is another SDN-based architecture for mobile carrier networks that infuses innovation in the mobile networks by enabling open interfaces and APIs to roll out new network features in a limited budget and less time [17]. A similar approach for the wireless mesh networks is followed in [18]. A detailed survey of software-defined and virtualized wireless mobile networks is presented in [19]. Chapters 5 and 6 of this dissertation elaborate our approaches of considering SDN for wireless and ad-hoc networks respectively.

2.4 SDN for Self-driving Networks

The crucial challenges of the ever-growing network control and management are discussed in [20], that also outlines the blueprint towards self-driving networks. This work also stresses the suitability of statistical inference and machine learning techniques for prediction problems. DeepRM describes a deep reinforcement learning solution that translates the network resource management problem into a learning problem [21]. It mentions that resources management problems are ubiquitous in computer networks and the prevailing wisdom of using heuristics is not a suitable way to solve such problems mainly due to lack of accuracy, flexibility, and complexity. [22] presents a machine learning based tool called WISE that is capable of predicting the effects of probable configuration and deployment changes in content distribution networks. A new paradigm called "Knowledge Defined Networking" is proposed in [23], that integrates machine learning, data analytics and SDN concepts to the existing Internet architecture. COBANETS utilizes unsupervised deep learning, probabilistic models, and network virtualization techniques for network optimization [24]. ANEMA is an autonomous network management architecture with self-optimization and self-healing properties that achieve autonomic behaviors in the network components [25]. A predictive resource scaling mechanism for cloud systems is proposed in [26], that addresses SLA violations and over-provisioning of resources. As discussed in Chapter 7, our contribution is mainly based on the concept of the Knowledge Plane for Internet [27] and a distributed framework, called parameter server, for highly scalable deep learning models [28]. We extended these concepts with SDN principles and propose a practical framework for cloud data centers.

CHAPTER 3: A NOVEL APPROACH FOR SIMULATION AND ANALYSIS OF CLOUD DATA CENTER APPLICATIONS [1]

Considering the unprecedented surge in demands and expectations of smart cloud-based services and applications, there is an increasing demand of scalable tools that can effectively evaluate the performance of these services and provide insights for improving their design and implementation. A realistic and accurate emulation platform is a prerequisite for the efficient assessment of smart cloud data centers, and for improving the capability and flexibility of utilizing the vast resources available to smart cloud applications. The platform needs to be scalable and diverse enough to handle out of the box experiments, as well as simple enough for ease of use and management. The overheads incurred to meet these goals directly hamper the performance of a framework. Therefore, mitigation of the overheads adds to the salient features. In this chapter, we present a reference model that strives to meet such requirements while addressing overheads. We demonstrate proof of the concept using off-the-shelf software components and present some test cases of the performance results obtained by the implementation of our platform.

3.1 Introduction

Smart cloud computing technologies allow companies and users to focus more on innovation rather than infrastructure and resources. The sheer diversity of applications (for example, healthcare, image optimization, traffic engineering, to name a few) imposes varied configuration, extensibility, customization requirements. It is not feasible to use the production networks to evaluate vastly heterogeneous nature of applications. An evaluation platform is necessary for dynamic and flexible

---

[1]Related Publication: K. S. Atwal and M. Bassiouni, A novel approach for simulation and analysis of cloud data center applications, in Proceedings of the IEEE International Conference on Smart Cloud. IEEE, 2016, pp. 164169.

application modeling. The viable options to perform networking experiments is to employ testbeds, emulators, simulators, or variation of any of these techniques. However, benefits, drawbacks, and the importance of using such tools in isolation are well known to the research community [29], [30]. Furthermore, the essential requirements, such as high performance, flexibility, scalability, realism, granularity, simplicity, extensibility, overheads, robustness, are the characteristics that leave no option to come up with a framework without having some trade-offs.

The data center environments, the building blocks of cloud computing, span from hundreds to millions of computing components. The federated clouds are gaining prominence to establish global geographic presence and proximity to the end users for content delivery network (CDN) type services, that directly contribute towards CapEx and OpEx savings.

One of the foremost facets that contribute to the widespread penetration of cloud computing is its ability to dynamically adapt resources provisioning with respect to peculiar user demands or other constraints (e.g., time-dependent services, load-balancing, burst traffic, QoS). Hybrid clouds, which impose different ownership, authorization and authentication requirements, are the specific traits of clouds that are being widely adopted. Therefore, it becomes imperative to consider these aspects while evaluating such environments.

Many evaluation tools are available for analysis of cloud systems. However, each tool is developed to perform specific kind of experiments. Therefore, these tools are not suitable to fulfill the above mentioned exhaustive set of requirements. Furthermore, a minimum level of uniformity lacks from these platforms. In this chapter, we propose a generic reference model for evaluation platforms to overcome these limitations. The simulation and emulation tools are incompatible with each other due to independent developments, varying requirements, and customizations etc. Therefore, a reference model can be seminal for interoperability and portability.

Software-defined networking (SDN) is recently emerged as a promising paradigm, that is con-

12

sidered to revolutionize the way we think of conventional networking [31]. SDN simplifies the network orchestration by separating the control plane and data plane. Furthermore, programmability, centralized control, flow-based decision making, are the other important concepts of SDN. It would be interesting to apply SDN principles to envisage the reference model for the evaluation platforms.

Following are the major contributions of this chapter:

- It underlines the ever evolving design requirements to model smart cloud services and applications.

- A generic reference model is proposed to meet the broad spectrum of requirements in a systematic and structured manner. The model is implemented using off-the-shelf software components, integrated with small tweaks to enhance adaptability.

- The proposed model is evaluated to assess its feasibility and applicability. We present the results obtained through a number of experiments performed with the implementation of our platform.

Rest of the chapter is organized as follows. Related work is discussed in Section 3.2. Section 3.3 presents the proposed reference model. Then, evaluation of the proposal is given in Section 3.4, followed by an elaboration of results in Section 3.5. Finally, Section 3.6 describes the conclusion and future scope of the work.

3.2 Related Work

SDN-based cloud simulator is developed in [32]. CloudSimSDN is based on CloudSim [33], that extends functionality of the existing platform by incorporating SDN principles in it. However, the

13

platform lacks the ability to harness virtualization technologies and also does not support realistic simulation environments considering middleboxes and other features. Many other platforms extend or improve the CloudSim as per specific requirements. SENDIM is an integrated framework that aims to simplify development and management of heterogeneous components of clouds by following SDN principles [34]. Centralized orchestration of cloud infrastructure is the main consideration of SENDIM. NS-3 is a very popular platform that has capabilities of a simulator as well as an emulator [35]. Mainly due to its extensible and flexible architecture, NS-3 can simulate a diverse set of networking scenarios and topologies. However, NS-3 provides limited support for realistic SDN paradigm. A cloud-based distributed platform is proposed in [36]. However, it aims to simulate the distributed computing scenarios by exploiting the capabilities of cloud computing, rather than specifically evaluating the models of cloud systems. Mininet-based emulation platform for clouds is proposed in [37]. CloudNaaS framework is presented in [38], that aims to provide better granularity of the network services to the customers. EMUSIM is proposed in [39], that integrates emulation and simulation to perform cloud computing experiments. The main focus of EMUSIM is to evaluate the real applications running in the cloud rather than their software modeling, so that the accuracy and understanding of applications could be improved for performance and cost gains. MobiCloud is the platform to evaluate requirements and efficiency of communication between mobile infrastructure and the cloud resources [40]. A detailed survey of cloud computing simulation tools is presented in [41].

Most of the available platforms for evaluation of cloud data centers either do not follow the principles of SDN paradigm or impose significant constraints on extensive analysis of clouds features. Furthermore, many existing evaluation platforms do not support multi-data center scenarios. And, several tools are developed by keeping very specific requirements in mind, that limit the scope for flexibility and extensibility.

Advantages of realizing the OpenFlow-based SDN principles in enterprise or carrier-grade net-

14

works are explained in [42], that also presents the crucial challenges being faced by the design and management of evolving cloud technologies. SDN-based cloud computing architecture is proposed in [43]. OpenFlow-based SDN deployment in globally distributed data centers is described in [44]. B4 deployment shows the optimum resources consumption while minimizing the overheads (nearly 100% links utilization is reported).

3.3 System Model

Motivated by the requirements highlighted in the introduction section, we incorporate the following characteristics in our reference model:

- Layered design

- Centralized Control

- Modularity

- Programmability

- Interoperability

Each of these characteristics is briefly described below.

3.3.1 Layered Design

The cloud systems follow abstractions at various levels to achieve flexibility and management. And, layering is the natural concept to implement abstractions in computer networking. Hierarchy of layers also helps to attain scalability and robustness. Therefore, following the convention, we

15

also formulate our model in layers. Although not strictly enforced, following is the blueprint of favorable layers to implement the model:

Cloud applications layer: Applications and services can either be provided by the smart cloud provider or deployed by users themselves. It also includes middleware frameworks as applications. The application layer provides APIs to implement control functionality or any kind of business logic.

Platform layer: Cloud providers allocate the computing resources to users as per their requirements. Users deploy their own software solutions without dealing with the underlying hardware or software complexities. The major distinction between platform layer and control layer (described below) is that users need not control or manage the underlying cloud resources, but perform their operations on the available settings. However, users have full control over whatever is deployed by themselves on top of the provided infrastructure.

Control layer: Logically centralized control plane is decoupled from the application logic or physical resources. The reasoning for this separation is to have provision for all components of the system to evolve independently. Also, the management of entire network can be easily done by the centralized control.

Abstraction layer: Various virtualization technologies are already available and others are still evolving that can be utilized for on-demand resources profiling in smart cloud environments. The underlying physical infrastructure needs to be virtualized to perform a seamless migration, rapid elasticity, load-balancing, and other operations. Resource pooling can also be done to serve multiple customers using the multi-tenant model. Physical and virtual data planes can be incorporated into the data center networks.

Physical layer: Massive physical resources are abstracted by leveraging virtualization techniques

to achieve resource adaptation, mobility, security, fault-tolerance etc. Easy reconfiguration and adaptation of hardware resources in the cloud infrastructure is a crucial aspect.

As per requirements, these layers can be implemented anywhere in the hierarchy of the proposed model. For instance, the control layer can be implemented in the distributed control plane, and abstraction and physical layers can be implemented in the data center infrastructure.

3.3.2 Centralized Control

The centralized control is an important ingredient of SDN design philosophy. It simplifies the development of cloud-based networking services and applications, and assists to overcome error prone enforcement and modification of policies. The centralization of global network view helps to achieve efficient resources utilization in highly demanding environments. Furthermore, the centralized network monitoring enables easier maintenance and troubleshooting as well as efficient power utilization. Although the centralized control eliminates the complications of network management, it has its own limitations. Specifically, the centralized control may suffer from scalability, resilience, and security issues among others. Therefore, it is important to deal with these challenges while taking benefits of centralized control. To address these limitations, we introduce the concept of distributed controller and proxy controller. As explained below in detail, the proxy controller alleviates the requirement of distributed state management while providing the global network view, and the distributed controller provides scalability and robustness while mitigating the overheads by having peer-to-peer communication among nodes.

17

3.3.3 Modularity

A modular architecture is the prerequisite of any software system that allows it to evolve rapidly and meet extensibility needs of its users. Any user-centric platform should be designed in such a way that its users can fine tune and customize it for specific purposes. It should provide enough support for the development and integration of new models or subsystems in a plug-and-play fashion. Besides providing flexibility, a modular architecture also simplifies troubleshooting and maintenance, since any issue can be tracked, isolated, and fixed in a systematic way without perturbing the other independent components. As mentioned earlier, modularity in our model is realized by having a layered structure.

3.3.4 Programmability

Programmability is the complementary feature of modularity that provides opportunities to implement customized software components on-top-of or beside existing platforms. The APIs exposed by the programmable systems must be flexible enough to carry unimagined tasks, as well as simple and accessible enough for usability. Programmability is also adapted in SDN stack to control the forwarding devices (data plane) from a vantage point. Users implement the required functionality in the form of applications by using favorable language and submit to the controller. The controller is responsible to deploy the sought functionality at the forwarding devices. Similarly, network administrators can compose and enforce policies at the data plane via the control plane.

3.3.5 Interoperability

Interoperability enables compatibility, portability, and communication among unrelated components. Considering the heterogeneity of technologies, devices, and applications of smart clouds,

18

interoperability becomes a crucial feature of such environments. For a highly adaptable and usable model, it should be neutral to any particular platform and also should not be dependent on any specific technology or software component implementation. Federation of multiple cloud providers is an apt usability of interoperability. The major task is to find a common set of specifications and interfaces to achieve the desired level of interoperability among systems. Our proposed model is a step forward towards this task.

With these characteristics, Figure 3.1 shows the layered taxonomy of the proposed reference model. The data center infrastructure is separated from the control sub-system. Furthermore, the control plane is designed in a hierarchy that is managed by two different controllers. The middle layer is a distributed controller with multiple nodes (labeled Node 1, Node 2, ..., Node n) to scale out at a large level. Each node is responsible for handling a set of physical resources at the underlying layer. The nodes communicate with each other via communication channels to share the relevant information that is not available to a particular node. By exposing flexible APIs, the distributed controller provides opportunities of extensibility and programmability by allowing programmers and network managers to deploy applications and policies in the form of modules.

The proxy controller at the top layer maintains a global view of the network and is responsible for enforcing network-wide policies. The proxy controller also handles the competing resources sharing among nodes of the distributed controller, and resolve configuration conflicts, if they arise. The main purpose to have a separate proxy controller is to maintain a coherent and consistent network state without incurring much overhead that is inevitable if distributed data store or another similar mechanism is used. Finally, the layers of the hierarchy can be defined recursively as per scalability and granularity requirements.

19
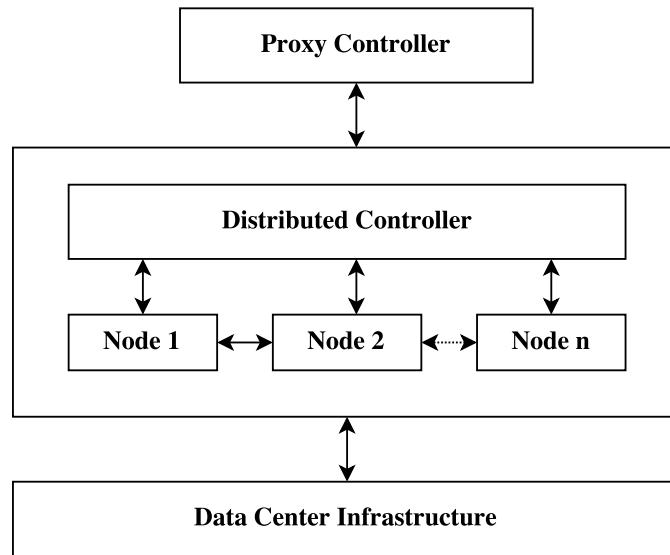
Figure 3.1: A reference model for the evaluation tools for cloud-based applications

Notice that the proposed model is transparent enough to allow abstractions at multiple levels by deploying any of available virtualization tools and techniques. However, the model is most suitable to evaluate cloud environments having the promising non-strict layered distributed design shown in Figure 3.1. The model is not directly applicable to cloud systems with purely strict distributed design. Extending the platform to these cloud systems is a future scope of work.

3.4 Evaluation Platform

To validate the potential of our proposed model, we use ONOS [45], FlowVisor [46], and Mininet [47] as off-the-shelf tools. Some functionality of these tools is tweaked to set up an environment that closely resembles and complies with our model. Initial experiments are performed to highlight the effectiveness of the proposal. A brief description of these tools is given below.

20

3.4.1 ONOS

Open Networking Operating System (ONOS) is an open source implementation of SDN control plane, that allows development and deployment of network control applications. Due to its design characteristics, ONOS can be used for access networks, data center networks, enterprise networks, as well as other networking scenarios. In order to achieve high performance, scalability, robustness, availability, ONOS is designed to operate as a distributed cluster of symmetric nodes. Moreover, modular approach of ONOS provides extensibility and flexibility to implement customized features and services. ONOS exposes sufficient set of APIs to support instant implementation and deployment of network applications or any other business logic. Intent-based programming support by ONOS significantly simplifies the network policy enforcement and application development process. Finally, ONOS is compliant with OpenStack and other clouds related technologies.

3.4.2 FlowVisor

FlowVisor is an SDN-based proxy controller that enables switch-level network virtualization by creating isolated slices of a production network. In its standard form, FlowVisor works as a transparent proxy controller between the control plane and data plane. However, we performed some customizations to deploy it as the proxy controller at the root of our hierarchical model. Besides slice isolation, the centralized policy enforcement, separation of control and virtualization logic, and global network monitoring are the other traits of FlowVisor that make it a suitable choice for our platform. To achieve optimum performance gains as well as maximum resource utilization, while at the same time incurring minimum overheads, we can delegate the workload to the transparent proxy controller and distributed control plane by appropriately exploiting these features.

3.4.3 Mininet

Mininet is the lightweight container-based network emulation tool for instant prototyping of large-scale networks. To scale up to hundreds of nodes, Mininet takes advantage of the already available features of Linux-based operating systems, such as OS-level virtualization, network namespace, virtual ethernet pairs, and processes. The testing scripts and scenarios of Mininet can be deployed to the real network with exactly same code. Mininet emulates links, hosts, and switches to evaluate a network topology. The flexible interface of Mininet allows it to easily configure with any software or hardware switch or controller running in the real or simulated network, provided that the machines have IP-level connectivity. Further, Mininet is extended for the cluster mode support to emulate distributed networks, that makes it suitable to deploy with distributed control plane comprises of ONOS instances.

By default, the global network topology state is cached in memory on each instance of ONOS. And, the joining and leaving nodes are managed by the cluster membership management, that is implemented using Hazelcast's distributed structure. However, these tasks introduce overheads that can be subverted. To overcome the overheads, ONOS instances can be deployed as data center entities, along with a proxy controller, FlowVisor in our test case scenario, at the top layer to centrally configure and manage resources. Therefore, we accordingly customized ONOS and FlowVisor for global network view and nodes management.

At the simplest level, a tree-like topology is used in data center networks where end hosts connect to top-of-rack switches. These edge switches are connected to aggregation switches, which are further connected to core switches in the topmost layer. As shown in Figure 3.2, we designed similar scenario using the tree topology provided by Mininet and varying fanout to increase the number of switches and hosts. Figure 3.3 shows the evaluation scenario of the proposed reference model. As in diagram, ONOS instances communicate with each other to share the information

22

required for a local instance; however, the global topology management is handled by FlowVisor.
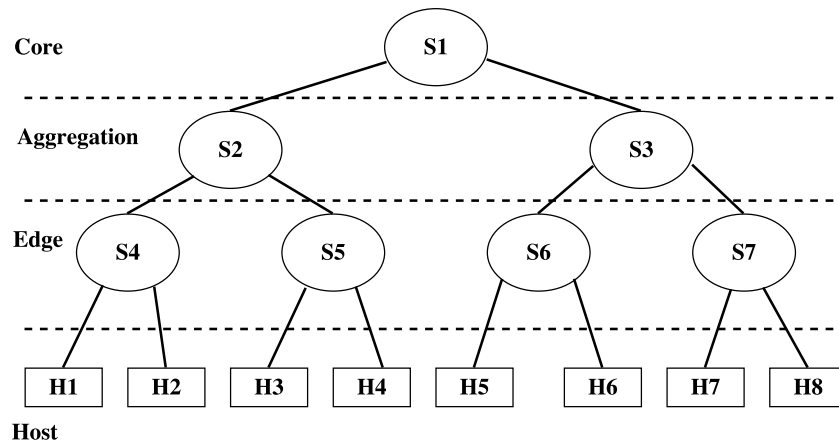
Figure 3.2: Data center topology
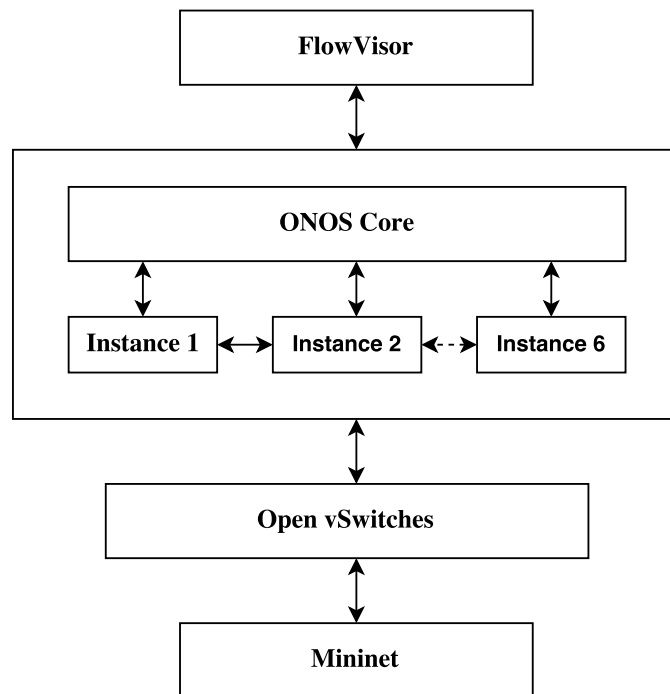
Figure 3.3: The evaluation platform for the proposed model

23

3.5 Preliminary Results

We used Intel (R) Xeon (R) CPU E5-2603, running at 1.60 GHz, with six cores, 64 GB RAM, 2 TB hard disk, and Ubuntu 14.04.1 LTS 64-bit operating system. Multiple instances of ONOS are deployed by having separate Virtual Machines (VMs) for each instance, using the Oracle Virtual Box. Each experiment is averaged by 20 runs, and Open vSwitch [48] is used for all topologies.

End-to-end bandwidth, measured in Mbps, of a varied number of switches and hosts is summarized in Table 3.1. Similar results are presented in [47], but without specifying the type of controller deployed to perform the experiments. Mininet supports several controllers, including the OpenFlow reference controller that is natively implemented in Mininet, and a remote controller. The remote controller can be any off-the-shelf controller running outside of the VM, a different physical machine, or anywhere in the network. The controller type is a crucial configuration component since the reference controller and remote controller significantly affect measurements. To benchmark the difference, we deployed the linear topology with exactly same number of switches and hosts as was done in [47]. The variation is highlighted in Figure 3.4. The factor for this difference is the use of a remote controller that significantly decreases the bandwidth. And, by increasing the number of switches and hosts, the total available bandwidth is shared and reduced for each device; however, the platform provides usable bandwidth in aggregation. We use ONOS as the remote controller, but the results may vary for other types of controllers.

24

Table 3.1: End-to-end bandwidth

Switches	Hosts	Bandwidth (reference)	Bandwidth (remote)
1	2	26624	269
10	10	15360	24.4
20	20	11264	9.66
40	40	7127	3.65
60	60	5079	2.30
80	80	3604	1.66
100	100	2467	1.23
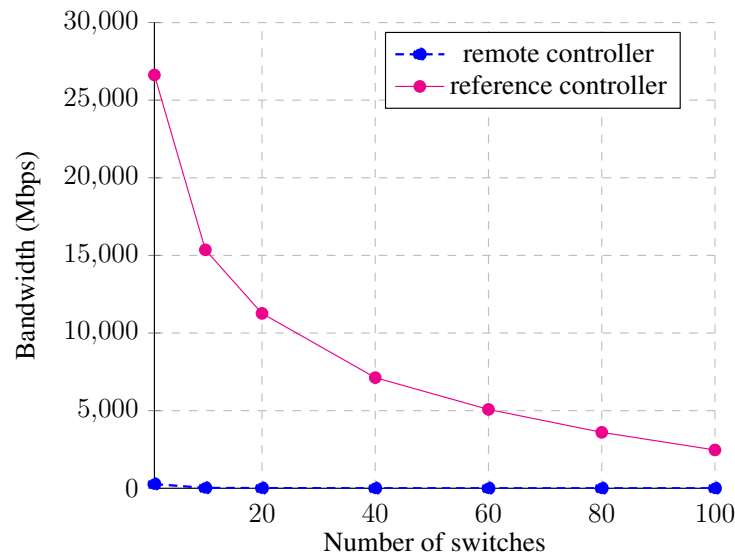
Figure 3.4: Comparison of the remote and reference controller bandwidth

Figure 3.5 shows the time required to setup and tear down a topology in cloud environment. Since our main focus is the switching system of clouds, the graph shows only that particular information. More details of the topology are given in Table 3.2. We can observe that although time grows rapidly for larger topologies and further optimizations are possible, still the waiting time to start a

25

topology is reasonable considering the booting time required for switches.
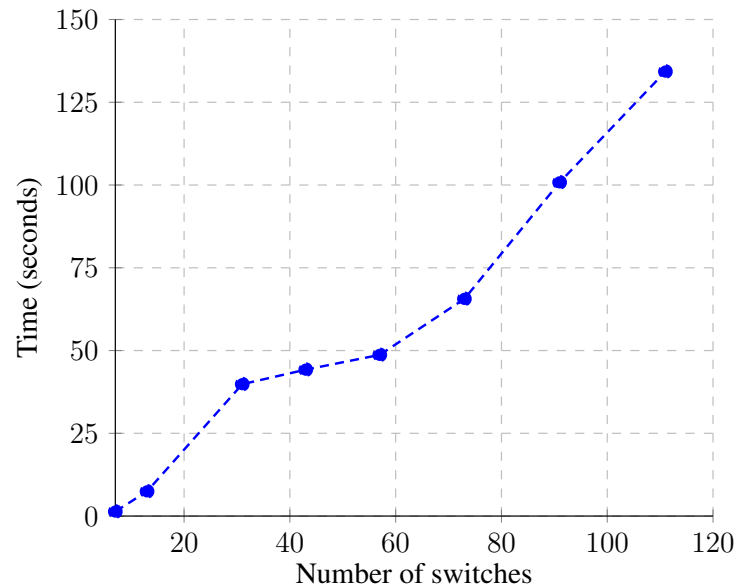
Figure 3.5: Topology setup and tear down time

We measured CPU utilization to assess overhead incurred by the platform. As in previous experiments, the number of switches and hosts were varied to get the relative performance metric of the simulation environment. Figure 3.6 shows the graph of CPU consumption. It indicates that the rate of growth is fairly low. Therefore, we can infer that the platform carries out large-scale operations while minimizing overheads.

26

Table 3.2: Topology setup and tear down time

Switches	Hosts	Time (sec)
7	8	1.364
13	27	7.465
31	125	39.840
43	216	44.228
57	343	48.700
73	512	65.612
91	728	100.812
111	1000	134.277
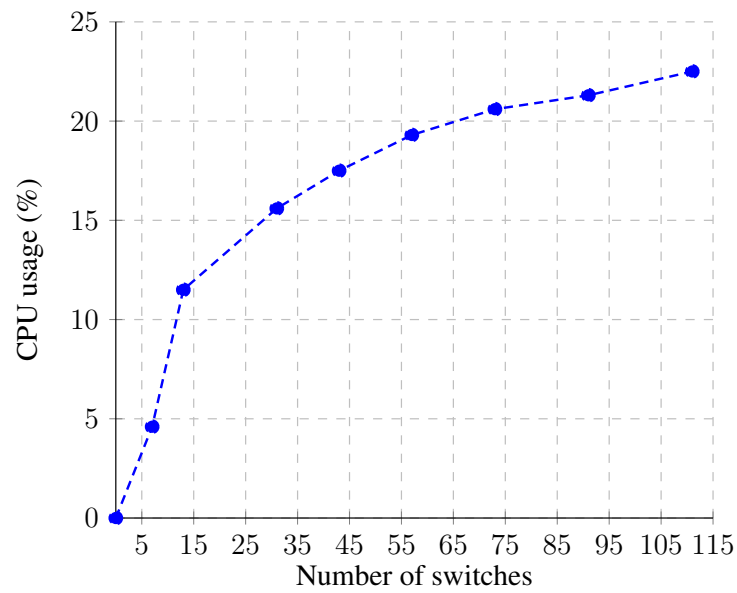
Figure 3.6: Overhead of CPU utilization

3.6 Conclusion and Future Work

An exhaustive and realistic evaluation platform is essential to architect smart cloud computing systems. Various tools already exist for simulation and analysis purpose. However, most of the available tools somehow fall short of providing a level of realism and flexibility. Moreover, lack of conformity is another major issue. Therefore, we propose a generic reference model, that is carefully designed to meet an extensive set of requirements. The proposed model can be adopted to implement a concrete platform, and we have demonstrated this by using existing software components. Initial results are also presented to demonstrate the potential usefulness of the model. As future work, we would like to assess our model for support of exhaustive test cases including, Service-Level-Agreement (SLA) testing, on-demand testing, and embedded testing.

CHAPTER 4: A SCALABLE PEER-TO-PEER CONTROL PLANE ARCHITECTURE FOR SOFTWARE DEFINED NETWORKS [1]

Control plane scalability is one of the major concerns in Software Defined Networking (SDN) deployment. Although the centralization of the control plane by decoupling it from the data plane facilitates ease of network management, however, it introduces new challenges. One of these challenges is to maintain performance, consistency, and scalability while minimizing the corresponding overheads. In this chapter, we propose an architecture that allows the control plane to evolve at a hyper-scale level as well as address important performance and reliability issues. A hierarchical control plane architecture with peer-to-peer communication among logically distributed controllers is designed with the goal of achieving optimum performance and consistency gains while mitigating overheads. A root controller is deployed at the top layer of the hierarchy to maintain global network view. The proposed model is helpful in improving network robustness against failures and supporting a desired level of reliability. To evaluate our model, we developed a realistic emulation platform using ONOS, FlowVisor, Mininet, and Open vSwitch. The proposed architecture is compared with earlier solutions and experimental results are presented to demonstrate the effectiveness of the proposed model.

4.1 Introduction

Software Defined Networking (SDN) has gained very much attention from academia as well as the industry in recent times. It is envisioned to overcome the existing shortcomings of data center networks, access networks, and enterprise networks. Learning from prior experiences [49], separation

---

of the control plane and the data plane is a crucial aspect for achieving SDN goals. Furthermore, centralized control, network programmability, and flow-based decision making are some of the other important features of SDN. Due to its salient features, SDN is being widely deployed in multi-datacenter and multi-domain networks by using OpenFlow protocol.

However, there are many challenges still need to be addressed. The control plane scalability is one of the prominent challenges among others. The issue of scalability escalates even further if we consider performance and robustness of the network, which are generally the prerequisites of any realistic network. Therefore, it becomes imperative to address the control plane scalability in a systematic manner by considering the other important aspects of a pragmatic network. In this chapter, we propose a model to deal with scalability, robustness, and performance of SDN control plane architecture. Although a single controller can handle sufficiently large number of requests with an acceptable average response time [7], still there are other factors that require multiple controllers deployment [50], such as:

- Geographically wide distribution of the network.

- High availability and low response time requirements for QoS or real-time services.

- Handling bottleneck of the single point of failure.

- Partitioning large-scale networks (e.g. data center, enterprise networks) into many sub-networks, that can be controlled separately.

- Management of segregated inter-networks by different proprietary domains.

There are multiple ways to deal with the scalability of the control plane. First, the performance of the physically centralized control logic (i.e single controller) can be increased by deploying more hardware resources or performing some optimization techniques for performance enhancements

30

up to a certain level [51]. Second, the overall workload of the control plane can be alleviated by minimizing the set of operations performed by it and devolving some functionality to other components [52]. Third, multiple controllers can be deployed that form a physically distributed or logically centralized control plane [53]. Given the earlier reasons for requirements of multiple controllers, we chose the last option to address the control plane scalability issue; therefore, other approaches are out of scope. Furthermore, multiple controllers could be deployed in a fashion such that the control plane is fully decentralized and physically distributed [54], or logically centralized [55]. Our approach maintains logically centralized but physically distributed control plane.

Following are the major contributions of this chapter:

- We propose highly scalable control plane architecture for SDN, that can be adopted to meet optimum performance demands of enterprise and data center networks.

- Consistency of the network state is handled very efficiently while mitigating the corresponding overheads.

- The proposed model allows the network to be configured dynamically as per desired level of robustness requirements.

- An integrated emulation platform is developed to evaluate the proposed model, and corroborative results are presented.

Rest of the chapter is organized as follows. Related work is discussed in Section 4.2. Section 4.3 presents the system model of the proposed architecture. Then, evaluation platform for the proposed scheme is described in Section 4.4, followed by an elaboration of results in Section 4.5. Finally, Section 4.6 includes the conclusion and future scope of the work.

31

4.2 Related Work

Our model is motivated from Orion [10], which is a hierarchical control plane architecture for large-scale networks. Our proposal differs from Orion in many respects. First, the top layer of controllers is not distributed. Rather, a root controller is placed in the hierarchy to maintain the coherent network view while incurring minimum overheads. Secondly, communication channels among zone controllers are provided to serve information requirement of any controller at relatively local level. Finally, if any zone controller fails, load will be distributed either to existing neighboring zone controllers or to newly added controller, depending on the adaptive controller provisioning mechanism.

Zoning mechanism for hierarchical network optimization is proposed in [56], that does not consider coherent network abstraction at the application layer. Our model utilizes resources more efficiently by off-loading some functionality to the root controller at the top layer of the hierarchy. Dynamic controllers adaptability and controller-switch assignment/reassignment are proposed in [57], that uses distributed data store to maintain coherent network view and perform load adaptation among multiple controllers. However, load balancing can be handled more efficiently if it is controlled by a physically centralized entity; therefore, we delegate this functionality to the root controller. A distributed hierarchical control plane to improve scalability and service flexibility is proposed in [58]. However, it does not deal with failures, and our solution is more robust against failures at multiple layers. HyperFlow [8] is an event-based, logically centralized but physically distributed control plane architecture for OpenFlow. Although HyperFlow is resilient to network partitioning and component failures, however, it does not consider dynamic controllers adaptation and load balancing, which are crucial functionalities of data centers and other similar networks. Kandoo [9] proposes the two-layer hierarchy of controllers to achieve scalability. However, it also does not consider failure and adaptability of controllers. A formal model on SDN control plane is

32

presented in [59], which shows that the hierarchical organization is required to achieve scalability and elasticity of any practically feasible network.

Most of the existing logically distributed control plane architectures either fall short of robustness against failures or incur the communication or state management overheads that directly hampers the performance. Furthermore, the intense demands of data center type networks are not taken into consideration, such as dynamic resources provisioning based on user demands, optimum utilization of the existing infrastructure to save cost. Therefore, contrary to the state of the art solutions, our proposal uniquely addresses the control plane scalability while making sure to provide robustness and handle overheads in a systematic way.

4.3 System Model

Following are the main challenges for a logically centralized control plane architecture of SDN:

- The global network view has to be maintained coherently across multiple controllers in case of any component failure and dynamic network topology changes.

- The inconsistency and competing resources issues may emerge among multiple controllers that need to be resolved with the priority to achieve overall optimum state.

- In a multi-controller scenario with a global network view, it is important to have the state synchronization and reachability between all controllers.

- The communication overheads are inherently increased in the deployment of multiple components.

- Dynamic controllers adaptation is required for efficient resources utilization. For instance,

33

on-demand resources provisioning is needed in data center networks to fulfill user demands and meet the Service-Level Agreements (SLAs).

Other than that, an effective failover mechanism is needed for resilience and robustness of the network. Considering these aspects, we designed a logically centralized but physically distributed control plane architecture for SDN. By virtue of its design, the proposed model is not strictly physically distributed, since some functionality of the control logic is handled by the physically centralized controller. Figure 4.1 depicts the hierarchical model of our proposed system. The hierarchy is formed in such a way that the relatively local events are handled by the zone controllers and global events are handled by the root controller. The idea is to distribute the load among controllers, along with maintaining the coherent network state, while minimizing the communication and other overheads that contributes towards performance gains.
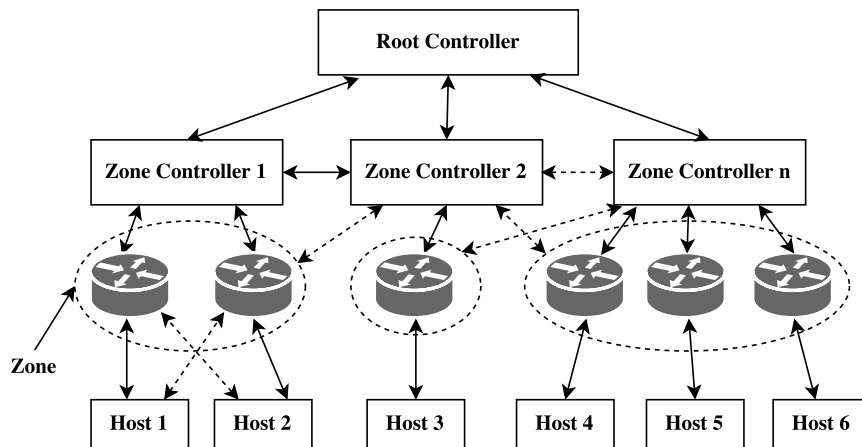
Figure 4.1: Hierarchical SDN architecture.

The root controller is mainly responsible for managing the global network view and some other network-wide functionalities, while the zone controllers directly control the underlying physical devices (i.e forwarding plane) via OpenFlow protocol. Furthermore, the communication channels

www.manaraa.com

are established between zone controllers for peer-to-peer communication. These channels are used to share any kind of information among zone controllers that is not available to a particular controller. Fetching the required information from neighboring controllers contributes to minimizing the latency that can be higher if the information is requested from the root controller at the top layer. Also, due to localized information sharing, the peer-to-peer interaction reduces the communication overheads in the network.

As shown in the diagram, the forwarding plane is formed by partitioning physical resources in zones. The zones can either be created as per segregation requirements of geographically distributed sub-networks, or as per multiple proprietary domain management services. The zones can also be configured for the purpose of evenly distributed load management and optimum resources utilization of a large enterprise. The reliable TCP connections are employed for peer-to-peer communication among zone controllers as well as interaction with the root controller at the top layer. The solid arrows in the diagram represent the established connections among various components, while the dotted arrows highlight the provision to add multiple connections for backup purpose.

The next major task is to categorize the roles and responsibilities of the root controller and the zone controllers. In this aspect, our main focus is to delegate functionalities so that the consistent network view is maintained without many overheads, and service requests are served as per proximity of the components. With these goals, the following subsections elaborate functionalities of the control layers and describe the modules defined to implement such functionalities.

4.3.1 Root Controller

Below are the main responsibilities of the root controller:

- Maintain the globally consistent network view and share it to the zone controllers, applica-

35

tion layer, and other services.

- Configuration of network components, such as zone controllers, switches.

- Dynamic provisioning of the zone controllers for average resources utilization, and failover mechanism enforcement in case of failures.

- Perform controller-switch assignment/mapping and initiate the switch migration when needed.

- Network statistics collection via zone controllers.

- Rules generation for network-wide policies enforcement.

Following are the modules defined to implement these tasks of the root controller:

Storage module: The network state information is stored in the storage module. The physical centralization of the network topology information alleviates the overheads of the distributed data stores or shared file systems.

Monitor module: The desired level of consistency in the network state is dependent on the real time topology changes and other events in the network. To achieve this objective, the monitoring module is carefully defined to continuously observe the network state and find a "sweet spot" or an optimal point having sufficient level of accuracy while minimizing the corresponding overheads.

Load adaption module: Optimum resources utilization is an important priority of the data center networks and enterprise networks. By getting the real time usage updates from the monitoring module, the load adaptation module provides dynamic provisioning of the network resources. It implements the load balancing methods that make sure to efficiently utilize the capacity of the zone controllers.

Partitioning module: The logically centralized but physically distributed control plane implies segregation of workload among multiple controllers. It closely resembles the multi-tenant and multi-domain model of the modern data centers. By implementing the clustering techniques, the

partitioning module creates slices of the network as per specific requirements and assign the slices to the zone controllers in an optimized way.

4.3.2 Zone Controllers

Below are the major responsibilities of the zone controllers:

- Flow management of the switches that belong to their respective zones.

- Computation, selection, and installation of the routes for the relevant flows.

- Network topology discovery and maintenance with coordination of the root controller.

- Handling host and switch management issues such as path failover, traffic engineering, quality of service, host specific updates etc.

Following are the modules that define these tasks of the zone controllers:

Path computation module: In SDN design philosophy, the complexity of the forwarding devices is significantly reduced by shifting the decision-making capability to the controller. The main objective behind this structural change is to let the control logic and forwarding logic evolve and innovate separately. Therefore, path computation and selection are performed by the controller, and the routes are installed in the switches either proactively or reactively by employing OpenFlow or any other similar protocol.

Events processing module: Events triggering is a very frequent activity in networks. The events processing module is implemented to handle the events generated by either users or other network components. It also coordinates with the root controller to notify the events in respective zones.

Application module: Users and network administrators implement network logic in the form of applications and submit to the controller for eventually deploying the sought functionality in the

37

forwarding devices. The application module exposes sufficient set of APIs to let the intended functionality gets quickly deployed without any hassle.

Communication module: One of our design objectives is to provide the required information to various components of the network without incurring much overheads and delay. To this end, the communication module is defined to manage the peer-to-peer interaction among zone controllers, as well as communications with the root controller at the top layer and switches at the bottom layer (via southbound channels).

Failover module: In case of any zone controller failure, the affected switches are either assigned to the other available controllers or a new controller is deployed in case of the insufficient capability of the running controllers. The recovery mechanism tries to stabilize the network without much loss, and the steady degradation strategy is applied in worst case.

Robustness: As explained earlier, the provision fo alternative connections is provided at the multiple layers of the proposed design. Zone controllers failure is handled by the logically distributed control plane. Similarly, the root controller can be replicated to a backup controller much easily due to its physical centralization. Furthermore, the timeout or heartbeat strategy can be used to decide any component failure, and the preemption can be enforced to migrate the affected devices back to their original source. Therefore, the proposed model provides sufficient opportunities to get the desired level of robustness against failures.

4.4 Evaluation Platform

To validate the potential of our proposed model, we use ONOS, FlowVisor, Open vSwitch, and Mininet to build an integrated emulation platform. By default, the global network topology state is cached in memory of each instance of ONOS. And, the joining and leaving nodes are managed by the cluster membership management, that is implemented using Hazelcast's distributed structure.

However, these tasks do not conform with our proposal. Therefore, we accordingly customized ONOS and FlowVisor for global network view and nodes management.

4.5 Results

We used Intel (R) Xeon (R) CPU E5-2603, running at 1.60 GHz, with six cores, 64 GB RAM, 2 TB hard disk, and Ubuntu 14.04.1 LTS 64-bit operating system. Cbench and iperf tools are used for benchmarking the results. Each experiment is averaged by 20 runs. The number of hosts and switches ranges from 20 to 120 per zone for all experiments.

For the comparative analysis, we benchmarked our model with Orion. The topology configuration parameters are equally chosen for accordance of both models. In the first experiment, flow setup rates of both models are compared. Figure 4.2 shows the relative results of the time required by both models. As shown in the diagram, the flow setup time is lower in our model mainly due to centralized control at the root controller. With only one zone, the controller of our model can handle 17279 new flows per second, while the Orion area controller handles 8114 flows per second. Furthermore, the rate of flow setup growth is stable while increasing the number of zone controllers.

The second experiment compares the delay time incurred in Orion and our model. Again, the testing parameters are same for comparing the two models. From Figure 4.3 we can see that our model experiences less delay as compared to Orion. The delay time of Orion is 14 ms in 5 number of areas and 20 switches, whereas our model has 12.3 ms delay with an equal number of zones and switches. The major factor for the lower average delay time of our model is the communication among zone controllers to share the topology state and other relevant information. An improvement of few milliseconds is crucial for the time-sensitive services and applications.
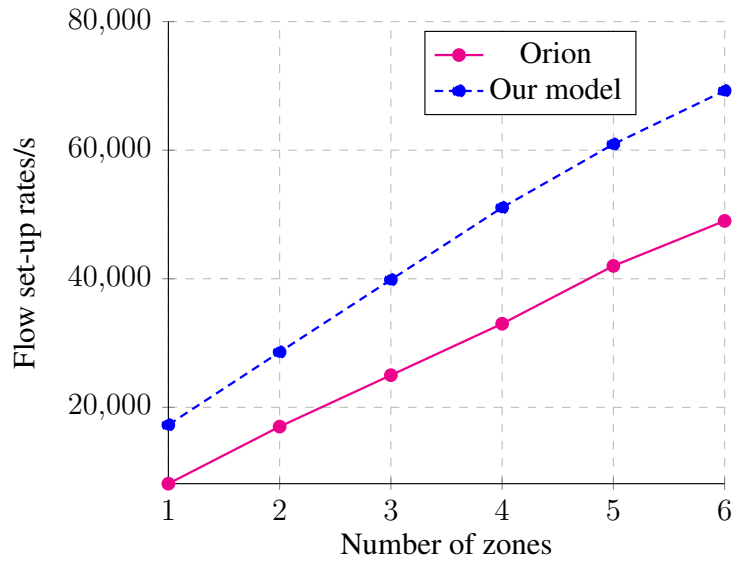
Figure 4.2: Comparison of the flow setup rate
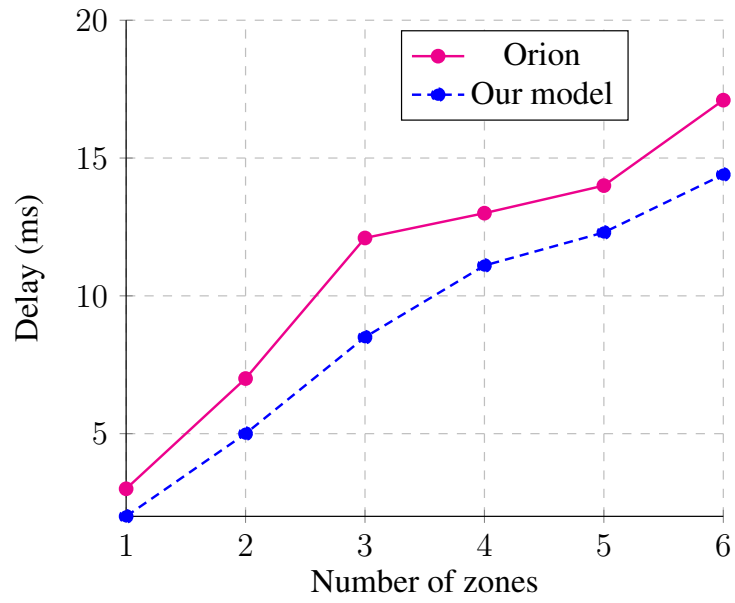
Figure 4.3: Comparison of the average delay time

To measure scalability, we performed experiments to evaluate the effect of increasing the number of zone controllers deployment. Figure 4.4 shows the time required to setup and tear down a topology with respect to zone controllers. We observe that the time grows relatively steadily given that the multi-fold addition of switches and hosts for each zone controller (120 switches, hosts per controller). It shows the large-scale capability of the proposed model.
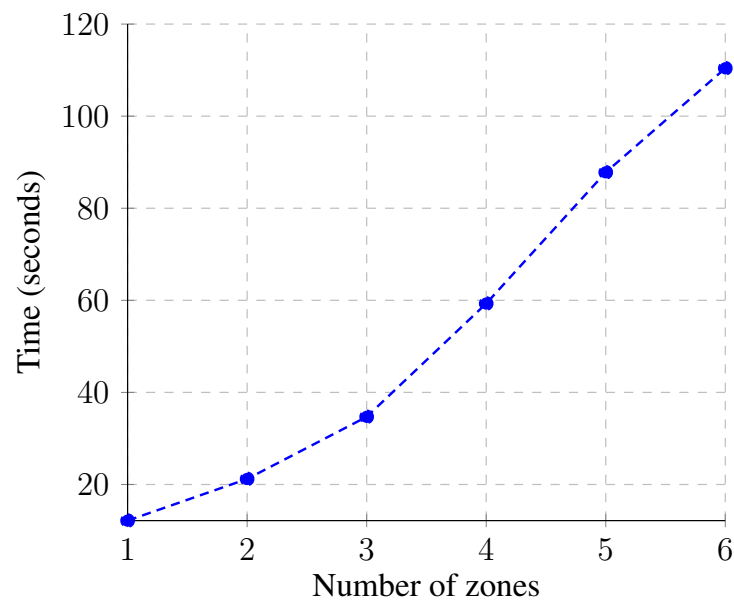
Figure 4.4: Topology setup and tear down time

We measured CPU utilization to assess the overheads incurred by our proposed model. As in previous experiments, the number of switches and hosts were varied to get the relative performance metric of the architecture. Figure 4.5 shows the graph of CPU consumption. It indicates that the rate of growth is fairly low corresponding to an increment in the number of zone controllers. Therefore, we can infer that the control plane carries out intended operations while minimizing the corresponding overheads.

41
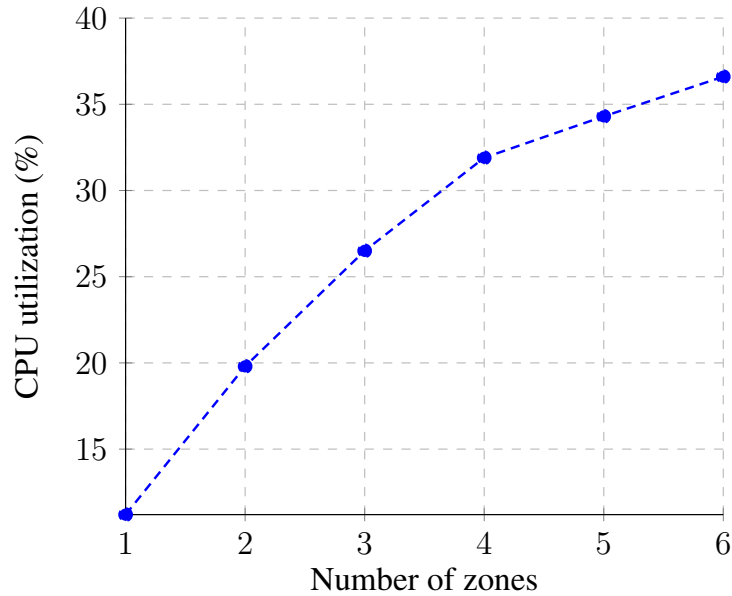
Figure 4.5: CPU utilization

4.6 Conclusion and Future Work

In this chapter, we propose a logically centralized but physically distributed control plane architecture for SDN, that aims to be highly scalable as well performance and robustness intensive while minimizing the overheads. Our model eliminates the need of the distributed data store, distributed protocols or any similar mechanism to maintain coherent network view. And, the modular approach with the layered architecture makes it suitable to deploy in the data center and enterprise networks. Comparative results are also presented to demonstrate the potential usefulness of the model. Components migration algorithms and events registration techniques are the future scope of the work.

42

CHAPTER 5: CLOUD-BASED SOFTWARE DEFINED VIRTUALIZED WIRELESS MOBILE ACCESS NETWORKS

Mainly due to exponential growth in smart devices based mobile computing, the access networks are gaining tremendous momentum. Software-defined networking (SDN), along with cloud computing and virtualization techniques, is considered as a major step forward from the conventional networking. Although SDN is being widely deployed in the data centers and enterprise networks, its adaptation in wireless mobile networks is still in an infancy stage. Unreliable channel and intermittent network connectivity limit the scope of SDN in the wireless context. However, by dealing with these issues, the benefits of the centralized control philosophy of SDN can be reaped for optimum spectrum sharing, QoS support and other services. In this chapter, we propose SoftAccess, a cloud-based architecture for mobile wireless access networks that follows SDN principles and implements virtualization techniques. Seamless network connectivity and mobility management are the crucial aspects of wireless access networks. The proposed model addresses these challenges while making sure to achieve optimum performance and robustness against failures by harnessing capabilities of SDN and cloud computing. We deployed a testbed to evaluate the proposed architecture. The comparative experimental results are presented to corroborate the effectiveness of the proposal.

5.1 Introduction

The smart devices and Internet of Things (IoT) based computing is creating an unprecedented amount of traffic in the conventional networks. Multi-billion devices are projected to get connected to the Internet in upcoming years, that opens the door for a web of mobile devices and corresponding apps with multitude of services having diverse set of quality parameters, network

43

capacity and other disruptive requirements. The hand-held and wearable devices have given enormous computing capabilities to its users. However, the network infrastructure is not being able to grow with similar innovations and advancements. Therefore, the research community is rethinking existing network architectures and structures from the ground-up. Software-defined networking (SDN) is one such paradigm recently emerged from these efforts [31], that is being considered as a breakthrough in recent networking technologies. SDN is being complemented with cloud computing and network virtualization techniques to fully exploit its capabilities for handling existing issues in the traditional networks. The tightly coupled vertical integration of network components inhibits the rapid network growth in the real-time scenarios and adds more complexity to already burdened resources. It further complicates the orchestration and management of the networks that results in an increase of capital and operational expenditures.

SDN aims to simplify network management by decoupling the decision-making control functionality from the forwarding devices. Its core principle is the separation of the control plane from the data plane so that all components may evolve independently in order to provides opportunities for network innovation.
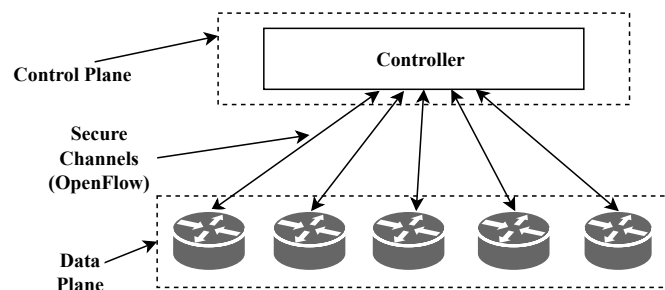
Figure 5.1: A typical SDN architecture.

As shown in Figure 5.1, the data plane is relieved from the control functionality that alleviates complexities of the switching devices. The decision-making capabilities are delegated to the cen-

tralized entity (either physically or logically centralized) that forms the control plane. The forwarding devices can be controlled and programmed from the control plane via secure communication channels using OpenFlow [3] or similar vendor agnostic APIs. It allows network operators and programmers to deploy their specific functionalities on the fly onto the otherwise dumb devices by implementing network apps as per requirements in hand. The users control the forwarding devices by pushing their requirements to the controller, that is responsible for enforcing the network policies accordingly.

By integrating SDN with other promising technologies such as network virtualization and cloud computing, the longstanding issues of the conventional networking can be tackled. However, SDN is mainly embraced for wired networks such as data centers, enterprise networks [44], and its applicability in wireless networks is still very limited. The unreliable communication channels and frequent connection loss due to unreachability or interference hinders secure and dependable communication among controllers and forwarding devices. These issues may lead to complete network breakdown, since the data plane is formed by the dumb devices that can not perform any operation without instructions from the control plane. Furthermore, the mobility management is a complex task that may exhaust the centralized control plane, if not handled carefully. Therefore, the unique characteristics of wireless access networks such as mobility, wireless medium, power conservation, do not allow direct applicability of SDN in these environments.

In this chapter, we propose SoftAccess, an architecture that incorporates SDN principles in the wireless access networks by utilizing capabilities of cloud computing and virtualization techniques. SoftAccess addresses the inherent challenges of wireless networks while providing SDN benefits in a systematic manner. Following are the main contributions of the chapter:

- We discuss the applicability of SDN principles in mobile wireless context.

- An architecture is proposed that allows SDN to be deployed in wireless scenarios while

45

making sure to address inherent challenges of such networks.

- Comparative results are presented to demonstrate effectiveness of the proposal. A testbed is deployed to perform the experiments.

- Some practically feasible applications of the scheme are also discussed.

Rest of the chapter is organized as follows. Related work is discussed in Section 5.2. Section 5.3 describes the architecture of SoftAccess. Then, evaluation platform and experimental results are presented in Section 5.4. Some practical applications of the proposed architecture are discussed in Section 5.5. Finally, Section 5.6 includes the conclusion and future scope of the work.

5.2 Related Work

Our work is closely related to [60] and [61]. SDN-based mobile cloud architecture for MANET environments is presented in [60]. It provides a framework for the components required by SDN in ad-hoc scenarios, and also discusses various use cases and services of the proposal. Our approach is applicable to the path selection, multipath transmission and other use cases mentioned in [60]. However, the wireless interfaces on nodes are statically preconfigured to the specific frequencies in [60]. On the contrary, SoftAccess can dynamically adapt to multiple frequencies via the control messages of the controller that employs virtualization techniques to do this job. An architecture for next generation cellular networks is devised in [61], that integrates SDN and network function virtualization (NFV) techniques to support distributed content routing, heterogeneous networks, and other techniques that strive to address shortcomings of existing LTE networks.

HetNet Cloud is an SDN-based cloud architecture for the implementation of core and access virtual heterogeneous wireless networks [62]. It allows the network operators to build their own

46

network by programming the leased network resources from the cloud, and use the spare bits of the OpenFlow packet model to identify virtual network entities. However, nodes mobility and connection loss are not considered by HetNet Cloud. A software-defined hyper-cellular architecture is proposed in [63], that integrates cloud-based radio access networks (RANs) and software-defined RANs to enable green and elastic wireless access. To manage user mobility, it deploys centralized base station as a controller. However, due to distributed control nodes, the state synchronization is a major challenge in this approach. The elasticity provisions of cloud-based evolved packet core (EPC) for virtualized 5G networks are presented in [64]. It introduces state sharing and synchronization mechanisms to achieve scalability and fault tolerance. The recent state of the art approaches for SDN in wireless networking are outlined in [65].

SoftRAN is an SDN-based control plane architecture for the radio access networks (RANs) [16]. It abstracts base stations in a local geographical area as a virtual base station that can perform load balancing and interference management. A mobile extension of SDN, called meSDN, is proposed in [66], that enables WLAN virtualization, QoS, and power efficiency improvements on mobile devices. Similar to our scenario, the meSDN framework also considers the global controller and local controller provisions for the control plane; however, it does not consolidate the cloud computing techniques that provide optimum resources utilization and user-centric service models. CloudMAC is an SDN-based architecture for enterprise WLANs that processes MAC frames on virtual access points hosted in a data center [67]. An SDN-based spectrum management architecture with a baseband virtualization for wireless networks is proposed in [68]. It shows the benefits for SDN principles in wireless spectrum sharing and other services.

Most of the existing approaches either fall short of exhaustively considering unique traits of wireless access networks or incur formidable cost and overheads for deployment in real life cases. However, contrary to the state of the art solutions, SoftAccess successfully accomplishes its design goals by effectively utilizing SDN and other technologies.

47

Figure 5.2 shows the summarized schematic diagram of so far existing approaches for SDN-based wireless mobile networks.
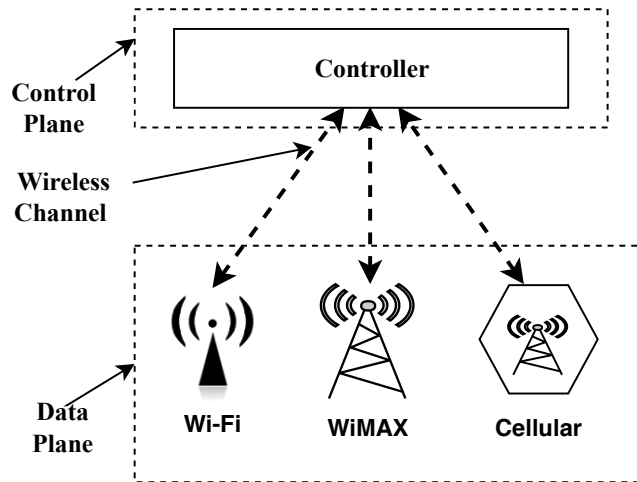
Figure 5.2: SDN in wireless scenario.

5.3 System Model

SoftAccess is an extension of our work in Chapter 4, that addresses scalability of SDN in data centers by having physically distributed but logically centralized control plane since physical centralization of control plane architecture does not scale well. However, mobility management, spectrum sharing, unreliable connectivity and other characteristics of wireless mobile networks were not considered in the earlier work. Also, virtualization and clouds techniques were not incorporated.

Following are the additional challenges of wireless access networks:

- Given the constant variations in link/channel conditions and significant computation of centralized route discovery, SDN principles can not be extended to wireless networks in a

48

straightforward manner.

- Smooth handover process of mobile devices among multiple domains is an arduous task.

- Computational intensive mobility management can throttle the control plane.

- Additional provisions are required to handle sparsity or density of continuously changing network topologies.

- Non-uniform network traffic of multiple co-existing radio access technologies pave the way for further complications.

- Finite energy and other resources scarcity with very stringent performance and latency needs are some of the other problems associated with wireless access networks.

SoftAccess is designed to uniquely address these issues by harnessing capabilities of cloud computing, virtualization, and other techniques. Figure 5.3 shows the architecture of SoftAccess. As shown in the diagram, it has a logically centralized but physically distributed control plane. However, due to a physically centralized global controller, it also possesses traits of physical centralization. The global centralized controller is placed in the clouds and the local controllers are deployed in the vicinity of the hardware infrastructure. The physical layer is formed by aggregating forwarding devices by virtualization techniques, i.e., the physical infrastructure is abstracted from the control plane. A varying number of switches represents the provision to aggregate any number of resources as per specific requirements. Virtualization can also be defined for the purpose of evenly distributed load management and optimum resources utilization of a large enterprise. The reliable TCP connections are employed for peer-to-peer communication among local controllers as well as interaction with the cloud-based global controller at the top layer.

As elaborated below in details, the global controller is mainly responsible for managing the global network view and some other network-wide functionalities, while the local controllers control

the underlying physical devices (i.e forwarding plane) via OpenFlow protocol. Furthermore, the communication channels are established between local controllers for peer-to-peer communication. These channels are used to share any kind of information among local controllers that is not available to a particular controller. Fetching the required information from neighboring controllers contributes to minimizing the latency that can be higher if the information is requested from the global controller at the top layer. Also, due to localized information sharing, the peer-to-peer interaction reduces the communication overheads in the network.

Figure 5.3: Hierarchical SDN architecture.

Following are the key characteristics of SoftAccess:

Cloud-based global controller: Clouds provide an abundance of processing, storage, and other resources on demand that allow computational and data intensive tasks to be executed on less capable mobile devices. It also gives the desired level of performance, robustness and QoS support

50

at very flexible cost models.

Seamless connectivity and mobility: In case of unavailability of global controller due to some reasons, the local controllers continue to serve the forwarding devices and keep the network alive. The peer-to-peer interaction among local controllers facilitates to keep a continuity of on-going communication sessions with vertical handovers or multiple stations.

Virtualization: It is a well known fact that integration and coordination of heterogeneous nature of technologies is not an easy task. To this end, virtualization of physical resources provides opportunities to configure multiple wireless interfaces that can operate at separate frequencies.

The controller deployment in clouds promise lower operational costs for using real-time computing resources, since resources can be added or removed on-demand and in very simplified manner. As a result, the network operators can expand their infrastructure as they need it, rather than provisioning in advance. Given that the control function of SoftAccess can be placed in a cloud as modules, it overhauls the conventional networks into a stateless network that results in control-oblivious data plane.

The inherent challenges of wireless networks may result in a significant bottleneck for physically centralized control plane. Therefore, local controllers are deployed to manage relatively localized events and keep the network stable in the case of unreachability or other issues of the global controller.

Following are the additional reasons of local controllers deployment close to the edge of the network:

- Due to closeness with the hardware infrastructure, the local controller has a more updated view than the global controller; hence it can better manage its localized resources.

51

- The mobility management is re-factored to the local controllers mainly due to their vicinity with the corresponding devices. Also, having multiple local controller nodes in place simplifies computation of otherwise complicated mobility management.

- SoftAccess allows mobile users to dynamically select the best possible available network and efficiently utilize coordinated and simultaneous use of multiple radio resources or interfaces on the devices equipped with multi-homed, multi-path capabilities. The local controllers implement this functionality by having direct control over the virtualized layer of physical resources.

- The local controllers possess uplink and downlink resources management provisions that allow access points to coordinate bi-directional transmissions with the shared-medium wireless MAC protocols.

Various tasks of the control plane of SoftAccess can be configured with programmable modules that allow flexibility, and simplifies troubleshooting and maintenance by tracking, isolating, and fixing any issue in a systematic way without perturbing the other independent components. We developed the following modules for SoftAccess:

Mobility manager: Mobility is characterized by many factors including position, direction, movement pattern, acceleration, density, and duration of the communication. The mobility manager consider these factors to keep track of the incoming and outgoing devices in its region. To achieve scalability and reliability, our approach follows the distributed mobility management (DMM) paradigm where local controllers coordinate among neighboring nodes to ensure seamless mobility and handover process. Also, having multiple local controller nodes in place simplifies the computation of otherwise complicated mobility management. The local controllers act as the data aggregators and provide address prefixes to the global controller which configures the reachable interfaces accordingly. The global controller also keeps track of the already registered devices in a region.

52

Furthermore, the local controllers provide functionality to dynamically change the anchor points for moving devices that alleviate signaling overheads during service sessions. Additionally, the local controllers calculate the flow matching rules, corresponding actions, and the path modifications to forward the data towards destination. OpenFlow protocol is employed for these configurations.

Spectrum manager: It is one of the most important components of SoftAccess. It provides fine-grained channel bandwidth management according to the type of services and applications. Given the heterogeneous requirements of a diverse range of devices, it is essential to fulfilling their transmission rate, delay, and other demands accordingly.

Statistics manager: The desired level of consistency in the network state is dependent on the real time topology changes and other events in the network. To achieve this objective, the local controllers collect statistics from their respective devices and submit reports to the global controller. Subsequently, the global controller uses this information to maintain the global network state and topology.

Storage manager: The network state information is stored and handled by the storage manager. The physical centralization of the network topology information alleviates the overheads of the distributed data stores or shared file systems and simplifies high consistency of network management from a vantage point.

Flow manager: As specified by the OpenFlow protocol, the flow manager uses wildcards and filtering mechanisms to calculate optimum paths for the flows. The local controllers are responsible to immediately handle the flow requests of devices and provide real-time updates to the information base of the global controller. The flow manager uses the statistics manager to measure various metrics for the privileged traffic steering and meet end users requirements.

Load manager: The load manager supervises dynamic provisioning of the network resources by

getting the real time usage updates from the statistics manager. It implements the load balancing methods that make sure to efficiently utilize the capacity of physical resources.

Events manager: The events manager is implemented to handle the events generated by either users or other network components. It also coordinates with the global controller to notify the events.

Application manager: There is a proliferation of applications for communication networks. Users and network administrators implement network logic in the form of applications and submit to the controller for eventually deploying the sought functionality in the forwarding devices. The application manager exposes sufficient set of APIs to let the intended functionality gets quickly deployed.

Communication manager: To mitigate latency and other overheads, the communication manager is defined to manage the peer-to-peer interaction among local controllers, as well as communications with the global controller at the top layer and switches at the bottom layer (via southbound channels).

Failover manager: Failure of even a single component can have a cascade effect on the entire network that can cause severe damages to services and resources. Therefore, the failover manager is designed to handle failures of a single or multiple components. In a case of any local controller failure, the affected switches are either assigned to the other nearby available controllers or a new controller is deployed by the global controller in case of the insufficient capability of the running local controllers. The global controller has a backup and restore provision in clouds to handle unexpected events.

5.4 Evaluation Platform and Results

To validate the potential of SoftAccess, we use ONOS, FlowVisor, Open vSwitch, and Mininet-Wifi to build an integrated testbed that is extended from our earlier emulation platform developed for evaluation of smart cloud applications [69]. The available open source components are customized and extended with required modules to setup the testbed.

We used physically separate machines with the configuration of Intel (R) Xeon (R) CPU E5-2603, running at 1.60 GHz, with six cores, 64 GB RAM, 2 TB hard disk, and Ubuntu 14.04.1 LTS 64-bit operating system. Multiple instances of ONOS are deployed on one machine by having separate virtual machines (VMs) for each instance, using the Oracle Virtual Box. Cbench, iperf and other network tracing utilities are used for benchmarking the results. Each experiment is averaged by 20 runs. The number of hosts and switches are varied at runtime for all experiments.

For the comparative analysis, we benchmarked SoftAccess with existing works. The topology configuration parameters are equally chosen for the sake of conformity other models. In the first experiment, packet delivery ratio with respect to mobility of 50 nodes is compared with [60]. Figure 5.4 shows the relative results of the packet delivery ratio of both models. It is evident from the diagram that SoftAccess achieves better packet delivery ratio even though it drops as the mobility increases due to routes unavailability and failures with higher speeds. Localized information sharing and efficient state management contributes towards packet delivery ratio improvement.

The control traffic comparison is shown in Figure 5.5. SoftAccess generates fewer control beacons due to peer-to-peer communication among the local controllers that avoids global controller involvement for flow setup and other control tasks. Therefore, it incurs less overheads that results in network scalability and performance improvements. It is worthwhile to notice that the control traffic grows as the mobility increases, due to more frequent positioning parameter changes that

55

require more control messages.
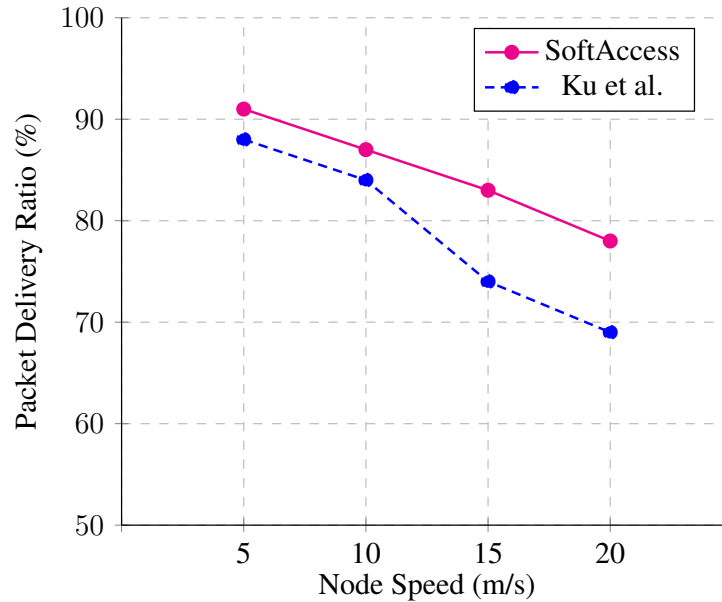
Figure 5.4: Comparison of the packet delivery ratio

Next, the controller failure comparison is shown in Figure 5.6. Evidently, [60] does not handle the situation of controller failure in their approach, therefore, the packet delivery ratio drops abruptly at the time of controller failure. On the other hand, SoftAccess has the provision to dynamically adapt load distribution among other controllers in case of failure. Therefore, the packet delivery ratio again reaches a stable state after an interruption of short time period. However, this recovery time may be higher for the stringent requirements of delay sensitive access networks. Reduction in this failover time is one of our future works.

Then we performed throughput comparison with [68]. Similar to above results, topology configuration and other parameters were equally chosen for the experiment. Figure 5.7 shows that SoftAccess achieve more throughput with respect to the number of links. And, compared to [68],

56
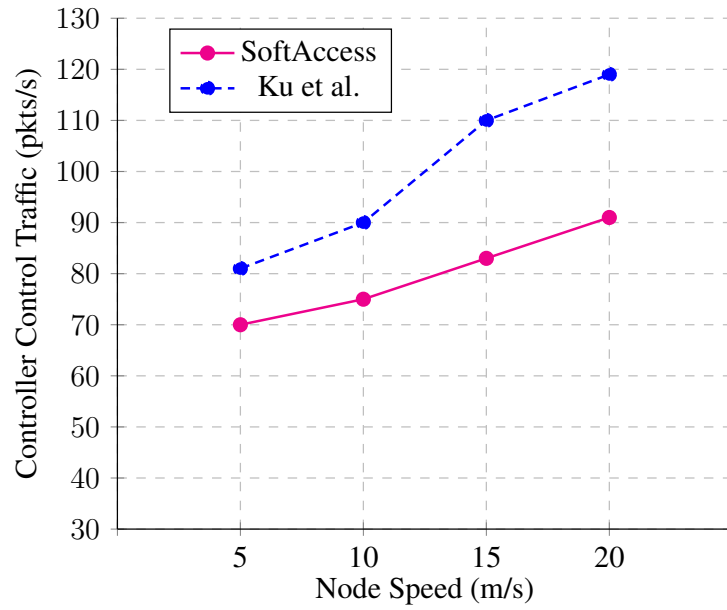
Figure 5.5: Control traffic comparison
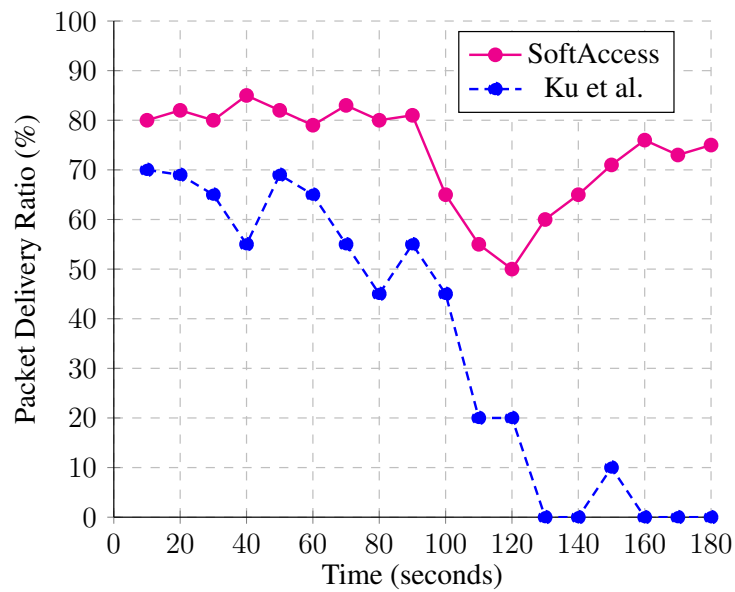
Figure 5.6: Controller failure comparison

57

SoftAccess throughput is stable with nodes mobility, that further underlines the effectiveness of our model. Throughput improvements are attributed to the less control traffic overheads and information localization.
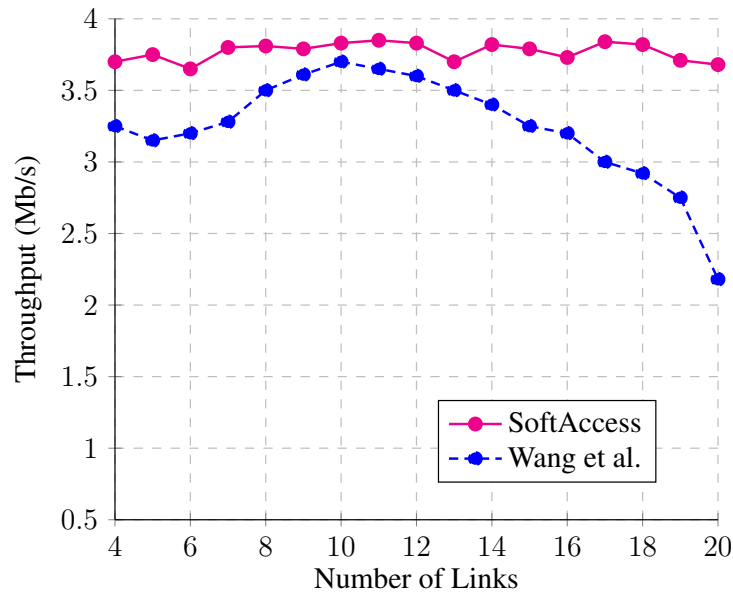
Figure 5.7: Throughput comparison

Finally, we measured the effects of nodes mobility on delay time. We can observe from Figure 5.8 that latency is increased with devices mobility. It leads us to infer that the mobility management is a crucial aspect for the delay sensitive, real-time services and applications.

58
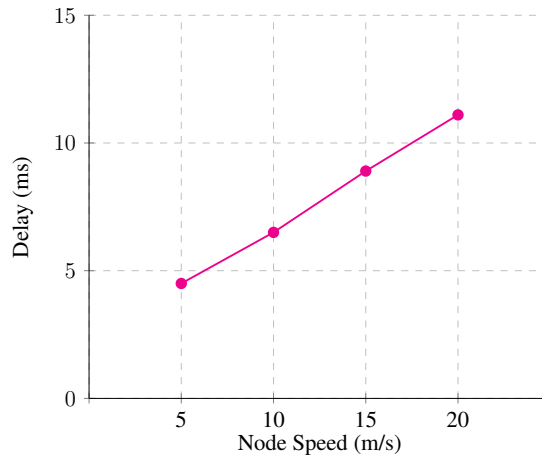
Figure 5.8: The delay time measurement

5.5 Practical Applications

Following are few of the potential use cases of SoftAccess:

Cellular networks: By the virtue of its design, SoftAccess is a suitable consideration for the radio access networks (RANs) and other telecommunication technologies.

Ad hoc networks: The mobile ad hoc networks (MANETs), vehicular ad hoc networks (VANETs), and sensor networks have very peculiar characteristics such as unpredictable mobility, power constraints and unreliable connectivity. Therefore, the cloud-based architecture of SoftAccess makes it a very plausible option in such scenarios that addresses mobility, resources limitations, and other issues.

Campus networks: OpenFlow-based wireless networks are already being deployed in campus networks [70]. SoftAccess complements the existing solutions by supporting additional features.

59

Enterprise WLANs: Optimum resources utilization, seamless connectivity and guaranteed QoS support are the major goals of the enterprise WLANs. The global network view of SoftAccess provides very flexible resources allocation and load balancing provisions.

5.6 Conclusion and Future Work

Largely due to substantially growing proliferation of smart mobile devices, cloud-based applications and services, the existing wireless access networks are unable to bear pressure on the network resources that can quickly get strained with very strict capacity, performance and latency requirements. In this chapter, we propose SoftAccess, a logically centralized but physically distributed architecture for wireless mobile access networks based on SDN principles, that aims to be highly scalable as well performance and robustness intensive while obviating the corresponding challenges. Along with SDN concepts, we harnessed cloud computing and virtualization techniques to meet the goals of mobile access networks. Our model eliminates the need of the distributed data store, distributed protocols or any similar mechanism to maintain coherent network view. Comparative results and practical applications are presented to demonstrate the potential usefulness of SoftAccess. Other than failover improvements, interference mitigation techniques and mobility management optimization are the future scope of the work.

CHAPTER 6: SDN-BASED MOBILITY MANAGEMENT AND QOS SUPPORT FOR VEHICULAR AD-HOC NETWORKS

Along with non-safety related applications, traffic safety is the major concern of the Vehicular Ad-hoc Networks (VANETs). However, the mobility management due to the high speed of vehicles, intermittent connectivity, and frequent topology variations are some of the crucial roadblocks. These challenges impose setback for quality of service (QoS) guarantee that leads to unfulfilled goals of VANETs deployment. The centralized control of the Software-Defined Networking (SDN) paradigm allows optimum utilization of global network view to meet the QoS requirements. Furthermore, by a systematic design of the SDN control plane, the issues of mobility management and poor network connectivity can also be addressed in an efficient manner. In this chapter, we propose an SDN-based architecture that utilizes cloud computing and deals with inherent constraints of VANETs. A logically distributed control plane is devised for seamless connectivity, mobility management, and QoS support. The proposed model achieves optimum performance and robustness against failures by harnessing capabilities of SDN and cloud computing. We implemented the QoS and routing applications to evaluate the proposed model. The comparative experimental results are presented to demonstrate the effectiveness of the proposed framework.

6.1 Introduction

The major goals of VANETs deployment are safety, traffic management, and infotainment. Real-time safety critical messages are disseminated to avoid imminent collisions or any other hazardous situations. Driver assistance with optimum route selection, toll avoidance (if preferred), congestion notifications with respect to travel time and fuel consumption contributes towards efficient traffic management. Due to the environment it operates in, VANET faces many crucial challenges. High

61

mobility of vehicles makes rapid changes in the dynamic network topology that pose significant challenges for connection establishment and on-going communication sessions [71]. Similarly, varying density of nodes with unforeseen circumstances further complicates topological instability of the network. However, the objectives of VANETs, particularly safety-related, strictly demands some baseline assurance to its applications in terms of QoS and other performance metrics [72]. Therefore, time sensitivity and the QoS are essential aspects of VANETs realization in practice. VANETs comprise of road-side units (RSUs) and in-vehicle on-board units (OBUs) devices that implement the protocols and contain the equipment for communication. Vehicle-to-vehicle (V-to-V) communication is defined as the ad-hoc mode where vehicles interact with each other and share useful information without the supervision of any coordinating authority. On the other hand, vehicle-to-infrastructure (V-to-I) and vice versa is the infrastructure mode, where communication is handled by the permanently fixed or temporarily deployed RSUs. A typical vehicular networks scenario is depicted in Figure 6.1.
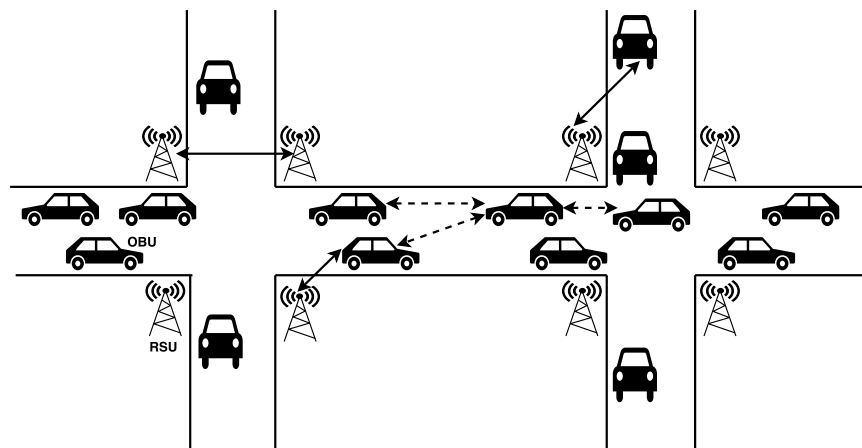
Figure 6.1: A typical VANET scenario.

Software Defined Networking (SDN) paradigm relies on the separation of the decision-making capability and the forwarding functionality. The control plane is defined as the centralized entity that possesses all decision-making proficiency and the data plane consists of dumb forwarding devices

that are controlled and managed by the controller via an open interface (e.g. OpenFlow). The global network view and real-time statistics updates contribute to exploit consistent network state for simplified administration and programmability. However, physical centralization of control plane has inherent limitations, some of which are highlighted in Chapter 4, along with a practically feasible solution.

In this chapter, we present design, implementation and evaluation of an SDN-based hierarchical architecture that incorporates cloud computing to deal with pertinent issues of VANETs. Rest of the chapter is organized as follows. Related work is discussed in Section 6.2. Section 6.3 presents the architecture of the proposed model. The two applications for the control plane are described in Section 6.4. Then, evaluation platform and comparative results for the proposed mechanism are elaborated in Section 6.5. Some potential benefits of the proposed architecture are discussed in Section 6.6. Finally, Section 6.7 includes the conclusion and future work.

6.2 Related Work

The SDN-based architecture for vehicular networks is presented in [73]. Its feasibility is demonstrated by comparing SDN-based routing with legacy routing protocols. However, it requires that the traffic from any wireless node needs to run through its own SDN module, the operation that incurs substantial overheads in ad hoc environments. In our approach, all traffic need not travel through every substrate en-route to the destination. RSU cloud [74] is a vehicular cloud to implement the Internet of Vehicles that benefits from the flexibility and programmability offered in SDN. However, given that the RSU cloud comprises of roadside RSUs, it has a limited applicability in V-to-V communication and the network may get disconnected if there is no connection between vehicles and RSUs. By using the control and management features of SDN, the application-level QoS metrics for online real-time applications are proposed in [75]. An SDN-based architecture

63

for heterogeneous vehicular communication in [76] shows that the logically centralized control plane provides agile configuration capability for multiple devices and network resources, and the vehicle trajectory predictions can be utilized to reduce the management overheads. An SDN enabled technique for high-performance multicast in vehicular networks is outlined in [77], that uses the network topology information provided by the global network view for making an efficient scheduling decision. This work further underlines the benefits of SDN principles in VANETs scenario. The scalability problem of VANETs is outlined in [78], that also suggests some ways to address it. Due to its hierarchical model and logically centralized control plane, our proposed architecture deals with the scalability problem of VANETs in a better way.

Li et al. [79] present an optimization strategy to make a balance between latency and corresponding cost for an SDN-based vehicular networks. This work shows the possibility of optimum network performance by applying SDN concepts to VANETs. Our previous work addresses the QoS issue by categorizing data into multiple classes and assigning priorities to upload or download requests [80]. GPSR [81] and CLWPR [82] are the notable position-based routing protocols for VANETs. A cloud-based hierarchical architecture for vehicular networks with the goals of efficient resources management and reliability of cloud services for vehicles is presented in [83].

Most of the existing approaches either fall short of considering the SDN and clouds based technologies or incur formidable cost and overheads for deployment in real life cases. The proposed model successfully accomplishes goals of VANETs by effectively utilizing SDN and other techniques.

6.3 System Model

Given its peculiar characteristics, VANETs are categorized as distributed and self-organized networks. On the other hand, SDN paradigm is based on the principle of centralized control. There-

64

fore, consideration of SDN for VANETs is a challenging task. However, there are few aspects of vehicular networks that can be exploited along with SDN principles. For example, vehicles follow a predictable topology by having a GPS service for maps of roads and streets, that allows traffic optimization with global network view and other SDN techniques. Following are additional challenges of considering SDN for VANETs:

- Given the constant variations in link/channel conditions and significant computation of centralized route discovery, SDN principles can not be extended to vehicular networks in a straightforward manner.

- Smooth handover process of vehicles among multiple domains is an arduous task.

- Computational intensive mobility management can throttle the control plane.

- Additional provisions are required to handle sparsity or density of continuously changing network topologies.

- Non-uniform network traffic of multiple co-existing radio access technologies add further complications.

We propose an SDN-based hierarchical architecture to address these issues by incorporating a layered design and harnessing capabilities of cloud computing. Figure 6.2 shows the high-level design of our model, and internal components of the global controller and OBUs (mounted on each vehicle) are depicted in Figures 6.3 and 6.4 respectively. The global centralized controller is placed in the cloud and the local controllers are deployed in OBUs. It can be observed from the diagrams that the control plane is composed of a single global controller in the cloud and a local controller per vehicle (per OBU) that results in a logically centralized but physically distributed control plane. The global controller contains the core modules, which are the building blocks

65

of control functionality and provides the platform to implement and deploy various applications on top of the control plane. Additionally, the global controller maintains a centralized database repository for a consistent view of the topology and other network-wide activities that is utilized by the scheduler to implement QoS related queues and enforce the specific policies. In the V-to-I mode, vehicles interact with the global controller via RSUs. And, RSUs exploit services of the global controller to fulfill requirements of the nodes in their respective coverage regions.
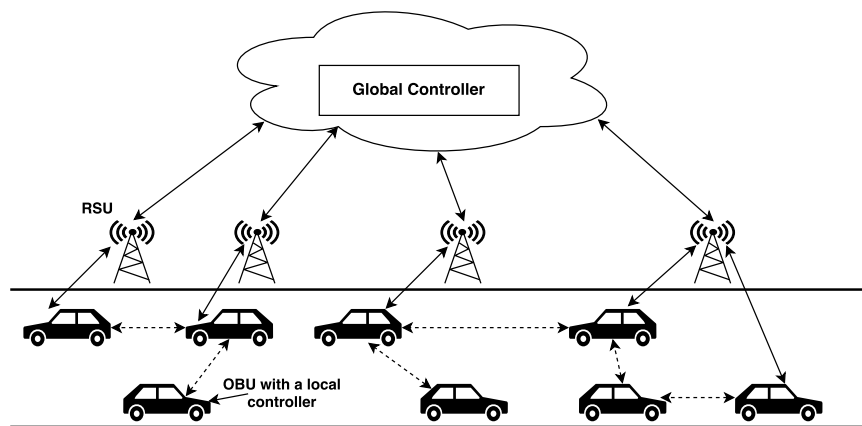
Figure 6.2: SDN-based VANET architecture.
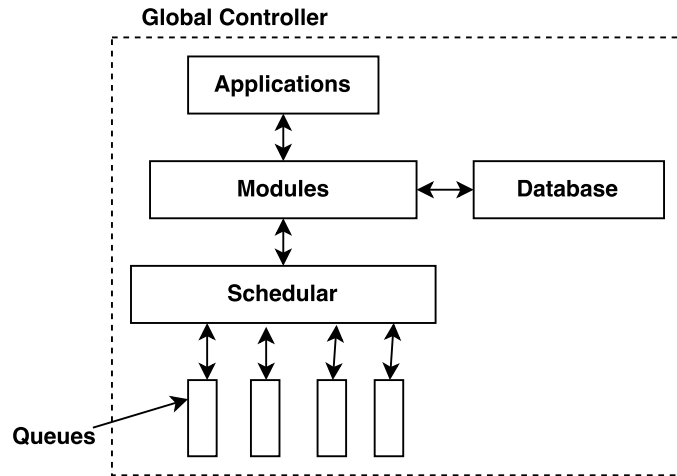
Figure 6.3: Internal components of the global controller.
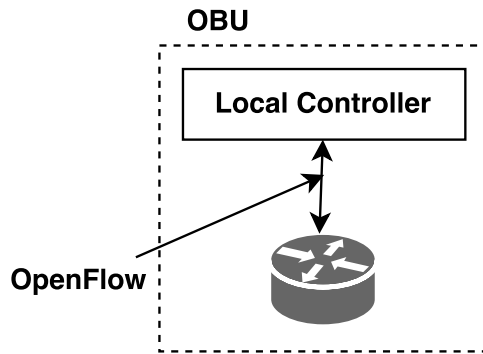
Figure 6.4: Internal components of the local controllers.

The local controllers are deployed in the vicinity of the vehicular nodes for the reasons given below:

- Due to short radio range of vehicles and relatively sparse deployment of RSUs, the communication among vehicles and RSUs is sporadic.

67

- High speed of vehicles adds to frequent disruptions in the network topology. Additionally, the high mobility of vehicles gives very short time duration to share data among other vehicles and the en-route RSUs.

- Sometimes the associated delay of connection establishment with RSU is too much to bear for delivering an emergency message.

- The local controllers abstracts the handover points from the vehicles, and obviate the vehicles to run their own mobility protocols. Consequently, operations at the forwarding plane of vehicles are simplified that contributes to network performance and efficiency improvements. Furthermore, the mobility management is re-factored to the local controllers mainly due to their vicinity with the corresponding devices.

- The peer-to-peer information sharing among local controllers improves multi-hop communication support. Vehicles rely on the local controllers which handle the dynamic selection of V-to-V or V-to-I mode.

- Due to proximity with the hardware infrastructure, the local controller has a more updated view than the global controller; hence it can better manage its localized resources.

- In general, the non-safety related applications such as audio/video streaming, software updates, file sharing, Internet access etc. are resources-intensive; therefore, it makes sense to deploy such applications at the global controller for V-to-I communication. Similarly, safety related real-time applications require immediate attention; hence, should be deployed in the proximity of the consumers for V-to-V communication. With this in mind, we aim to deploy safety applications at local controllers and non-safety applications at the global controller.

The global controller is mainly responsible for managing the global network view and some other network-wide functionalities, while the local controllers control the underlying physical devices

68

(i.e forwarding plane) via OpenFlow protocol. Furthermore, the communication channels are established between local controllers for peer-to-peer communication. These channels are used to share any information that is otherwise unavailable to a particular controller. Fetching the required information from neighboring controllers contributes to minimizing the latency that can be higher if the information is requested from the global controller at the top layer. Also, due to localized information sharing, the peer-to-peer interaction reduces the communication overheads in the entire network. Following are the key characteristics of our model:

Cloud-based global controller: Clouds provide an abundance of processing, storage, and networking resources on demand that allow computational and data intensive tasks to be executed on less capable mobile devices. It also gives the desired level of performance, robustness and QoS support at very flexible cost models. Moreover, the centralization aspect of clouds tends to naturally apply to the globally centralized controller of our model. Additionally, the cloud-based architecture provides mobile users the ubiquitous communication capability and information access regardless of the physical location.

Seamless connectivity and mobility: In case of unavailability of global controller due to the reasons mentioned before, the local controllers continue to serve the vehicles and avoids fragmentation of the network. The peer-to-peer interaction among local controllers facilitates to keep a continuity of on-going communication sessions with vertical handovers, which is a challenging task for the conventional networks, given the diversity of technologies and network characteristics.

6.4 Applications for the Control Plane

SDN-route: As mentioned in [84], the network programmability capability provided by SDN can eliminate triangle routing problem since the binding cache of a mobile node can be placed on the

shortest path and the centralized control of SDN mitigates protocol complexity. Furthermore, this approach implies faster handover and less overhead on wireless links. And, global network visibility alleviates the multi-hop flooding of routing information that leads to network scalability and performance gains. Therefore, by taking cues from RouteFlow [85] and RCP [86], we developed a routing application called SDN-route that leverages advantages of global network view for route optimization. However, opposed to [84], we delegate the binding cache storage capability to the local controllers rather than switches. Conforming to the OpenFlow protocol, SDN-route uses wildcards and filtering mechanisms to calculate optimum paths for the flows. Additionally, the control functionality is removed from the end devices and shifted into a higher control layer, only the final outcome of route processing is distributed to the substrate via OpenFlow. SDN-route uses the statistics manager to measure various metrics for the privileged traffic steering and meet end users requirements.

SDN-QoS: The policy of traffic prioritization by the the IEEE 802.11e standard for QoS is not sufficient, because it provides no way to prioritize traffic of the same access category (AC) and the priority is statically given in a specified manner to predefined classes. On the contrary, an interesting characteristic of SDN is that it allows flow-based QoS control in a fine-granular and flexible way. We exploited this capability to dynamically achieve QoS in VANETs and implemented the QoS mechanism proposed in [80] within this framework.

6.5 Evaluation Platform and Results

To validate the potential of our model, we use ONOS, FlowVisor, Open vSwitch, and Mininet-Wifi to build an integrated testbed that is extended from our earlier developed emulation platform for evaluation of smart cloud applications in Chapter 3. The available open source components are customized and extended with required modules to setup the testbed. The global controller, local

controllers, and the forwarding plane are deployed on physically separate machines to achieve realism and performance of the platform. The ONOS instances are deployed as the local controllers that communicate with each other to share the required information; however, the global topology management is handled by FlowVisor.

We used separate machines with the configuration of Intel (R) Xeon (R) CPU E5-2603, running at 1.60 GHz, with six cores, 64 GB RAM, 2 TB hard disk, and Ubuntu 14.04.1 LTS 64-bit operating system. Cbench, iperf and other network tracing utilities are used for benchmarking the results. Each experiment is averaged by 20 runs. Unless otherwise mentioned, the number of vehicles are randomly varied from 20 to 180 at runtime for all experiments.

For the comparative analysis, we benchmarked our model with existing works. The topology configuration parameters are equally chosen for the sake of conformity with the other models. In the first experiment, packet delivery ratio with respect to average speed of vehicles is compared with GPSR [81] and CLWPR [82]. It is evident from Figure 6.5 that our routing application achieves better packet delivery ratio even though all three models degrade in performance as the mobility increases due to routes unavailability and failures with higher speeds. Localized information sharing and efficient state management contributes towards packet delivery ratio improvement.

71
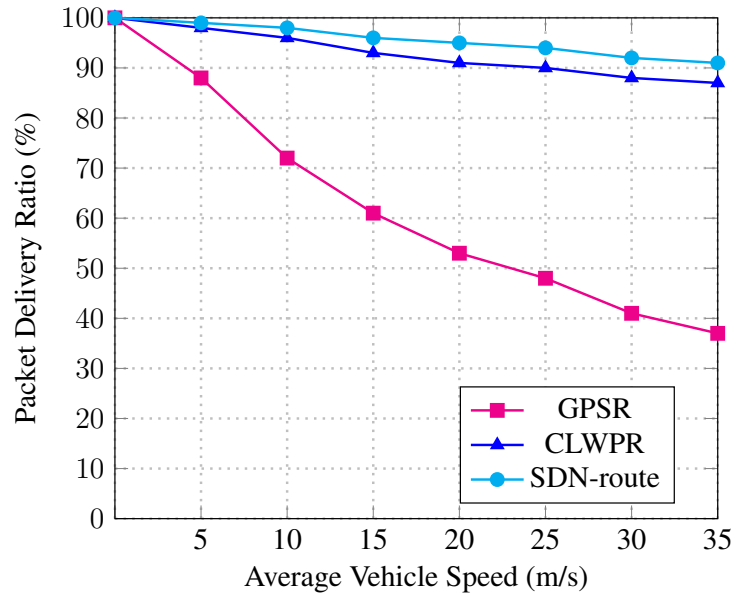
Figure 6.5: Comparison of the packet delivery ratio

Then we compared the end-to-end delay with variation in speed of vehicles. We can observe from Figure 6.6 that SDN-route outperforms GPSR and CLWPR in terms of delay. As mentioned earlier, peer-to-peer coordination among local controllers significantly improves the localized information sharing and reduces the communication overheads that contributes for minimum delay and seamless experience for end users. However, it must be noted that the delay tends to increase with mobility of vehicles. It leads us to infer that the mobility management is a crucial aspect for the delay sensitive, real-time services and applications.
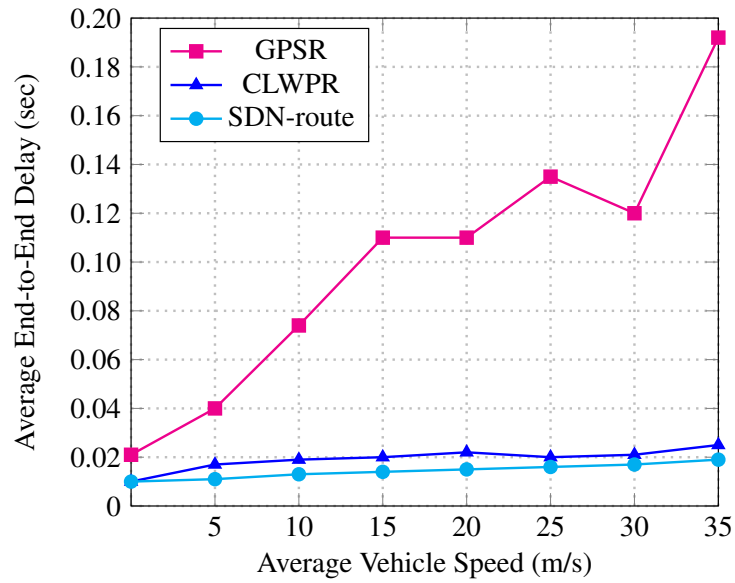
Figure 6.6: Comparison of end-to-end delay
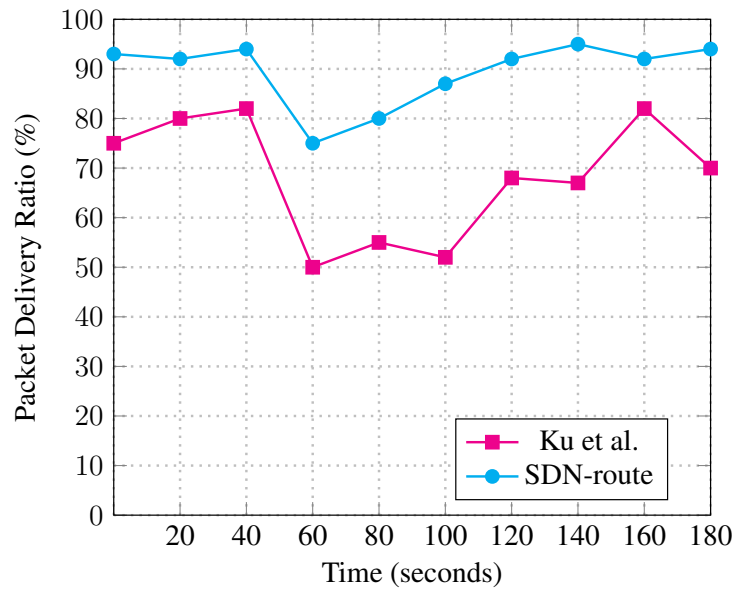
Figure 6.7: Controller failure comparison

Next, to assess the robustness of our model, we compared the controller failure scenario with [73], and results are shown in Figure 6.7. Similar to above results, topology configuration and other parameters were equally chosen for the experiment. We can see that SDN-route recovers after an abrupt failure of a local controller. Our proposal has the provision to dynamically adapt load distribution among other controllers in case of failure. Therefore, the packet delivery ratio again reaches a stable state after an interruption of short time period.

To evaluate the QoS capability of our proposal, we compared it with [80]. Comparison of service ratio is shown in Figure 6.8. We gained a significant improvement in the service ratio with respect to arrival rate of requests that underlines the effectiveness of incorporating SDN principles in conventional VANETs scenarios. The service ratio improvement is attributed to the QoS support achieved by the coherent and global network view provided by SDN control plane.
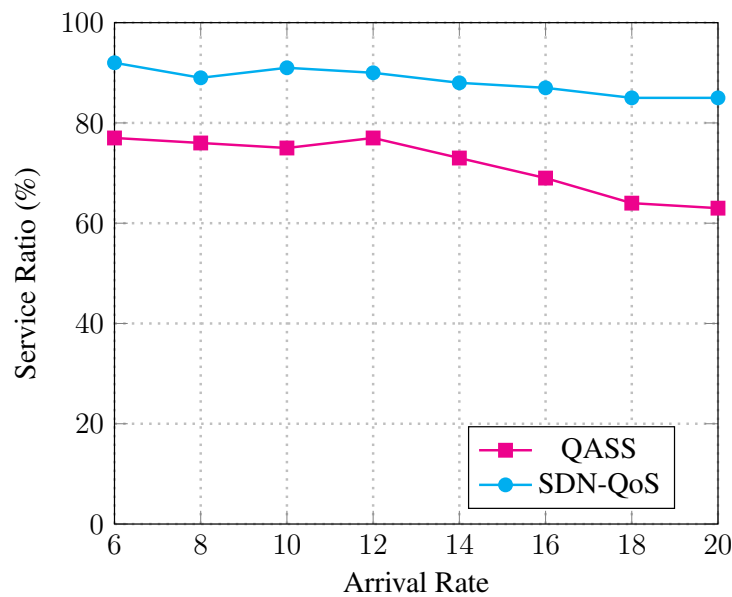
Figure 6.8: Comparison of service ratio

Figure 6.9 shows the impact of vehicles density to handover latency with respect to increasing speed of vehicles. It illustrates that the number of vehicles as well as the increase in speed adversely

74

affect the handover latency. The higher number of vehicles and speed incur more congestion and communication overheads that contributes to increase in handover latency. However, mainly due to consistent network state information and less communication overheads, our model experience acceptable latency.
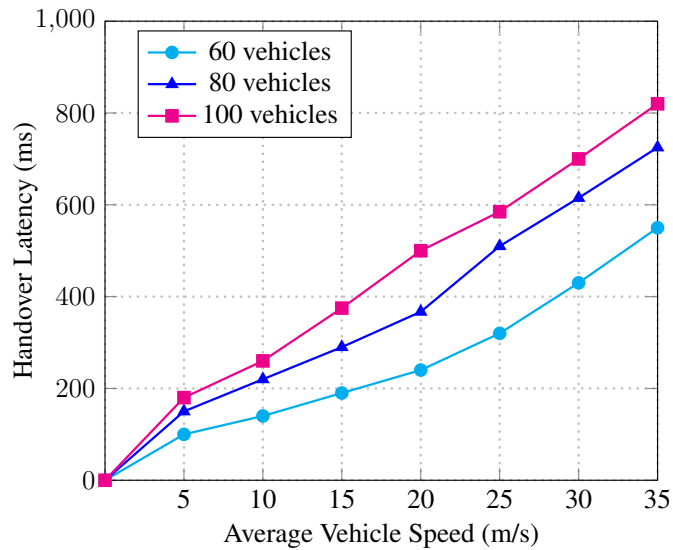
Figure 6.9: Measurement of handover latency

6.6 Benefits

Following are some potential benefits of the proposed model:

- Considering the trend of reorganizing traditional backbone and IP networks with SDN principles, the proposed model pave the way for seamless integration of VANETs with fixed IP networks.

- For the proof-of-concept, we described the architecture for the wireless LAN only; however, as mentioned in [76], it can be deployed with cellular networks and other technologies

75

of heterogeneous nature. Furthermore, the IEEE 802.21 standard for Media Independent Handover (MIH) services can be utilized to enable handover among disparate networks [87].

- As mobile devices are rapidly moving from conventional computing to apps development, our model provides a platform to the network programmers to develop in-vehicle apps that can be deployed via control plane.

- Due to ad hoc nature of VANETs, the IEEE 802.11p MAC protocol does not offer the possibility for the centralized polling-based channel access or the collision-free phase [88]. However, the control plane of our model has knowledge about all communicating nodes, and the controller can act as a coordinator to alleviate constraints of the existing MAC protocol.

6.7 Conclusion

Fast mobility of vehicles, dispersed nodes, intermittent connectivity, real-time communication requirements pose a significant challenge for VANETs to achieve its goals. Due to its intrinsic characteristics, SDN concepts can not be directly applied for wireless networks in general and VANETs in particular. Considering all these factors, we devised a SDN-based architecture for VANETs that systematically addresses challenges of vehicular communications, while eliminates limitations of SDN for such scenarios. By delegating control functionality among a global controller in clouds and local controllers on each OBU, it provides an opportunity for V-to-V, V-to-I, as well as hybrid communication. Our model eliminates the need of the distributed data store, distributed protocols or any similar mechanism to maintain coherent network view. We implement two applications to demonstrate efficient mobility management and QoS support of the proposal. Furthermore, comparative results and potential benefits are presented to validate usefulness of our model. Multihoming to simultaneously connect vehicles with multiple access networks and optimum data dissemination strategies for the proposed model are the future scope of the work.

76

CHAPTER 7: CONNECTING THE DOTS TOWARDS SELF-DRIVING NETWORKS

The cloud data centers are going through an unprecedented growth from past few years. In an era of real-time video streaming, on-demand gaming, door-step e-commerce services, and highly inter-connected social networks, cost-effective service models, adaptive resources provisioning and upfront applications availability contribute significantly towards such a stellar growth. However, there are many challenges that must be addressed in a systematic manner to meet the requirements of increasingly demanding current and upcoming applications of the cloud computing paradigm. Optimum resources management, instant response time, interoperability among a diverse set of emerging technologies and innovative applications are a few of these challenges. On the other hand, the recent trend in softwarization of networks, particularly enabled by network function virtualization (NFV) and software-defined networking (SDN) principles, provides immense opportunities to better utilize the network resources by programmable abstractions with an efficient control and management techniques. Furthermore, machine learning based solutions are gaining prominence in resource optimization problems and autonomous systems. Therefore, in this chapter, we strive to connect the dots by state-of-the-art methodologies in networking and machine learning domains and utilize these developments to grapple with the challenges of the cloud-based systems. We propose DeepSDN, an SDN-based solution that harnesses existing machine learning techniques to move a step closer towards self-driving networks. The comparative results obtained from an experimental testbed corroborates effectiveness of our approach and suggest a way forward towards autonomous network management.

7.1 Introduction

No matter how far we progress in the networking research, there seems to have few challenges that remain at least partially unsolved. Part of it is due to the fact that other related technologies are also evolving in-line or faster than networking. Additionally, our own expectations as end users are also pushing the limits with the emergence of new technologies and devices. Heterogeneity, scalability, resources management, quality of service (QoS), high availability, are few such challenges that stay with us for a long time. Integration of augmented or virtual reality (AR/VR) platforms with a rapid surge of mobile hand-held and wearable smart devices equipped with omnipresent sensors are introducing an entirely new class of challenges in networking. D. Clark et al. emphasized that the current Internet architecture cannot sustain the futuristic demands, and suggested to build a network that can assemble, reassemble, and heal itself without any external intervention [27]. They further argued that such a network cannot be built incrementally with contemporary techniques. Therefore, they proposed a new construct, called Knowledge Plane, that relies on AI techniques and cognitive systems to build and maintain high-level models of the network. However, the vision of Knowledge Plane did not take off for practical implementations mainly due to the sheer complexity of inherently distributed networks and lack of suitable AI techniques specifically designed to realize such a goal. However, recent developments of network softwarization and other related themes change the trend in the networking domain. Particularly, software-defined networks (SDN) and network function virtualization (NFV), are significantly pushing boundaries of network innovation. Logically centralized control, programmability, and global network view provided by SDN principles addresses complexity and simplifies network management [89]. Similarly, emerging techniques of AI such as machine learning and deep learning are being employed beyond Computer Vision and Robotics domains.

In simplest terms, machine learning is a domain of artificial intelligence that deals with selecting an

78

appropriate response in a particular situation. Choosing the next move in games like Chess, Deep Blue, Go etc. is a classic example. In the case of networking, optimum resources allocation, load balancing, and making recommendations for personalized user experience are few considerable candidates. Broadly speaking, the learning approach can be relevant for any problem that requires a number of decisions to satisfy an objective function. Such techniques are especially favorable for the problems that are hard to solve otherwise due to lack of precise modeling, inevitable trade-off for accuracy to deal with complexity, and scalability concerns. Additionally, high replication rate of these techniques provides an opportunity to quickly expand the system in similar domains once the base model is ready and optimized up to a certain level.

However, the success of learning-based approaches is highly dependent on the availability of huge data to learn from past experiences. Moreover, it is a compute-intensive task that requires a lot of resources to gain a certain level of accuracy. The availability of a vast amount of raw and meta-data due to the repetitive nature of cloud services along with virtually unlimited resources in cloud data centers make the learning techniques well suitable for such an environment. And our reliance on clouds for essentially all sorts of digital activities probably makes them the most appropriate platform to be taken care of. Therefore, in this chapter, we coupled SDN principles with recent advances in AI and introduce a framework that incorporates intelligence capabilities to the platform by leveraging state-of-the-art networking technologies along with machine learning and data analytics techniques, which aims to grapple with challenges of cloud computing. Following are the major contributions of this chapter:

- We present the design, implementation, evaluation of DeepSDN for data center networks. A practical deployment of our proposal shows that we are moving a step closer towards letting networks make many decisions on its own.

- A revised parameter server is devised by utilizing SDN principles with an existing parameter

79

server architecture for distributed machine learning algorithms.

- We also highlight some practical applications of DeepSDN.

- Finally, experimental results obtained from the CloudLab testbed shows the effectiveness of DeepSDN.

Rest of the chapter is organized as follows. Related work is discussed in Section 7.2. High-level system overview and the detailed architectural design is presented in Sections 7.3 and 7.4 respectively. Implementation details of our proposal are demonstrated in Section 7.5. Then, the platform evaluation and comparative results are elaborated in Section 7.6. Finally, Section 7.7 outlines some noteworthy applications of the proposal followed by the conclusion and future scope of the work in Section 7.8.

7.2 Related Work

Our work is mainly based on the concept of the Knowledge Plane for Internet [27] and a distributed framework, called parameter server, for highly scalable deep learning models [28]. We extended these concepts with SDN principles and propose a practical framework for cloud data centers.

The crucial challenges of the ever-growing network control and management are discussed in [20], that also outlines the blueprint towards self-driving networks. This work also stresses the suitability of statistical inference and machine learning techniques for prediction problems. DeepRM describes a deep reinforcement learning solution that translates the network resource management problem into a learning problem [21]. It mentions that resources management problems are ubiquitous in computer networks and the prevailing wisdom of using heuristics is not a suitable way to solve such problems mainly due to lack of accuracy, flexibility, and complexity. Therefore,

DeepRM considers learning approaches to tackle these problems and shows promising results compared with heuristics. However, DeepRM does not consider these techniques for large-scale distributed systems. [22] presents a machine learning based tool called WISE that is capable of predicting the effects of probable configuration and deployment changes in content distribution networks. It shows that WISE can accurately predict service response time of globally distributed CDN nodes. It also highlights the challenges to prepare datasets for training machine learning models and presents effective solutions to address these challenges.

A new paradigm called "Knowledge Defined Networking" is proposed in [23], that integrates machine learning, data analytics and SDN concepts to the existing Internet architecture. Similar to our approach, it also posits that SDN and other recent developments enable practical implementation of the Knowledge Plane. However, it evaluates the prototype using simple simulations without assessing scalability and other issues for data centers. COBANETS utilizes unsupervised deep learning, probabilistic models, and network virtualization techniques for network optimization [24]. It shows that the generative deep neural networks can be used to extract context representations and can be combined with other machine learning techniques for otherwise complex network management task. ANEMA is an autonomous network management architecture with self-optimization and self-healing properties that achieve autonomic behaviors in the network components [25]. It formulates multiple policies to express objectives of network administrators. However, it relies on the utility function theory, and hence, does not employ learning techniques for autonomous network management. A predictive resource scaling mechanism for cloud systems is proposed in [26], that addresses SLA violations and over-provisioning of resources. Our model follows a broader approach, that includes learning, decision-making, and self-optimization capabilities.

7.3 System Overview

Motivated by the conceptual proposal of the Knowledge Plane and recent advances in respective domains in networking and AI, we present a practically deployable framework for data centers. Figure 7.1 shows the high-level architecture of DeepSDN that endeavors to fulfill an objective towards self-driving networks. The control and forwarding planes of SDN are augmented with an intelligence layer to enable learning capabilities within existing networks. The cloud and network controllers perform traditional control and management functionality as described in previous works.

Clark et al. mentioned that "there is no way for the operator to express, or the network to model, what the high-level goal of the operator is, and how the low-level decisions relate to that high-level goal." DeepSDN fills this gap by leveraging capabilities of the Intent-based API that allows operators to express high-level goals in simplistic terms. The control plane maintains the global network view and complements the intelligence layer for cognitive decision making. The intelligence layer is a unified system that engages knowledge-based functionality for decision making. The following section describes the functional building blocks of DeepSDN.

7.4 Architecture Design Details

Our goal is same as the initial vision of the Knowledge Plane, i.e., "the ability of the network to know what it is being asked to do, so that it can more and more take care of itself, rather than depending on people to attend to it." Therefore, we retain the blueprint of the Knowledge Plane by incorporating the earlier proposed attributes such as global perspective, compositional structure, and cognitive framework in DeepSDN. Figure 7.2 shows the detailed design of the intelligence layer.
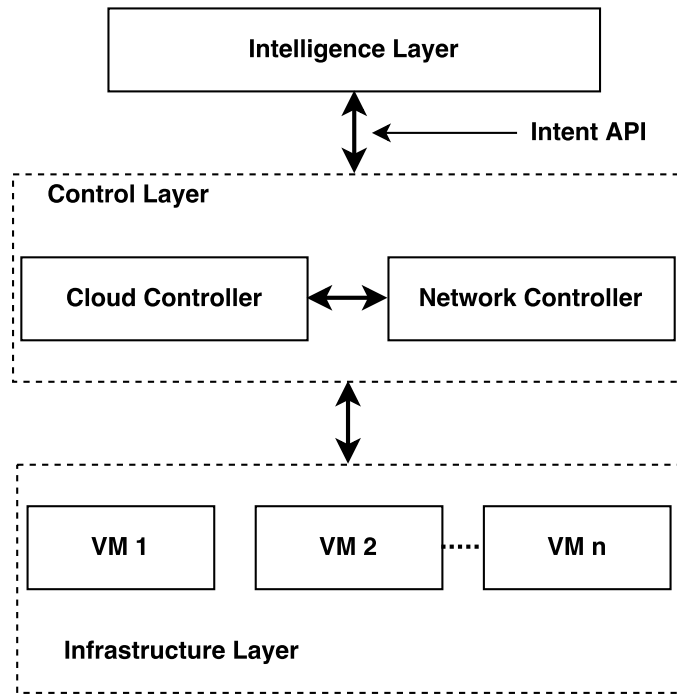
82

Figure 7.1: High-level overview for the SDN-based self-driving networks
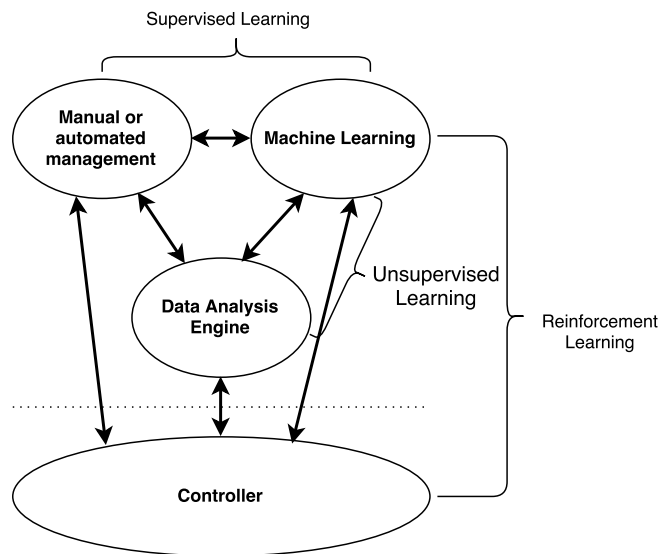
Figure 7.2: Detailed design of the intelligence layer

7.4.1 Intelligence Layer

Similar to the Knowledge Plane, the intelligence layer is a separate construct that complements control plane and data plane of SDN paradigm. The proposed model operates in the supervised learning mode when inputs/outputs and intended actions are manually defined. On the other hand, with internal state representation, it operates in unsupervised mode by learning itself with an assistance from the data analysis engine and the control plane. Similarly, it operates in reinforcement learning mode when it plays with various configurations of the system and learns from rewards of the actions taken. Additionally, with the internal state representation in place, the supervised mode operates in tandem with the unsupervised mode. The deep learning incorporates multiple layers of representation with non-linear modules that can be complemented with the supervised learning techniques to produce training data instances from the available raw data [90].

The supervised learning is rooted on well-defined criteria, systematic experimental model, and labeled data, whereas, reinforcement learning is based on uncertainty. Reinforcement learning is a specific machine learning technique that allows the learning algorithms to make autonomous decisions and optimize the system by gaining experience through interactions, observation, and feedback in terms of rewards. The intrinsic characteristic of reinforcement learning is that it achieves the objective function by interaction between the learning algorithm and the environment it operates on. Reinforcement learning is proved to be effective along with deep neural networks [91], which shows that the learning agents may optimize their goals by generalizing the past experience to tackle new situations with only very limited knowledge. Essentially, the goal of reinforcement learning techniques is to satisfy or optimize the reward function [92]. Therefore, to get desirable results, it is very important to define the reward function very clearly. By keeping manual control in the loop, we try to address this issue by letting operators or end-users to define their own reward function as per appropriate requirements. However, there is a scope to further refine this operation.

The unsupervised learning operates on the unlabeled data for the learning purpose and it does not follow the action-reward-feedback loop.

The reinforcement learning techniques can either be model-based or model-free, the former being slower and compute-intensive but more accurate and effective, whereas, the latter being faster and inexpensive but slow learners [93]. However, the model-based learning can be optimized by utilizing insights from supervised learning that provides feedback based on instant actions and corresponding effects, which expedite the iterative transitioning for better planning and estimation. It is a challenging task to come up with a practical reinforcement learning solution that operates online, i.e., learn by continuous communication between the learner and an environment in a real-time scenario. This line of research is a future scope of the work.

7.4.2 Revised Parameter Server

The parameter server is a distributed machine learning framework that maintains the global parameters and state in server nodes and distributes the workload over worker nodes [94]. The salient features of the framework include asynchronous communication, flexible consistency models, scalability, fault tolerance, and support for diverse machine learning algorithms. The parameter server framework is specifically designed for cloud computing environments considering machines unreliability, data loss, and performance fluctuations due to unpredictable network latency and varied workloads. Furthermore, vectors and matrices data structures are used to represent shared parameters, considering that linear algebra data types are more convenient for machine learning applications. For a detailed description, we refer readers to the initial proposal [94] and an improved version of parameter server [28]. Some other implementations of the parameter server architecture are also presented in [95] and [96].

As shown in Figure 7.3, we design the revised parameter server (RPS) by utilizing SDN capability

85

in distributed machine learning scenario. More precisely, we refactored the "server" functionality of parameter server to the global network controller and kept everything else essentially the same as the original proposal. Given that the SDN control plane already has global configuration information and other network-wide statistics, it makes sense to utilize this feature towards streamlining the overall platform for cloud applications. Other than that, all other functionality is kept pretty much the same, such as range-based non-blocking push and pull operations for data communication, the capability to execute user-defined functions in the controller platform, user-defined filters, optimized parallelization including data and model parallelism, elastic scalability and robustness. The consistent hashing technique is adapted from the original proposal to store the parameters in the controller.

RPS is designed for general-purpose large-scale learning techniques that are particularly well-suited for the iterative computations inherent in deep networks training. Like earlier proposal, RPS is also capable of running multiple algorithms simultaneously, that is accomplished by partitioning worker nodes into groups. However, a worker manager is also included to handle localized events within a group. It is also responsible to manage local statistics, monitor workers activity, and dynamically schedule jobs for workers. Moreover, the worker manager of each group coordinates with the controller to maintain a consistent view of the system that is handled by the resource manager at the global level. Worker groups get an updated copy of parameters from the servers and send back the calculated gradients to the corresponding servers that use the gradients to iterate towards the objective function. Furthermore, in reinforcement learning settings, RPS does not require any prior knowledge of the system, and it can support a diverse set of objective functions. Also, in the case of reinforcement learning, the workers operate as agents that learn to make decisions by receiving rewards based on the self-controlled actions taken to perform a task. The agents observe versatile characteristics of the system during training sessions and prepare a structured log of objects or events for future reference. Finally, independent parameter namespace support is also

86

incorporated to increase parallelization and other purposes.
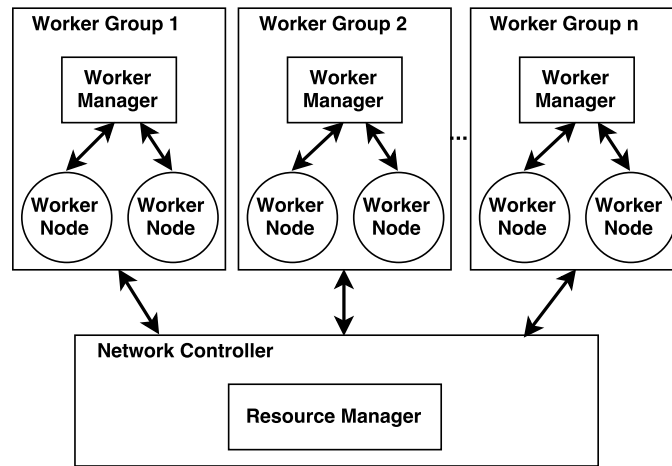
Figure 7.3: The Revised Parameter Server

7.4.3 Data Analysis Engine

The data analysis engine is used to process raw data so that the machine learning techniques can extract relevant features from the training data to achieve the objective function. In general, the learning algorithms start from the initial state and iteratively converge towards an optimal solution that is defined by the objective function. Considering that the raw data may have been collected in months and years, and may consist of terabytes to petabytes, the analysis engine is included as a separate component for scalability and high performance. Additionally, since accuracy and convergence rate of the learning algorithms is highly dependent on training data, therefore, the analysis engine is solely responsible for this task. In case of unsupervised learning, when sufficient training data examples are unknown or unavailable, the analysis engine processes raw data to find a pattern or structure to be used by the algorithms. And the parameters are used to communicate an estimate of current sample set for the generative model of the problem in hand.

7.4.4 Control Layer

The control plane oversees a set of forwarding devices. In SDN context, handling state distribution is a major task of the control plane. We harness the global view and logically centralized aspects of SDN control plane to design an integrated system based on the parameter server for distributed machine learning techniques. Particularly, we exploited the control plane for coordinating various parameters of worker groups by moving the "server" semantics of the parameter server to the logically centralized controller. Furthermore, the open-ended programmatic interface of the control plane is utilized to build management as well as learning-based applications. Figure 7.4 shows the detailed design of the control layer.

```
┌──────────────────────────────────────────────────┐
│  ┌────────────────────────────────────────────┐  │
│  │          Global Network Controller          │  │
│  └────────────────────────────────────────────┘  │
│        ↕             ↕              ↕             │
│  ┌──────────┐  ┌──────────┐   ┌──────────┐       │
│  │Instance 1│  │Instance 2│ ··│Instance n│       │
│  └──────────┘  └──────────┘   └──────────┘       │
└──────────────────────────────────────────────────┘
        ↕             ↕              ↕
  ┌──────────┐  ┌──────────┐   ┌──────────┐
  │ Server 1 │  │ Server 2 │   │ Server n │
  │ ┌──────┐ │  │ ┌──────┐ │ ··│ ┌──────┐ │
  │ │ VM 1 │ │  │ │ VM 2 │ │   │ │ VM n │ │
  │ └──────┘ │  │ └──────┘ │   │ └──────┘ │
  └──────────┘  └──────────┘   └──────────┘
```
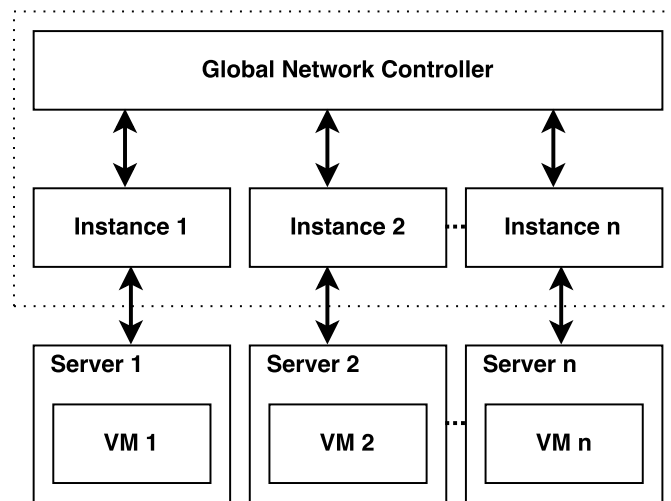
Figure 7.4: Detailed design of the network control

The global view is constructed by topology and state information gathered by each controller instance. The database maintains the consistent data store in a key-value structure for distribution and persistence. With a hierarchy of the global controller and multiple local instances, the control plane is logically centralized by physically distributed architecture for scale-out performance

in cloud data centers [89]. Multiple controller instances are individually responsible to handle a group of workers and multiple forwarding devices. Additional controller nodes can be dynamically provisioned as forwarding plane or workers load increases. The global view and the data store together serve as a glue between the worker groups and the clusters of servers. Due to the asynchronous approach, the servers and workers run independently and in parallel. The controller has a complete and up-to-date state of parameters.

The vast amount of raw data can swamp shared resources of the control plane. Therefore, partitioning and aggregation are used as complementary techniques for scalability and to avoid saturation of resources. Specifically, partitioning is preferred over replication so that the control applications can run on multiple independent instances without overwhelming the capacity. Similarly, aggregation of resources in a cluster of servers and a group of workers enables operations that are too costly to execute otherwise on an individual component. The other aspects of the control plane are implemented as software modules explained below:

Monitor manager: The desired level of consistency in the network state is dependent on the real-time topology changes and other events in the network. Additionally, state management is needed to implement elastic resources provisioning on user demand. To achieve these objectives, the local controller instances collect statistics from their respective zones and submit reports to the global controller. Subsequently, the global controller uses this information to maintain the global network state and topology.

Storage manager: The network state information is stored and handled by the storage manager. The physical centralization of the network topology information alleviates the overheads of the distributed data stores or shared file systems and simplifies high consistency of network management from a vantage point.

Events manager: The events manager is implemented to handle the events generated by either users or other network components. It also coordinates with the global controller to notify the

89

events.

Application manager: Users and network administrators implement network logic in the form of applications and submit to the controller for eventually deploying the sought functionality in the forwarding devices.

Communication manager: To mitigate latency and other overheads, the communication manager is defined to manage the peer-to-peer interaction among local controllers, as well as communications with the global controller at the top layer and switches at the bottom layer via southbound channels.

Failure manager: Failure of even a single component can have a cascade effect on the entire network that can cause severe damages to services and resources. Therefore, the failover manager is designed to handle failures of one or additional components. In case of a controller instance failure, the overall system continues operating by load distribution of the failed node and deployment of additional nodes if required. Failure manager is responsible to detect and address failures. It also provides runtime extensibility to the system as dynamic resources allocation is a significant tenet of cloud computing. The global controller has a backup and restore provision in clouds to handle unexpected events.

Software modules of DeepSDN are programmed as loosely-coupled components that can be dynamically integrated with the core system without compromising dependencies among running modules.

7.4.5 Intent-based API

The goal of an intent-based API is to let network operators and administrators define *what* they intend the network to do without specifying that exactly *how* to get it done [97]. The underlying network modules implement the desired state via low-level policies without any manual interven-

90

tion. The intent API maps high-level commands of network operators to low-level instructions, formulate the relevant policies for network components, and create the intended state by enforcing these policies. Furthermore, to maintain the state, intent framework constantly monitor the network to detect and resolve suboptimal performance or any policy conflict due to high load of components failure.

7.5 Implementation

To demonstrate the usability of DeepSDN, we implemented the Downpour SGD algorithm presented in [94] that is a variant of asynchronous stochastic gradient descent (SGD) for large-scale training models and datasets. The traditional SGD technique is a sequential method for training deep neural network that is commonly used for optimization problems. Downpour SGD is a high-performance distributed learning algorithm that runs iteratively in the independently operating worker nodes. The compute-intensive task of gradient calculation is divided among all of the workers and the results are collected by the parameter servers of the respective workers.

ONOS serves as a control plane of DeepSDN that controls and manages the forwarding devices as well as functions as the parameter server for worker nodes. ONOS consists of many salient features for optimization such as low-latency data store, optimized data model, caching layer to reduce communication overheads, the in-memory topology view, network I/O optimization, several event channels for events notifications, and a network API specifically designed for network applications [45]. Other than these features, ONOS also provides inbuilt support for the Intent-based API that makes it a very suitable choice for DeepSDN implementation. Therefore, we extended ONOS code-base by implementing Downpour SGD algorithm and making it compatible with the revised parameter server architecture described earlier.

To provide reliability, the design techniques of ONOS are adapted to handle failures of multiple components. Zookeeper [98] is used for coordination of consensus-based distributed algorithms, membership management, and failure handling. Furthermore, the keepalive and hardware-based probing techniques are employed to monitor various components and links of the system. Additionally, the control applications can either rely on Zookeeper functions or implement their own techniques to detect instance failures.

7.6 Evaluation Platform and Results

To test the validity of our proposal, we designed a deep neural network with a fully connected hidden layer with 50 neurons and a sigmoid activation function that operates on approximately 100,000 parameters to achieve the objective function. The training datasets are prepared from the raw traffic traces collected by deploying a folded clos topology [99] using the CloudLab testbed platform [100]. We deployed 200 workers and 25 servers on separate machines, and each machine is Intel Xeon E5-2603 with 6 cores, 128 GB RAM, 4 TM hard drive and 10 Gb Ethernet connectivity. Furthermore, we used 10,000 samples to train the deep neural network and additional 1,000 samples to validate the results.

For each experiment, the monitor manager of DeepSDN captures traffic and export raw data in the tcpdump format for further processing to prepare the training data. However, the data collected by the monitor manager is not suitable to be directly used by the training models. Therefore, further processing and transformations are necessary to extract relevant features from the raw data. Therefore, for the datasets to be representative and consistent with the experiments, the techniques mentioned in WISE [22] are employed to transform the raw data into the training data that is understandable to the training model. Specifically, dependency structures, pair-wise independence testing, and indexing techniques are followed from WISE. Furthermore, as mentioned

92

in [22], cross-validation is inherently unscalable to deal with overfitting, therefore data decomposition and pruning techniques are used that produces evenly distributed training datasets across multiple buckets. During raw data collection, multiple delays including processing delay, queuing delay, propagation delay, were taken into consideration. Additionally, link-capacity, distance, 5-tuple headers, packet count, traffic volume, routing parameters, and response time were used to prepare the training datasets.

Accurate performance prediction is among the primary goals of DeepSDN. Therefore, we evaluated estimation of response time metric that is produced by the reinforcement learning approach based on training from the existing dataset. Figure 7.5 shows that our model provides a relatively good estimation of response time as compared to the actual metric. However, we believe that the slight inaccurate prediction arises due to insufficient training data.
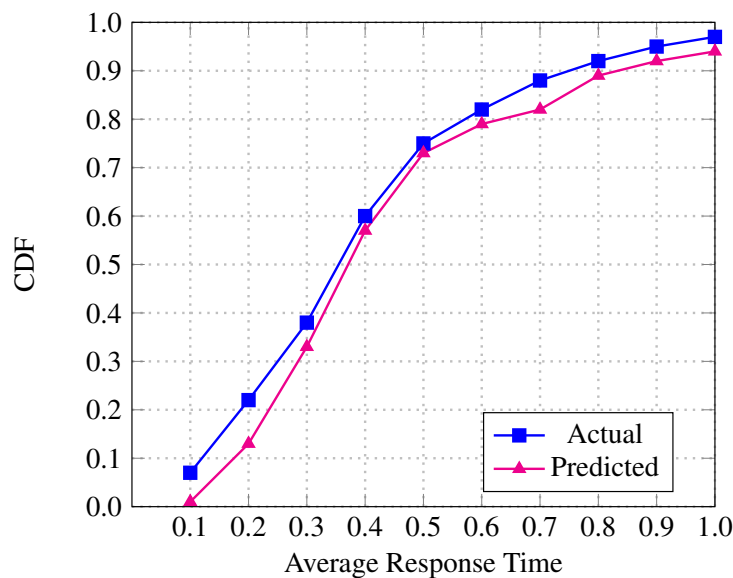

Figure 7.5: Comparison of an average response time

To further analyze the impact of training data on prediction accuracy, we performed experiments with a varied number of training samples. As evident in Figure 7.6, the estimation accuracy is

93

significantly dependent on the training dataset. It is also noteworthy that this dependency tends to diminish after a certain number of samples and the model starts to converge. The actual number of samples required to attain a certain level of accuracy depends on the quality of the dataset and the training model characteristics among other factors.

Next, Figure 7.7 demonstrates that the learning model improves with time and starts to converge after a certain number of iterations. As mentioned earlier, the reinforcement learning approach optimizes the prediction based on learning from the reward received in response to the action taken in the previous iteration. This process continues until the defined objective function is achieved or the optimization level is reached accordingly to the availability of dataset and training model.
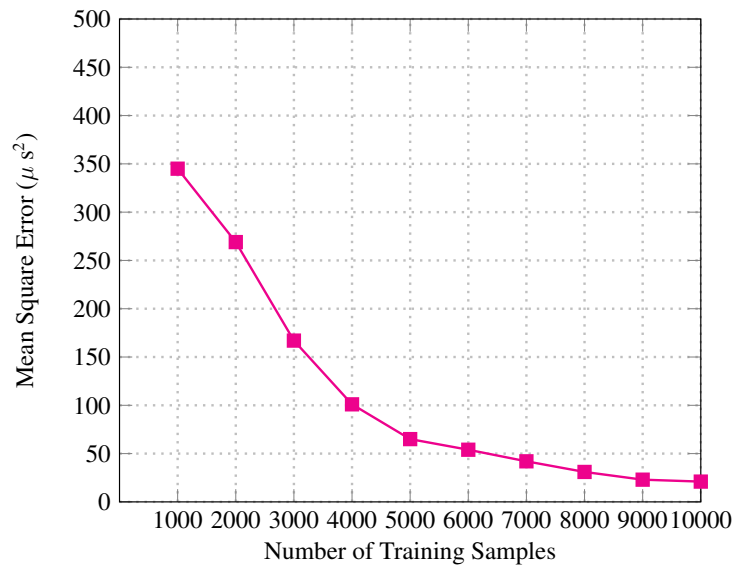
Figure 7.6: Prediction error of response time
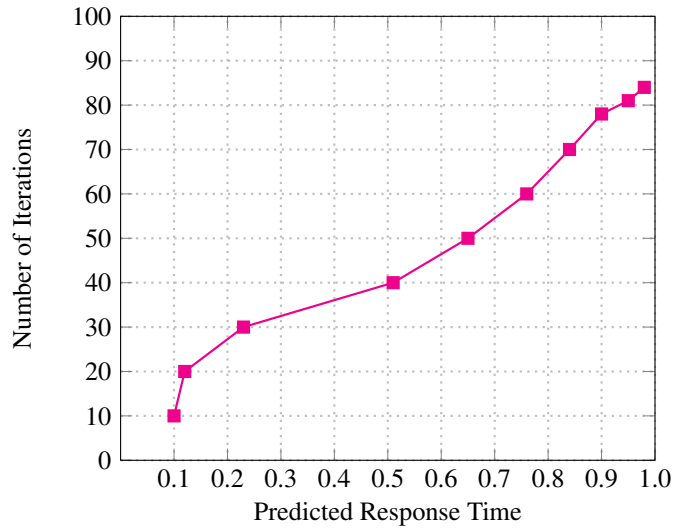
Figure 7.7: Learning accuracy w. r. t. number of iterations

Another design goal of DeepSDN is large-scale applicability for cloud data centers. To this end, we assessed the CPU consumption of our framework by increasing the number of parameter servers. We can see from Figure 7.8 that there is a steady increase in CPU utilization corresponding to parameter servers. Therefore, we conclude that the distributed architecture of DeepSDN is capable to handle large-scale learning models.

Finally, we also demonstrate the scalability of DeepSDN by evaluating the time taken by the training model with a number of parameter servers. We can observe from Figure 7.9 that the number of parameter servers plays a crucial role for training purpose. However, as highlighted in previous results, the number of servers is relevant up to a certain point, after that we do not gain the similar speedup by further increasing the number of parameter servers.

Figure 7.8: Scalability of DeepSDN

Figure 7.9: Comparison of training time

96

7.7 Notable Applications

- With DeepSDN, an intelligent orchestration engine can be deployed that decides location, availability, and timeline to offer on-demand services. Additionally, by taking the operating characteristics of servers and network congestion into consideration, resources consumption forecasting can be done that helps to define adaptive utilization policies.
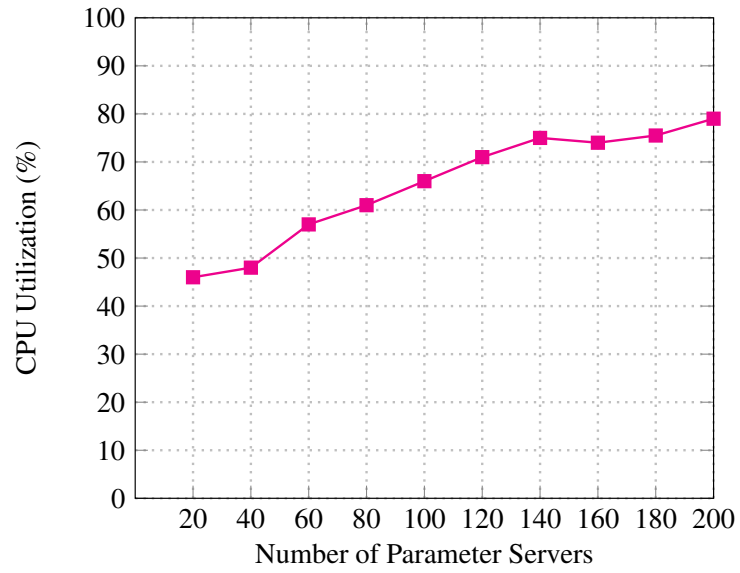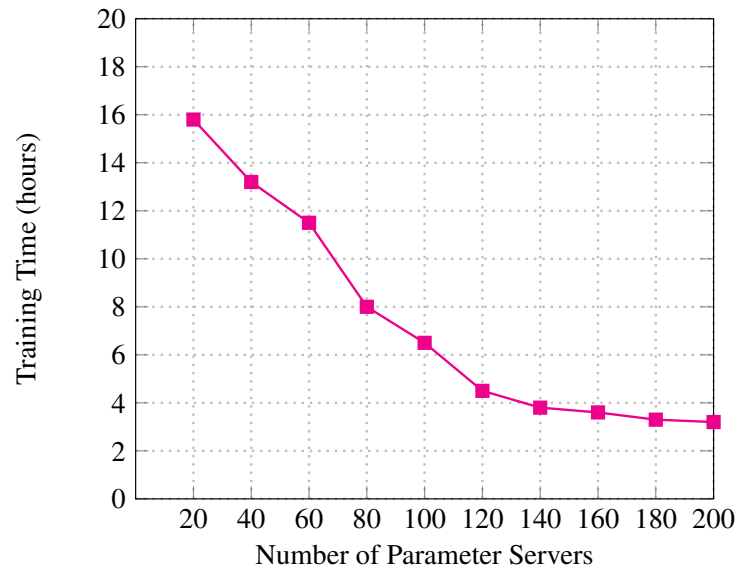
- DeepSDN also addresses the longstanding traffic measurement issue of large clouds by enabling learning-based techniques for sophisticated monitoring and statistical techniques that can be implemented based on distinct user requirements on top of the common set of APIs exposed by the system [101]. The unified framework would substantially simplify the job of network engineers and operators for the measurements task. Based on the traffic matrices and counters collected by the monitor manager, learning-based inference techniques can be developed for various purposes including dynamic provisioning, diagnosis, and optimizations.

- Machine learning techniques are successfully being employed for prediction of multiple performance metrics such as processing time, storage space, and bandwidth requirements. WISE [22] mentions that the reasoning systems such as DeepSDN can be extended to other realms such as routing, policy decisions, and security configurations.

- An adaptive machine learning-based model is proposed in [102] that predicts live migration characteristics to improve efficiency and utilization of data centers. It shows that important metrics responsible for performance degradation can be successfully modeled and, based on these predictions, machine learning techniques can be applied to mitigate SLA violations.

- As presented in [103], reinforcement learning approach can be utilized for dynamic configuration of virtual machines. The learning agent iteratively interacts with the environment

and choose optimum configuration parameters to get rewarded. The reward function can be defined according to resources utilization policies or SLA requirements.

7.8 Conclusion and Future Work

Although the vision to have intelligent networks is here from a while, only recent advances in the building blocks of such a vision have enabled us to propose a concrete framework that materializes the initial design goals. Such improvements include advanced machine learning and deep learning techniques, network softwarization, impressive processing capabilities, parallel processing techniques, and very large-scale data processing platforms. Furthermore, cloud computing provides a powerhouse of processing and ocean of data readily available to be utilized in a systematic and efficient manner. However, owing to its tremendous growth, cloud data centers also face many challenges that hinder to maintain the growth rate. Optimum resources utilization, interoperability among diverse technologies, and instant response time for seamless user experience are among such challenges. Past research suggests that these issues essentially boils down to efficient network management and optimization problems. And recent advances in machine learning and deep learning make these techniques plausible candidates for such problems. Additionally, SDN principles and abundance of physical resources with readily available raw data in data centers motivate us to consider learning techniques to grapple with challenges of cloud computing. Therefore, we strive to utilize relatively interdisciplinary techniques and propose a framework, called DeepSDN, by connecting the dots. Additionally, we extended the earlier proposed parameter server architecture for distributed machine learning problems and presents a revised parameter server that uses centralized control, global view, and programmability features of SDN for implementing the learning techniques to achieve optimized control and management. Broadly speaking, we believe that the level of complexity introduced by unprecedented scale and time-bound performance requirements

98

can be effectively handled by self-driving network management, and this chapter takes us a step closer towards this direction.

The future line of research includes consideration of other learning techniques for DeepSDN, optimizations for resources sharing, and holistic approaches to prepare amenable training datasets.

CHAPTER 8: CONCLUSION

In this dissertation, we have addressed the scalability issue of SDN and considered this nascent technology for wireless and ad-hoc networks. We have shown that having a single controller is not a feasible option for SDN deployment at large-scale, and logically centralized but physically distributed control plane is a pragmatic approach from the design and requirements perspective. Therefore, we proposed a peer-to-peer control plane architecture with a hierarchy of controllers to manage the data plane of their respective domains. The peer-to-peer interaction among adjacent controllers reduces latency and minimizes communication overheads of the control messages. Additionally, dynamical controllers provisioning allows adaptive load-balancing and provides robustness against failures. The consistent global network view is maintained with appropriate state synchronization methods that prioritize overall efficiency and usability of the network. Furthermore, the software modules are implemented to deploy such a practical control plane architecture. The evaluation results are presented to compare our proposal with existing models, which shows the effectiveness of our design decisions and optimization techniques. Some feasible use-cases of the proposal are also discussed.

We have also shown that SDN principles cannot be applied directly for wireless, mobile, and ad-hoc networks. Unstable and often unpredictable topology, sporadic connectivity, and unreliable channel conditions pose unique challenges for such environments. However, if somehow we manage to address these concerns, SDN has the potential to revolutionize the wireless domain just like its wired counterpart. For example, the centralized control would allow better spectrum sharing and QoS techniques due to having a global network perspective. Therefore, we proposed an SDN-based architecture for wireless access networks that conforms to the environment it operates in and provides opportunities for network innovation in a unique way. To do so, we borrowed the idea of the scalable control plane from our earlier work and extended it with cloud computing, virtu-

alization, and other techniques that make it suitable for access networks. The evaluation results show that we achieve equivalent or better performance as compared to other similar proposals. Additionally, we applied these techniques for vehicular ad-hoc networks (VANETs) and showed their usefulnesses. Particularly, we designed and deployed routing and QoS applications on top of the controller platform of SDN and demonstrated the effectiveness of these techniques.

SDN has emerged as a significant technology to simplify network management. However, network designers still need to rely on the control platform that deals with low-level operations at the infrastructure layer. This closed-form model is highly complicated to analyze and optimize manually. On the other hand, the recent success of artificial intelligence techniques for computer vision and robotics fields has motivated the research community to contemplate these techniques for computer networks as well. More precisely, the machine learning and deep learning approaches are shown to be very effective for automation and optimization problems, especially if a large and accurate dataset is provided for training purpose. Also, these techniques require abundant computing resources to perform well under limited time constraints. We observe that the cloud data centers have a huge amount of raw or readily available data, coupled with nearly unlimited resources. And, our increasingly high reliance on clouds for all sorts of digital activities are posing serious performance challenges for data center networks. Therefore, we proposed a machine learning based platform called DeepSDN that utilizes SDN for optimizing and streamlining cloud data centers. The experimental results obtained from an experimental testbed corroborates effectiveness of our approach and suggest a way forward towards autonomous network management. However, the lack of standardized datasets and the machine learning algorithms specifically tailored for computer networks is a major concern that must be addressed to achieve this goal.

101

LIST OF REFERENCES

[1] A. A. Lazar, "Programming telecommunication networks," in *Building QoS into Distributed Systems*. Springer, 1997, pp. 3–22.

[2] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and control element separation (forces) protocol specification," Tech. Rep., 2010.

[5] D. Sheinbein and R. Weber, "Stored program controlled network: 800 service using spc network capability," *The Bell System Technical Journal*, vol. 61, no. 7, pp. 1737–1744, 1982.

[6] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 15–28.

[7] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Presented as part of the 2nd USENIX Workshop*

on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, 2012.

[8] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, pp. 3–3.

[9] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 19–24.

[10] Y. Fu, J. Bi, Z. Chen, K. Gao, B. Zhang, G. Chen, and J. Wu, "A hybrid hierarchical control plane for flow-based large-scale software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 117–131, 2015.

[11] B. Görkemli, S. Tatlıcıoğlu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic control plane for sdn at scale," *IEEE Journal on Selected Areas in Communications*, 2018.

[12] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (sdn)," *Computer Networks*, vol. 112, pp. 279–293, 2017.

[13] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling sdns," in *Software Defined Networking (EWSDN), 2012 European Workshop on*. IEEE, 2012, pp. 1–6.

[14] C. J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zúñiga, "An architecture for software defined wireless networking," *IEEE wireless communications*, vol. 21, no. 3, pp. 52–61, 2014.

[15] L. E. Li, Z. M. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Software Defined Networking (EWSDN), 2012 European Workshop on*. IEEE, 2012, pp. 7–12.

[16] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: Software defined radio access network," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 25–30.

[17] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Communications magazine*, vol. 51, no. 7, pp. 44–53, 2013.

[18] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE network*, vol. 29, no. 4, pp. 24–30, 2015.

[19] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 4–18, 2015.

[20] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," *arXiv preprint arXiv:1710.11583*, 2017.

[21] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 50–56.

[22] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering what-if deployment and configuration questions with wise," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 99–110.

[23] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, 2017.

[24] M. Zorzi, A. Zanella, A. Testolin, M. D. F. De Grazia, and M. Zorzi, "Cognition-based networks: A new perspective on network optimization using learning and distributed intelligence," *IEEE Access*, vol. 3, pp. 1512–1530, 2015.

[25] H. Derbel, N. Agoulmine, and M. Salaün, "Anema: Autonomic network management architecture to support self-configuration and self-optimization in ip networks," *Computer Networks*, vol. 53, no. 3, pp. 418–430, 2009.

[26] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems." *CNSM*, vol. 10, pp. 9–16, 2010.

[27] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 3–10.

[28] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server." in *OSDI*, vol. 1, no. 10.4, 2014, p. 3.

[29] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, pp. 392–403, 2001.

[30] M. Allman and A. Falk, "On the effective evaluation of tcp," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 59–70, 1999.

[31] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[32] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, "Cloudsimsdn: Modeling and simulation of software-defined cloud data centers," in *Cluster, Cloud and*

Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on.   IEEE, 2015, pp. 475–484.

[33] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[34] P. Kathiravelu *et al.*, "Sendim for incremental development of cloud networks: Simulation, emulation and deployment integration middleware," in *2016 IEEE International Conference on Cloud Engineering (IC2E).*   IEEE, 2016, pp. 143–146.

[35] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 15, p. 17, 2008.

[36] S. Guan, R. De Grande, and A. Boukerche, "A multi-layered scheme for distributed simulations on the cloud environment," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2015.

[37] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos, "Sddc: A software defined datacenter experimental framework," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on.*   IEEE, 2015, pp. 189–194.

[38] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "Cloudnaas: a cloud networking platform for enterprise applications," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 8.

[39] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013.

106

[40] D. Huang, X. Zhang, M. Kang, and J. Luo, "Mobicloud: building secure cloud framework for mobile computing and communication," in *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*. Ieee, 2010, pp. 27–34.

[41] A. Ahmed and A. S. Sabyasachi, "Cloud computing simulators: A detailed survey and future direction," in *Advance Computing Conference (IACC), 2014 IEEE International*. IEEE, 2014, pp. 866–872.

[42] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Cloud computing networking: challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 54–62, 2013.

[43] R. Buyya, R. N. Calheiros, J. Son, A. V. Dastjerdi, and Y. Yoon, "Software-defined cloud computing: Architectural elements and open challenges," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE, 2014, pp. 1–12.

[44] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[45] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[46] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?" in *OSDI*, vol. 10, 2010, pp. 1–6.

[47] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[48] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of open vswitch," in *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, 2015, pp. 117–130.

[49] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," *ACM SIG-COMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[50] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Computer Communications*, vol. 67, pp. 1–10, 2015.

[51] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIG-COMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 13–18.

[52] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[53] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 1–6.

[54] A. S.-W. Tam, K. Xi, and H. J. Chao, "Use of devolved controllers in data center networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. IEEE, 2011, pp. 596–601.

[55] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[56] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–8.

[57] A. A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Elasticon: an elastic distributed sdn controller," in *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*. ACM, 2014, pp. 17–28.

[58] A. Koshibe, A. Baid, and I. Seskar, "Towards distributed hierarchical sdn control plane," in *Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), 2014 International*. IEEE, 2014, pp. 1–5.

[59] Y. Liu, A. Hecker, R. Guerzoni, Z. Despotovic, and S. Beker, "On optimal hierarchical sdn," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 5374–5379.

[60] I. Ku, Y. Lu, and M. Gerla, "Software-defined mobile cloud: Architecture, services and use cases," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2014, pp. 1–6.

[61] V. G. Vassilakis, I. D. Moscholios, B. A. Alzahrani, and M. D. Logothetis, "A software-defined architecture for next-generation cellular networks," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[62] M. Rahman, C. Despins, and S. Affes, "Hetnet cloud: Leveraging sdn & cloud computing for wireless access virtualization," in *2015 IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*.   IEEE, 2015, pp. 1–5.

[63] S. Zhou, T. Zhao, Z. Niu, and S. Zhou, "Software-defined hyper-cellular architecture for green and elastic wireless access," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 12–19, 2016.

[64] E. Cau, M. Corici, P. Bellavista, L. Foschini, G. Carella, A. Edmonds, and T. M. Bohnert, "Efficient exploitation of mobile edge computing for virtualized 5g in epc architectures," in *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*.   IEEE, 2016, pp. 100–109.

[65] D. B. Rawat and S. Reddy, "Recent advances on software defined wireless networking," in *SoutheastCon, 2016*.   IEEE, 2016, pp. 1–8.

[66] J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K.-H. Kim, and T. Nadeem, "mesdn: mobile extension of sdn," in *Proceedings of the fifth international workshop on Mobile cloud computing & services*.   ACM, 2014, pp. 7–14.

[67] J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "Cloudmac: towards software defined wlans," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 4, pp. 42–45, 2013.

[68] W. Wang, Y. Chen, Q. Zhang, and T. Jiang, "A software-defined wireless networking enabled spectrum management architecture," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 33–39, 2016.

[69] K. S. Atwal and M. Bassiouni, "A novel approach for simulation and analysis of cloud data center applications," in *Proceedings of the IEEE International Conference on Smart Cloud*. IEEE, 2016, pp. 164–169.

[70] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "Openroads: Empowering research in mobile networks," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 125–126, 2010.

[71] M. Bechler and L. Wolf, "Mobility management for vehicular ad hoc networks," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 4. IEEE, 2005, pp. 2294–2298.

[72] G. Yan and D. B. Rawat, "Vehicle-to-vehicle connectivity analysis for vehicular ad-hoc networks," *Ad Hoc Networks*, vol. 58, pp. 25–35, 2017.

[73] I. Ku, Y. Lu, M. Gerla, F. Ongaro, R. L. Gomes, and E. Cerqueira, "Towards software-defined vanet: Architecture and services," in *Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean*. IEEE, 2014, pp. 103–110.

[74] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for rsu clouds in support of the internet of vehicles," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 133–144, 2015.

[75] S. Gorlatch and T. Humernbrum, "Enabling high-level qos metrics for interactive online applications using sdn," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*. IEEE, 2015, pp. 707–711.

[76] Z. He, J. Cao, and X. Liu, "Sdvn: enabling rapid network innovation for heterogeneous vehicular communication," *IEEE network*, vol. 30, no. 4, pp. 10–15, 2016.

111

[77] Z. He, D. Zhang, S. Zhu, J. Cao, and X. Liu, "Sdn enabled high performance multicast in vehicular networks," in *Vehicular Technology Conference (VTC-Fall), 2016 IEEE 84th*. IEEE, 2016, pp. 1–5.

[78] T. Kosch, C. J. Adler, S. Eichler, C. Schroth, and M. Strassberger, "The scalability problem of vehicular ad hoc networks and how to solve it," *IEEE Wireless Communications*, vol. 13, no. 5, 2006.

[79] H. Li, M. Dong, and K. Ota, "Control plane optimization in software-defined vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7895–7904, 2016.

[80] A. Guleria, K. Singh, and N. Chand, "Qos aware service scheduling scheme for vanets," *International Journal of Distributed Sensor Networks*, 2015.

[81] T. Taleb, E. Sakhaee, A. Jamalipour, K. Hashimoto, N. Kato, and Y. Nemoto, "A stable routing protocol to support its services in vanet networks," *IEEE Transactions on Vehicular technology*, vol. 56, no. 6, pp. 3337–3347, 2007.

[82] K. Katsaros, M. Dianati, R. Tafazolli, and R. Kernchen, "Clwpr a novel cross-layer optimized position based routing protocol for vanets," in *Vehicular Networking Conference (VNC), 2011 IEEE*. IEEE, 2011, pp. 139–146.

[83] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Network*, vol. 27, no. 5, pp. 48–55, 2013.

[84] Y. Wang and J. Bi, "A solution for ip mobility support in software defined networks," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–8.

[85] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, C. N. Corrêa, S. C. de Lucena, and M. F. Magalhães, "Virtual routers as a service: the routeflow approach leveraging software-defined networks," in *Proceedings of the 6th International Conference on Future Internet Technologies*. ACM, 2011, pp. 34–37.

[86] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. Van Der Merwe, "The case for separating routing from routers," in *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. ACM, 2004, pp. 5–12.

[87] C. Guimaraes, D. Corujo, R. L. Aguiar, F. Silva, and P. Frosi, "Empowering software defined wireless networks through media independent handover management," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 2204–2209.

[88] A. Böhm and M. Jonsson, "Real-time communication support for cooperative, infrastructure-based traffic safety applications," *International Journal of Vehicular Technology*, vol. 2011, 2011.

[89] K. S. Atwal, A. Guleria, and M. Bassiouni, "A scalable peer-to-peer control plane architecture for software defined networks," in *Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on*. IEEE, 2016, pp. 148–152.

[90] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[91] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[92] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, vol. 2. Phoenix, AZ, 2016, p. 5.

[93] M. L. Littman, "Reinforcement learning improves behaviour from evaluative feedback," *Nature*, vol. 521, no. 7553, p. 445, 2015.

[94] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[95] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[96] L. Luo, J. Nelson, L. Ceze, A. Phanishayee, and A. Krishnamurthy, "Parameter hub: a rack-scale parameter server for distributed deep neural network training," *arXiv preprint arXiv:1805.07891*, 2018.

[97] S. Arezoumand, K. Dzeparoska, H. Bannazadeh, and A. Leon-Garcia, "Md-idn: Multi-domain intent-driven networking in software-defined infrastructures," in *Network and Service Management (CNSM), 2017 13th International Conference on*. IEEE, 2017, pp. 1–7.

[98] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems." in *USENIX annual technical conference*, vol. 8, no. 9. Boston, MA, USA, 2010.

[99] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," *Communications of the ACM*, vol. 54, no. 3, pp. 95–104, 2011.

[100] A. Akella, "Experimenting with next-generation cloud architectures using cloudlab." *IEEE Internet Computing*, vol. 19, no. 5, pp. 77–81, 2015.

114

[101] G. Varghese and C. Estan, "The measurement manifesto," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 9–14, 2004.

[102] C. Jo, Y. Cho, and B. Egger, "A machine learning approach to live migration modeling," in *Proceedings of the 2017 Symposium on Cloud Computing*. ACM, 2017, pp. 351–364.

[103] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin, "Vconf: a reinforcement learning approach to virtual machines auto-configuration," in *Proceedings of the 6th international conference on Autonomic computing*. ACM, 2009, pp. 137–146.